ROYAL INSTITUTE
OF TECHNOLOGY

Analysis and Management of Communication in On-Chip Networks

FAHIMEH JAFARI

Doctoral Thesis in Electronics and Embedded Systems
KTH Royal Institute of Technology
Stockholm, Sweden 2015

TRITA-ICT/ECS AVH 15:01
ISSN 1653-6363
ISRN KTH/ICT/ECS/AVH-15/01-SE
ISBN x-xxxx-xxx-x

KTH School of Information and
Communication Technology
Department of Electronic Systems
SE-164 40 Kista SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie licentiatesexamen i datalogi Torsdag den 19 February 2015 klockan 13:00 i sal E i Forum IT-Universitetet, Kungl Tekniskahögskolan, Isajordsgatan 39, Kista.

Tryck: Universitetsservice US AB

Abstract

Regarding the needs of low-power, high-performance embedded systems and the growing computation-intensive applications, the number of computing resources in a single chip has enormously increased. The current VLSI technology is able to support such an integration of transistors and add many computing resources such as CPU, DSP, specific IPs, etc to build a System-on-Chip (SoC). However, interconnection between resources becomes another challenging issue which can be raised by using an on-chip interconnection network or Network-on-Chip (NoC). NoC based communication which allows pipelined concurrent transmissions of transactions is becoming a dominate infrastructure for many core computing platforms.

This thesis analyzes and manages both Best Effort (BE) and Guaranteed Service (GS) communications using analytical performance approaches. As the first step, the present thesis focuses on the flow control for BE traffic in NoC. It models BE source rates as the solution to a utility-based optimization problem which is constrained with link capacities while preserving GS traffic services requirements at the desired level. Towards this, several utility functions including *proportionally-fair*, *rate-sum*, and *max-min fair* scenarios are investigated. Moreover, it is worth looking into a scenario in which BE source rates are determined in favor of minimizing the delay of such traffics. The presented flow control algorithms solve the proposed optimization problems determining injection rate in each BE source node.

In the next step, real-time systems with guaranteed service are considered. Real-time applications require performance guarantees even under worst-case conditions, i.e. Quality of Service (QoS). Using network calculus, we present and prove the required propositions for deriving performance metrics and then apply them to propose formal approaches for the worst-case performance analysis. The proposed analytical model is used to minimize total cost in the networks in terms of buffer and delay. To this end, we address several optimization problems and solve them to consider the impact of various objective functions. We also develop a tool which derives performance metrics for a given NoC, formulates and solves the considerable optimization problems to provide an invaluable insight for NoC designers.

Dedicated to my lovely parents, Mehri and Mohammad Reza,

who I am always indebted to

to my love, Abbas, and my little angel, Tina,

who have given me more than they will ever know.

Acknowledgements

First and foremost I would like to express the deepest appreciation to my former supervisor (current co-supervisor), Prof. Axel Jantsch, for his invaluable support both in my research and life during all years of my work in ICT school, even after he moved from KTH to the University of Vienna. I have been extremely fortunate to work under his supervision and I deeply appreciate his time, patience, and support throughout my Ph.D. career. His assistance, constructive comments, and technical insights have always been most helpful in addressing my research issues. I would also like to thank Prof. Ahmed Hemani for the support and advice he has provided as my supervisor after moving Prof. Axel Jantsch from KTH.

I would like to express my special thanks to my former co-supervisor, Assoc. Prof. Zhonghai Lu, for giving me generous amount of time whenever I needed some help. At many stages in the course of my research, I benefited from his advice, suggestions, and meticulous comments particularly so when exploring new ideas. His positive outlook and confidence in my research inspired me and gave me confidence. Also, I am particularly grateful to Assoc. Prof. Ingo Sander for reviewing my thesis.

I owe my deepest gratitude to all of my family members, especially my parents who have done everything for me, including sacrificing the joys of their own lives so that I could be happy and successful in my life. Thanks for being with me on each and every step of my life. It is their unconditional love that motivates me to set higher targets.

My heartfelt appreciation goes to my beloved husband, Abbas Eslami Kiasari, who experienced all of the ups and downs of my career during the last twelve years. Without his love, continued support and warm encouragement, I could not pursue my study and also this thesis would not have been possible. Last, but not least, I would also like to thank my lovely friends who have contributed immensely to my personal and professional time in Sweden.

Contents

List of Figures

List of Tables

List of Publications

- **Thesis Publications**
  - **Conference Proceedings**
    1. M. S. Talebi, F. Jafari, and A. Khonsari, "A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization". *In the Proceedings of the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 381-386, Istanbul, Turkey, October 2007.

    2. M. S. Talebi, F. Jafari, A. Khonsari, and M. H. Yaghmaee, "A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization". *In the Proceedings of the International Conference on Computational Science and its Applications (ICCSA)*, pp. 398-409, Kuala Lumpur, Malaysia, August 2007.

    3. M. S. Talebi, F. Jafari, A. Khonsari, and M. H. Yaghmaee, "Proportionally-Fair Best Effort Flow Control in Network-on-Chip Architectures", *In the Proceedings of the International Workshop on Performance Modeling, Evaluation, and Optimization of Ubiquitous Computing and Networked Systems (PMEO UCNS), in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Miami, Florida, USA, April 2008.

    4. F. Jafari, M. S. Talebi, A. Khonsari, and M. H. Yaghmaee, "A Novel Congestion Control Scheme in Network-on-Chip Based on Best Effort Delay-Sum Optimization", *In the Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pp. 191-196, Sydney, NSW, Australia, May 2008.

    5. F. Jafari, M. H. Yaghmaee, M. S. Talebi, and A. Khonsari, "Max-Min-Fair Best Effort Flow Control in Network-on-Chip Architectures", *In the Proceedings of the International Conference on Computational Science (ICCS)*, Part I, LNCS 5101, pp. 436-445, Krakow, Poland, June 2008.

    6. F. Jafari and M. H. Yaghmaee, "A Novel Flow Control Scheme for Best Effort Traffics in Network-on-Chip Based on Weighted Max-Min-Fairness",

In the Proceedings of International Symposium on Telecommunications (IST), pp. 458-463, Tehran, Iran, August 2008.

7. F. Jafari, M. S. Talebi, M. H. Yaghmaee, and A. Khonsari, "Throughput-fairness tradeoff in Best Effort flow control for on-chip architectures", *In the Proceedings of the International Workshop on Performance Modeling, Evaluation, and Optimization of Ubiquitous Computing and Networked Systems (PMEO UCNS), in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Rome, Italy, May 2009.

8. F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Optimal Regulation of Traffic Flows in Network-on-Chip", *In the Proceedings of the Design Automation & Test in Europe (DATE)*, pp. 1621- 1624, Dresden, Germany, March 2010.

9. F. Jafari, A. Jantsch, and Z. Lu, "Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling", *In the Proceedings of the International Conference on Computer Design (ICCD)*, pp. 445-446, Amherst, USA, October 2011.

10. F. Jafari, A. Jantsch, and Z. Lu, "Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling", *In the Proceedings of the Design Automation & Test in Europe (DATE)*, pp. 538-541, Dresden, Germany, March 2012.

- **Journal Papers**

  Accepted

  11. M. S. Talebi, F. Jafari, A. Khonsari, and M. H. Yaghmaee, "Proportionally Fair Flow Control Mechanism for Best Effort Traffic in Network-on-Chip Architectures", *International Journal of Parallel, Emergent, and Distributed Systems (IJPEDS)*, Vol. 25, No. 4, pp 345-362 Jul. 2010.

  12. F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Buffer Optimization in network-on-Chip through Flow Regulation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Vol. 29, No. 12, pp 1973-1986, Dec. 2010.

  Submitted

  13. F. Jafari, Z. Lu, and A. Jantsch, "Least Upper Delay Bound for VBR Flows in Networks-on- Chip with Virtual Channels", *Submitted to ACM Transactions on Design Automation of Electronic Systems (TODAES).*

14. F. Jafari, A. Jantsch, and Z. Lu, "Weighted Round Robin Configuration for Worst-Case Delay Optimization in Network-on-Chip", *Submitted to IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems (TCAD).*

- **Other Publications**

15. F. Jafari, S. Li, and A. Hemani, "Optimal Selection of Function Implementation in a Hierarchical Configware Synthesis Method for a Coarse Grain Reconfigurable Architecture", *In the Proceedings of the Euromicro Conference on Digital System Design (DSD)*, pp. 73-80, Oulu, Finland, August 2011.

16. S. Li, F. Jafari, A. Hemani, and S. Kumar, "Layered Spiral Algorithm for Memory-Aware Mapping and Scheduling on Network-on-Chip", *In the Proceedings of the NORCHIP conference*, pp. 1-6, Tampere, Finland, November 2010.

Part I

Introduction

Chapter 1

Introduction

The scope and direction of this thesis are indicated in this chapter. At first, an introduction of NoC structure is considered and communication management is mentioned as a research challenge. Then, the contributions are presented and finally the organization of the rest of the thesis is given.

1.1 On-Chip Interconnection Networks

Progresses in deep sub-micron technology have led to integrate hundreds of IP cores running multiple concurrent processes on a single chip. Although the speed of elements in such systems becomes faster, the International Technology Roadmap for Semiconductors (ITRS) illustrates that the wiring delay is growing exponentially because of the increased capacitance caused by narrow channel width and increased crosstalk. Therefore, the wiring and consequently communication between cores is one of the main limiting factors to be concerned.

As shown in Figure 1.1, bus-based architectures and point-to-point communication methodologies are some prevailing mechanisms for communication between several cores in System-on-Chip (SoC). However, these architectures have fundamentally some limitations in bandwidth, i.e. while the number of components attached to them is increased, physical capacitance on the wires grows and as a result its wiring delay grows even further. Therefore, as the number of cores keeps increasing, neither traditional bus-based nor point-to-point architectures, shown in Figure 1.1, can provide scalable solutions and satisfy the tight power and performance requirements posed by on-chip communication requirements. This issue makes significant changes in microprocessor architectures and, consequently, the current design methodology needs to change from computation-based design to communication-based design. The concept of Network-on-Chip (NoC) architecture [1] has been proposed as a promising alternative to exceed such a limitation of communication and overcome such an enormous wiring delay in the complex on-chip communications.

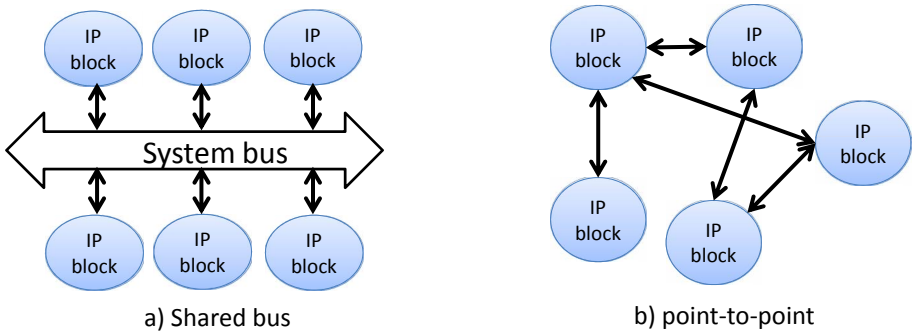


Figure 1.1: Bus-based and point-to-point architectures

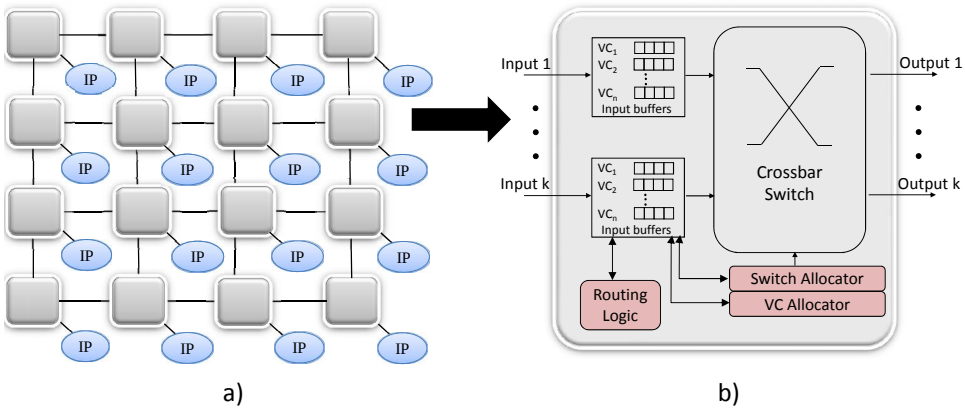


Figure 1.2: NoC architecture

The basic concept of NoC comes from the modern computer network evolution since communications are applied like a network and routers are inserted in between them as depicted in Figure 1.2 a). In fact, an NoC based multicore consists of multiple point-to-point links connected via routers. Messages can be relayed from any source node to any destination node over several links, by making routing decisions at the routers. In this respect, the switch-based interconnection mechanism shorten the required wiring, provides a lot of scalability and freedom from the limitation of complex wiring. The NoC approach can provide large bandwidth with moderate area overhead, compared to the traditional solutions. Besides scalability, the NoC approach offers increased reusability of the design.

Network topology in NoCs determines how switches and nodes are connected. For instance, the topology of network shown in Figure 1.2 a) is a two-dimensional mesh. Figure 1.2 b) depicts the microarchitecture of a typical router. The router in a two-dimensional mesh network has five input and output ports corresponding to the four neighboring directions and the local processing element (PE) port. The major router components include the buffers, routing logic, VC allocator, switch

allocators, and the crossbar switch. Most routers in on-chip network are input-buffered which means that they store packets in buffers only at the input ports. The following stages constitute the functionality of an on-chip network router:

- *Buffer Write (BW)*: A head flit is first decoded and buffered to its input VC on arriving at an input port.
- *Route Computation (RC)*: The routing logic performs RC to determine the output port for the packet. To this end, the head flit indicates the VC that it belongs to, the VC state is updated, and the next output port is computed based on routing algorithms.
- *Virtual-channel Allocation (VA)*: the head flit arbitrates for the available VC on its output port.
- *Switch Allocation (SA)*: the header flit arbitrates for access to its output port.
- *Switch Traversal (ST)*: the flit traverses the crossbar and is transmitted on the output port.
- *Link Traversal (LT)*: The flit is passed to the next node.

These stages are also known as *router pipeline* stages because they commonly implemented and performed based on pipeline techniques to improve overall latency and throughput in the network.

Routing algorithms select a route among possible paths from source to destination and are categorized into deterministic/oblivious and adaptive ones. There has been always a tradeoff between the degree of adaptivity and ease of design in routing algorithms. For example, the deterministic routing algorithms have no adaptivity which means they select a fixed route without considering the state of the network which results in simple design complexity. On the contrary, adaptive routing algorithms use dynamic information about the network and can make better decision in terms of the performance of the network. For instance, channel load information helps these algorithms to balance load in the network and thus improve the performance. As the degree of adaptivity in these algorithms is increased, the flexibility in routing paths and the design complexity are increased.

Switching mechanism in NoCs determines when and how network resources, such as links and buffers, are allocated and de-allocated to messages as they travel through the network. There are different types of switching techniques such as circuit switching, packet switching, and wormhole switching. The switching technique used in the network affects different performance metrics. For example, circuit switching reserves network bandwidth for the entire duration of the delivered data while it ties resources and may cause unnecessary delays. Packet switching needs large-sized buffers as it stores entire packets in a switch. By contrast, wormhole switching requires smaller buffer size while it reduces the ability of interleaving distinct messages over a physical channel which leads to less channel utilization.

Flow control mechanism determines how to handle the situation in which a message traversing the network needs to compete with other messages to acquire network resources. The implementation of the routing, switching, flow control, and router pipeline will exert influence on the efficiency at which buffers and links are used and thus overall network latency and throughput [2].

Regarding minimizing the implementation cost in on-chip networks, it is important to reduce area overhead. As buffers take a significant portion of the silicon area in NoCs [3, 4] buffer size in routers should be carefully minimized. On the other hand, the reduction of the buffer space in routers may cause the poor performance of the network. Moreover, the uniform distribution of buffer spaces is widely used by designers due to its simplicity. However, it may result in using unnecessary buffer space (silicon area) and low performance in the network. The present thesis addresses some of the issues in this subject.

1.2 QoS-aware Communication Management: A Major Research Challenge in NoC

Although the benefits of on-chip networks are considerable, numerous research challenges are presented to reach their full potential. As understood of [5], one of the major research problems in NoC design is QoS-aware communication management led to performance modeling and optimization in the network. To address this challenge, it is important to have a good analysis of the traffic communications, system requirements, and network metrics. This has also a huge impact on design costs, power, and performance. At this point, communication bandwidth and network latency are the key performance metrics, while area, power, and reliability are the key cost metrics.

Communications can be managed as offline by making optimal design decisions such as finding a sufficient configuration of buffers, optimal arbitration policy, optimal network topology, and appropriate traffic shaping through static flow regulation/control or as online decision making by flow control mechanisms and dynamic regulation with online feedback information in the case of run-time communication management.

While sharing resources results in increased overall performance and scalability, it also leads to unpredictable delays per individual flow. This nondeterminism can substantially degrade the overall performance in applications with real-time deadlines. Therefore, a daunting challenge faced by NoC designers is how to efficiently use the shared resources such as links and routers to encounter requirements of various applications and how to analyze deterministic bounds for communication delay and throughput. Providing QoS is considered to be a critical problem for applications executing on embedded multicore systems [6]. Contention in shared resources affects performance and QoS significantly. While this subject has been studied recently in Chip Multi-Processor (CMP) architectures, the same subject

exists in SoC architectures, which is even more severe due to the interference of shared resources between programmable cores and fixed-function IP blocks.

1.3 Contributions

The focus of the present thesis is on the resource constrained communication management, with the aim of minimizing the network cost or maximizing network utilization while preserving the required QoS. The author has studied performance analysis and optimization of NoC communications and proposed techniques to support QoS, for both BE and GS traffic flows. Contributions in this thesis are divided into the following categories:

1. Communication management for BE traffic flows

   Flow control based on different optimization scenarios

   - *Contribution:* A framework to provide QoS with the following objective functions:
     - Maximizing throughput
     - Providing fairness
     - Minimizing total BE traffic delays

2. Communication management for real-time systems with guaranteed services

   a) Flow regulation and performance analysis without Virtual Channel (VC) sharing

      - *Contribution:* Propose flow regulation and define regulation spectrum as a means to control delay and backlog bounds. Also, analytical models to derive worst-case delay and backlog bounds are defined.

   b) Performance analysis of flows with VC sharing in network based on aggregate scheduling.

      - *Contribution:* Propose analytical models for different resource sharing scenarios, classify and analyze flow interference patterns, propose and prove required theorems and finally derive per-flow worst-case delay bounds.

   c) Design optimization based on analytical performance models

      - *Contribution:* Define and solve optimization problems based on analytical models with the aim of minimizing the network delay bounds.

More features and discussions concerning problems and contributions are described in Chapter 3.

1.4 Thesis Organization

The present thesis consists of two main parts: a general introduction and discussion in Part I and a collection of papers in Part II. The remainder of Part I is organized as follows. Chapter 2 reviews the most significant related works and backgrounds. The contributions are elaborated in Chapter 3. Chapter 4 gives the conclusions and highlight directions for future work.

The collection of papers in Part II includes 10 conference proceedings, 2 journal papers and 2 submitted journal papers.

Chapter 2

Background and Related Works

The context of this chapter includes five sections. The first section considers the importance of quality of service and the possible approaches for providing it. Flow control as one of these approaches is described in the nest section. The third section is devoted to the NoC performance evaluation. The next section introduces the basics of network calculus and reviews some related works using network calculus theory. The final section introduces most significant optimization concepts and represents different categories of optimization problems.

2.1 Quality-of-Service (QoS)

QoS is particularly very important for communications with special requirements, such as communications for audio conversations or even for applications with stricter service demands.

The on-chip networks provide scalability and support for parallel transactions. The computational power of these architectures enables the simultaneous execution of several applications, with different time constraints. Therefore, it is expected that various applications such as real-time and multimedia, and computation-intensive algorithms such as video encoding and decoding algorithms, speech recognition, and 3D gaming, will be supported on a NoC environment. In this respect, NoC should be able to provide various levels of support for these applications. It must be also able to guarantee a timely exchange of data packets for a real-time application. On the other hand, as the number of applications executing simultaneously increases, the performance of such applications may be affected due to resources sharing. In this respect, applications can experience large latency fluctuations for packet delivery because of network congestion. Such variability and non-determinacy result in degradation of overall application performance which is not obviously acceptable for applications with real-time deadlines.

To ensure applications requirements are met, mechanisms are necessary for ensuring proper isolation. As the NoC is one of the main shared components in

NoC-based MPSoCs, meeting communication requirements of applications is a crucial aspect of QoS mechanisms in these systems. QoS metrics includes delay, delay variation (jitter), throughput, error rate, and the rate of packet loss etc [7]. In fact, QoS specification can be expressed by performance metrics and can be categorized by worst-case bounds, average values, and percentiles etc.

A significant number of existing studies in this subject have developed mechanisms to provide delay and throughput guarantees. In [8], authors propose Time-division-multiplexing (TDM) circuit-switching to guarantee bandwidth and latency. Nostrum NoC [9] creates virtual circuits and defines containers to provide bandwidth guarantee. In SonicsMX [10], authors insert interval markers for acquiring bandwidth and providing soft guarantees on minimum bandwidth and maximum delay. Authors in [11] and [12] offer efficient throughput guarantees. In [13], authors investigate end-to-end delay and packet loss as QoS metrics to quantify buffering requirements and packet switching techniques in NoC nodes. A QoS-aware routing algorithm proposed in [14] partially adapts with the traffic congestion for meeting different QoS requirements such as average delay and jitter. Authors in [15] integrate the QoS and error control schemes considering latency, jitter, error rate etc. The present thesis particularly investigates throughput and delay as QoS parameters.

Since over half of research studies are devoted to timing aspects [16], there is a need for research into NoCs to provide deterministic bounds for communication delay and throughput. In NoCs, QoS mechanisms concerning timing guarantees are commonly handled by following approaches:

- One possible solution to this problem is to add some redundant links, nodes and buffers to over-dimension the network. The network employs these links when congested.

- Another possible solution is reserving resources like VCs with a mechanism of resource allocation between different traffic flows [17], [18], [19], [20], [9]. For instance, some links can be reserved for real-time applications to guarantee a timely delivery of data packets from source node to destination node.

Both solutions are able to raise the latency problem but increase the cost and power consumption in the network.

- A cost-efficient solution is to provide multiple priority levels to the data traffic, which can be supported within the network such that the urgent traffic can have a higher priority than the regular traffic [21], [22], [23], [24]. To transmit the data packet for a real-time application in time, either some links are reserved for real-time data or priority-based scheduling is implemented. However, without appropriate scheduling algorithms in such systems, a data packet belonging to a lower priority application may be starved. Methods to safeguard global fairness to network hot spots have been proposed in [11].

- Finally, QoS-aware communication management is another solution for providing QoS in NoCs. This provides a QoS framework for managing traffic communications by allocating a certain amount of resources to each flow and/or shaping traffic flows. The framework may be modeled statically by making optimal design decisions such as finding optimal arbitration policy and static flow regulation or dynamically by flow control mechanisms and dynamic flow regulation with online feedback information.

  QoS-aware flow control algorithms have been proposed to avoid the spikes in delay by regulating traffic at the NI and to ensure fairness [25], [26], [27], [28].

2.2 Flow Control

Flow control determines how to handle the situation in which a traffic flow traversing the network needs to compete with other flows to acquire network resources, such as channel bandwidth and buffer capacity. Such control mechanisms try to avoid resource starvation and congestion in the network by regulating traffic flows which compete for shared resources. The majority of flow control presented in the NoC domain relies on switch-to-switch or end-to-end mechanisms.

2.2.1 Switch-to-switch flow control mechanisms

Switch-to-switch flow control mechanisms exchange control signals between the neighboring routers to regulate the traffic flow locally [29], [30], [31], [32] [33], [34]. The switch-to-switch flow control can be categorized into credit based, on-off, ACK-/NAK, and handshaking signal based mechanisms

- *Credit based flow control:* In this mechanism, the count of data transfers is kept by an upstream node, and therefore the available free slots are termed as credits. A credit is sent back when the transmitted data packet is either consumed or further transmitted. Authors in [35] and [36] use credit based flow control in QNoC.

- *On-off based flow control*: Credit based flow control requires upstream signaling for every flit, while on- off based flow control decreases upstream signaling. Off signal is sent when the number of free buffers falls below threshold F_{off} and On signal is sent when the number of free buffers rises above threshold Fon.

- *ACK/NACK protocol*: This technique keeps a copy of a data flit in a buffer. Once an ACK signal is received, the flit is deleted from the buffer and if a NACK signal is asserted then the flit is scheduled for retransmission. Authors in [37], [38], and [39], use this mechanism in XPIPES implementation.

- *Handshaking signal based flow control*: This mechanism sends a VALID signal whenever a sender transmits any flit. The receiver consumes the data flit

and then acknowledges by asserting a VALID signal. In [40], authors use handshaking signals in their SoCIN NoC implementation.

Since switch-to-switch approaches do not need explicit communication of control information between source and destination, they have a small communication overhead. However, they do not regulate the actual packet injection rate directly at the traffic source level. Indeed, these approaches rely on a backpressure mechanism in which the availability of the buffers in the downstream routers is propagated to the traffic sources. Consequently, before the traffic sources get congestion information, the packets generated in the meantime can seriously congest the network.

Several works have been presented to overcome this issue. In [41], a predictive flow control algorithm for on-chip networks is proposed in which each router predicts the buffer occupancy to sense congestion. This scheme controls the packet injection rate and regulates the number of packets in the network. This works tries to reach the simplicity of the switch-to-switch algorithms, while controlling the source nodes similar to the end-to-end algorithms. In [42], link utilization is used as a congestion measure and a prediction-based controller determines the source rates. Dyad [3] controls the congestion by switching from deterministic to adaptive routing when the NoC faces congestion. However, the method cannot guarantee that congestion is resolved since the alternative paths may also be congested.

2.2.2 End-to-end flow control mechanisms

End-to-end flow control mechanisms regulate the packet injection rate at the source nodes in order to conserve the number of packets in the network. Flow control is well studied for data networks [43]- [46]. A wide variety of flow control mechanisms in data network belongs to the class of end-to-end control schemes which is mainly based on the window-based scheme like TCP/IP. In window-based mechanisms, a source node can only send a limited number of packets before the previously sent packets are removed from the network. In this respect, routers and intermediate nodes avoid the network from congestion by dropping packets deterministically (as in DropTail) or randomly (as in RED). Therefore, sent packets are subject to loss and the network must aim to providing an acknowledgement mechanism. One limitation of end-to-end control mechanisms is the large overhead incurred when sending the feedback information [46]. Moreover, the unpredictable delay in the feedback loop can cause unstable behavior as the link capacities increase [47].

Compared to off-chip networks, on-chip networks pose different challenges. The reliability of on-chip wires and more effective link-level flow-control allows NoCs to be lossless. Therefore, there is no need to utilize an acknowledgment mechanism like what exists in off-chip networks and researchers face to a slightly different concept of flow control. The work presented in [48] employs the end-to-end flow control for guaranteed service along with the basic link-level control in on-chip networks. Authors in [49] present a comparison of the overhead of flow control algorithms.

2.3 NoC Performance Evaluation

The NoC designers should be aware of performance requirements and cost constraints to have enough for the choice of design parameters. They must also be able to provide a framework for dynamic and static resource allocations in order to meet the QoS requirements of different applications. Therefore, they need to derive an accurate and fast performance evaluation regarding different configurations exploring the design space. As network performance evaluations are highly dependent on the traffic patterns variation, a first step towards understanding and unraveling network performance related issues is how to model traffic flows in the network. Workloads are usually simulated and there are different ways to do that such as reading traces from files; generating synthetic traffic on the fly; running application programs in a system simulator; etc. Section 2.3.1 categorizes and discusses different workloads.

2.3.1 NoC Workloads

To evaluate an NoC design it is necessary to investigate workload models that can have significant impact on network performance. Several types of traffic patterns are discussed as follows:

- **Execution-driven workload:**

  Traffic patterns are generated by running the intended applications on the platform. Consequently, both the processor cores and the NoC infrastructure are modeled in the traffic pattern. As execution-driven workload emulates the processors in addition to the NoC itself, it is the most accurate and appropriate traffic pattern to use. However, requires a full-system implementation and suffers from long evaluation time.

- **Trace-driven workload:**

  In this kind of workload, only the network model are evaluated and processor core are considered as a "black-box" that only generates packets according to the collected trace. This workload can be an efficient alternative to execution-driven workload under realistic applications.

  The major drawback of these two kinds of workload is that the achievement of a complete coverage of all the expected traffic is very difficult and complex because the number of benchmarks is limited. Moreover, the simulation time is long such that it cannot be used in the optimization loop [50] [51].

- **Synthetic workloads**

  Due to complexity of developing and controlling of trace-driven workloads, synthetic workloads are used frequently in NoCs simulation. Besides of simplicity to design and manipulate, synthetic workloads can help analyze and characterize NoC applications. They can also be used to generate new traffic

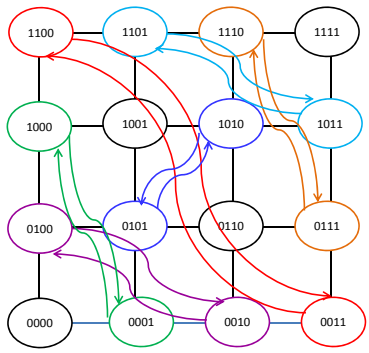


Figure 2.1: Bit-reversal distribution

traces with features that are not covered by existing applications. Among different types of synthetic models, statistical and arrival curve-constrained traffic models are described as follows.

– *Statistical Traffic Models*

  Due to high versatility of statistical traffic models, they can be used to carefully design synthetic workloads employing statistical approaches. There are two major parameters generated by this type of traffic models including packet length distributions and temporal distribution. The temporal distribution refers to the distribution of inter-arrival time of packets; such as *Periodic* process and *Poisson* process. In a Periodic process, the packets inter-arrival times are fixed and known while Poisson process incorporates fluctuations in the inter-arrival times based on the exponential distribution.

  Other parameters, such as spatial distribution, routes, etc., are of less importance. The spatial distribution represents the distribution of the destination of packets in the network. Several common examples of spatial distributions used in NoC are uniform, transpose, bit-reversal, and shuffle traffic patterns [51]. Figure 2.1 shows a bit-reversal distribution in the 8×8 mesh topology as an example.

  NoC performance evaluations are predominantly based on the Poisson traffic characteristics [52], namely, the packet inter-arrival times and the packet service time at each router are exponentially distributed. Although recent researches have demonstrated these assumptions may not hold for some NoC applications [53–55] and Poisson model is not able to model all significant features in this network, it is still one of the most widely used traffic model in NoCs.

– *Arrival Curve-constrained Traffic Models*

  To speed up time-to-market, computation and communication are developed separately and concurrently. Therefore, the communication plat-

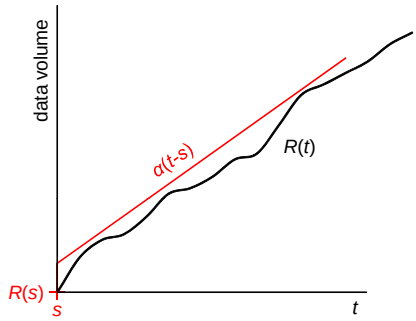


Figure 2.2: Arrival curve

form is developed without sufficient traffic knowledge. In this respect, it is very important to be able to analyze and evaluate network communication performance with various traffic patterns extensively so as to make the right design decisions.

Network Calculus is a generic theory conceived to derive upper bounds on network traversal times. This theory is able to model all traffic patterns with bounds defined by arrival curves. In this respect, designers can capture some dynamic features of the network based on shapes of the traffic flows [56]. The concept of arrival curve is defined as below.

Definition 1. *Arrival Curve [57]*: Given a wide-sense increasing function α defined for $t \geq 0$, we say that a flow R is constrained by α if and only if for all $s \leq t : R(t) - R(s) \leq \alpha(t - s)$.

We say that R has α as an arrival curve, or also that R is α-smooth.

Note that the condition is over a set of overlapping intervals, as Figure 2.2 illustrates.

In this respect, network calculus-based analytical models employ arrival curves constraining traffic workloads to compute upper bounds.

NoC performance models are categorized into analytical-based and simulation-based models.

2.3.2 Simulation-based Models

SoC designs are becoming increasingly complex with time and have tight constraints in terms of performance, cost, energy consumption, dependability, flexibility, security, etc. In order to be sure that design of such a complex SoC device is truly correct, it is logical to early simulate the design beforehand implementation because the implementation of the billion transistors early and then discovering out a design problem would be very disastrous. A simulation tool should be able to explore the architectural design space quickly, evaluate a design of the network architecture with a variety of regular traffic models and application-oriented traffic,

and estimate design quality in terms of performance, cost, power and reliability etc. Overall, simulators are applicable for following purposes:

- Evaluation of various hardware designs without implementing costly hardware systems.
- Making opportunities for evaluating non-existing components or systems.
- Estimating design metrics including performance parameters. Simulators are able to generate a large set of performance data by a single execution.
- Debugging before implementing the system. Once an error is detected in a real system, it typically needs re-booting and re-running the code or the design to re-produce the problems while some simulators are able to run code backward by a controlled environment to debug the design.

Currently a few public simulation tool exists to aid NoC designers to make the decision. Table 2.1 briefly introduces some of existing open-source (code available) NoC simulators.

The present thesis particularly employs Booksim to validate the proposed analytical models.

2.3.3 Analytical Models

Simulation tools are commonly used to explore the design space for the estimation of performance metrics. Although simulators are flexible and provide highly accurate estimations, due to complexity of modern SoCs, it is a very time consuming process that can hardly be used during the iterative exploration phase of the design. The non-linear behaviour of system performance makes the process even harder especially for estimation of worst-case performance metrics. Moreover, simulators are not scalable with the network size since they increase the computational complexity of performance metrics estimation in larger systems.

For these reasons, analytical models are proposed as an alternative approach for efficient and reasonably accurate performance evaluations. Analytical models promise a fast evaluation of performance metrics that allows for a larger design space to be explored. They can provide a clear relationship between inputs and outputs and other design parameters in the network. They can make it possible to understand the effects of these parameters on the performance of a system. Analytical models provide a perfectly general insight of a system, but some small details may be not well represented because they often use simplifications which their impact should be considered carefully. When the analytical techniques are too abstract and distant from reality or too complex to find a solution, simulation results are used to evaluate performance in the system.

Regarding application requirements, analytical techniques for both the average [69] and the worst-case [70, 71] performance metrics are needed to be employed.

Table 2.1: The list of some open-source NoC simulators

Simulator	Framework	Characteristics
Booksim [58], Stanford University	C++	Topo: 2D mesh, torus, trees, etc.; Traffic: uniform, transpose, etc.;
NoCsim [59], Texas A&M University	SystemC	Topo: k-ary n-cube & arbitrary topological extensions; Routing: source-based, dynamic & multicast; Flow control: dynamic & static; Switching mechanism: packet switched
Nostrum NoC Simulation Environment (NNSE) [60], KTH Royal Institute of Technology	SystemC	Topo: 2D mesh, torus; Flow control: wormhole routing and reflection routing; No parallelism;
Noxim [61], University of Catania	SystemC	Topo: 2D mesh; Traffic: random, transpose, etc.;
Worm_Sim [62], Carnegie Mellon University	C++	Topo: different topologies such as 2D mesh & torus; Routing: different routing algorithms; Traffic: built-in; Power: Ebit, Orion 1; No multiple VCs support;
gpNoCsim [63], Bangladesh University of Engineering and Technology (BUET)	Java	Topo: All; No parallelism;
Xmulator [64], IPM School of Computer Science & Sharif University of Technology	C#	
Nirgam [65], University of Southampton	SystemC	Topo: 2D mesh, torus; Routing: XY, adaptive OE, source routing; No parallelism; Switching mechanism: wormhole;
DARSIM [66]	C++	Topo: All; Support parallelism;
SICOSYS [67], University of Cantabria, Spain	C++	Topo: limited; No parallelism;
TOPAZ [68], University of Cantabria, Spain	C++	Derived from SICOSYS; 50K lines of code; Support parallelism;

2.3.3.1 Average-case performance models (Best Effort Communications)

For applications with Best Effort (BE) communications, designers aim in providing the highest performance at a given cost, which is maximizing the average-case performance metrics under the design constraints. These applications may have soft real-time requirements, non-time-critical requirements, which must normally be satisfied, but can sporadically be disregarded at cost of a small decrease in quality of the output, like audible or visual artifacts in an audio or video stream.

To design a more efficient system, the average execution time of the application is concerned in performance analysis. A variety of mathematical approaches are used for modeling the average-case performance in NoC such as the *queuing-theory-based models* [69, 72, 73]. Queuing approaches often use probability distributions like Poisson to model traffic in the network while Poisson distribution used in queuing model is not appropriate for characterizing traffic patterns in NoC applications because it is not able to model all significant features in this network. Queuing theory generally evaluate average quantities of metrics in an equilibrium state and characterizing their transient behavior is a very difficult problem for this approach.

2.3.3.2 Worst-case performance models (Guaranteed Service Communications)

Many NoC applications have real-time constraints on traffic flows, which means they have strict requirements on communication latency and bandwidth and need guaranteed QoS to delivery packets. In real-time systems with Guaranteed Service (GS) communications, the design goal is to provide a minimum level of performance at the lowest possible cost. In such systems, it is very important to evaluate worst-case delay bounds and guarantee that tasks will always be finished before the predetermined deadline. Different analytical approaches proposed for deriving the delay bounds in NoCs include *dataflow analysis*, *schedulability analysis*, *Real-Time Bound (RTB) formulation*, and *network calculus*.

Dataflow analysis is a deterministic approach based on graph theory in which the pattern of communication among cores and switches are deterministic and predefined [74]. To capture dynamic behavior, it must be used with restricted models such as DDF. In fact, the expressiveness is typically traded off against analyzability and implementation efficiency in this analytical approach.

Schedulability analysis is a mathematical formalism for analyzing the timing properties in real-time systems. In this respect, a set of tasks, their worst-case execution time, and a scheduling policy are given as inputs and the model determines whether these tasks can be scheduled such that deadline misses never occur [69]. Compared to the other mathematical formalisms, this approach uses simpler event models and consequently the performance model is easily extracted with less accuracy.

Real-Time Bound (RTB) formulation [70] is inspired by schedulability analysis and derives delay bounds when all the intermediate buffers along the path of the target flow are full, and the target flow loses arbitration at all routers against the contention flows [75, 76].

Network calculus is a mathematical framework for deriving worst-case bounds on maximum latency, backlog, and minimum throughput in network-based systems. It is a promising method for analyzing performance guarantees and considering quality of service in the network. This theory can characterize all traffic patterns and some dynamic features of the network based on defined arrival curves and shapes of the traffic flows [77]. It is also able to abstract many scheduling algorithms and arrival

classes as multiplexed arrival flows at a single queue by service curves. The service curves through a network can be convolved as a single service curve. Hence a multi-node network analysis can be simplified to a single-node analysis. Regarding these two features, network calculus can analyze many scheduling algorithms and arrival classes over a multi-node network in a uniform framework while most of other analytical methods separately model different combination of them [78]. This thesis applies network calculus to present formal approaches for QoS analysis in network-based SoC communication.

In [79], authors have surveyed four popular mathematical formalisms -*dataflow analysis, schedulability analysis, queueing theory,* and *network calculus*- along with their applications in NoCs. They have also reviewed strengths and weaknesses of each technique and its suitability for a specific purpose.

2.4 Network Calculus Theory

Since the thesis applies *network calculus* theory to propose worst-case analytical models, this section recapitulates the concepts from network calculus [57] which are relevant for this thesis and looks into some related works based on this theory.

2.4.1 Basic Concepts of Network Calculus

Network calculus [57] is a theory dealing with queueing type problems encountered in computer networks, with particular focus on quality of service guarantee analysis. It gives a theoretical framework for worst-case performance analysis in deterministic queueing systems and is able to express and analyze constraints imposed by the network components such as link capacity, traffic shapers (e.g. leaky buckets), congestion control, and background traffic.

Assuming a system consists of an input, a transfer function and an output, the input is an abstraction of the traffic flow and the transfer function is an abstraction of the scheduling. The input and transfer function are referred to as arrival curve and service curve, respectively. Network calculus can also be used to express departure function as well as arrival and service curves.

A key difference of network calculus to conventional system theory is using the *min-plus algebra* in which *addition* and *multiplication* are replaced by *minimum* and *addition*, respectively. The reason to switch to min-plus algebra is that it is able to preserve linearity by transforming complex non-linear queueing systems into analytically tractable linear systems.

In min-plus algebra, $\wedge$ denotes the infimum or, when it exists, the minimum, $f \wedge g = min(f, g)$; $\vee$ denotes the supremum or, when it exists, the maximum, $f \vee g = max(f, g)$; $+$ is the "multiplication" operation. It can be verified that min-plus algebra has similar properties as the conventional algebra such as the closure property, associativity, commutativity, and distributivity.

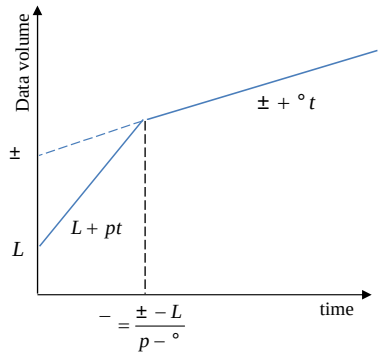


Figure 2.3: TSPEC arrival curve

As in conventional system theory, a key operation in network calculus is the *min-plus convolution*. The min-plus convolution, denoted by $\otimes$, and the *min-plus deconvolution* denoted by $\oslash$, are respectively defined as:

$$(f \otimes g)(t) = inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}$$
$$(f \oslash g)(t) = sup_{s \geq 0} \{f(t+s) - g(s)\}$$

where $f, g \in F$ and F is the set of wide-sense increasing functions.

Arrival curve and service curve are the most significant concepts in network calculus. As defined in Section 2.3.1, an arrival curve denotes the largest amount of traffic allowed to be sent in a given time interval.

Since the arrival curve defines a bound on the arrival traffic, it can be considered as an abstraction of the traffic regulation algorithm. *Leaky Bucket (Token Bucket)* [80] is the most common regulation algorithm which its arrival curve is defined as $\alpha(t) = \sigma + \rho t$ for $t > 0$; where σ and ρ are the burstiness and average rate, respectively. Thus, the long-term rate is ρ and at most σ data units can be sent at once. This arrival curve only considers the average behavior of traffic and no peak behavior is modeled.

To model both the average and peak behavior of flows, the present thesis employs *Traffic SPECification (TSPEC)*. With TSPEC, a traffic flow is characterized as $\alpha(t) = min(L + pt, \sigma + \rho t)$; where L is the maximum transfer size, p the peak rate ($p \geq \rho$), σ the burstiness ($\sigma \geq L$), and ρ the average rate. As shown in Fig. 2.3, $\alpha(t) = L + pt$ if $t \leq \theta$; $\alpha(t) = \sigma + \rho t$, otherwise. This arrival curve is an abstract of a *Dual Leaky Bucket*.

A service curve defines a bound on the service provided by network elements such as links, routers, and regulators in order to present an abstract model for their behavior.

Definition 2. *Service Curve [57]*: Consider a system S and a flow through S with input and output functions R and R^*, respectively. We say that S offers to the

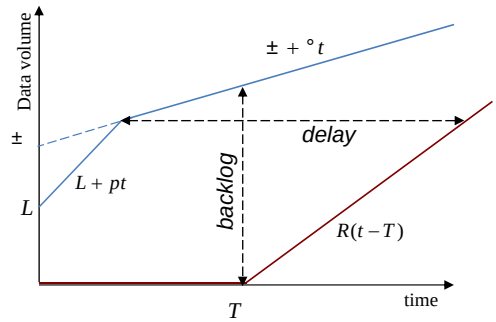


Figure 2.4: Bounds for TSPEC flow served by a latency-rate server

flow a service curve β if and only if $\beta \in F$ and $R^* \geq R \otimes \beta$.

A prominent service model is the rate-latency function $\beta_{R,T} = R[t-T]^+$, which means that $\beta_{R,T} = R(t-T)$ if $t > T; \beta_{R,T} = 0$, otherwise. In this definition, R is the minimum service rate and T the maximum processing delay.

We then introduce the *backlog* and *delay* bounds which are two basic bounds of network calculus.

Theorem 1. *(Backlog Bound [57]). Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β, the backlog $R(t) - R^*(t)$ for all t satisfies: $R(t) - R^*(t) \leq sup_{s \geq 0} \alpha(s) - \beta(s)$.*

Proof. The proof can be found in [57]. □

The theorem says that the backlog is bounded by the vertical deviation between the arrival and service curves.

Theorem 2. *(Delay Bound [57]). Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β, the delay $d(t)$ for all t satisfies: $d(t) \leq h(\alpha, \beta)$.*
where $h(\alpha, \beta)$ is the supremum of all values of $\delta(s)$ and $\delta(s) = inf\{\tau \geq 0 : \alpha(s) \leq \beta(s + \tau)\}$.

Proof. The proof can be found in [57]. □

The theorem says that the delay is bounded by $h(\alpha, \beta)$ which is the horizontal deviation between the arrival and service curves. Figure 2.4 shows bounds for TSPEC flow served by a latency-rate server.

2.4.2 Network-calculus-based Models for Deriving Upper Delay Bounds

In [81], authors evaluate performance and cost metrics, such as latency, energy consumption, and area requirements by using a proposed analytical approach based on network calculus. They apply their proposed model for different topologies including 2D mesh, spidergon, and WK-recursive and show that WK-recursive outperforms two other topologies in all considered metrics. Although this model considers trade-offs between different metrics, it is very simple and is not accurate since it does not analyze virtual channel effects and cannot model all interferences between flows sharing a resource in the network.

There are different works which evaluate performance metrics in networks employing aggregate scheduling and are able to model virtual channel impacts and analyze more accurate models for different resource sharing scenarios. Such analytical models are particularly challenging because of their complexity as there has been always a scalable tradeoff between accuracy and ease of analysis in NC-based models. Aggregate scheduling arises for various network infrastructures such as internet and NoC. The Differentiated Services (DiffServ) [82] is an example of an aggregate scheduling-based architecture in the Internet. The authors in [83] present a survey on this subject. The analytical method proposed in [84] obtains a closed-form delay bound for a generic network configuration under the fluid model assumption. To look into the influence of packetization, the method is extended in [85]. Although these models can derive sufficient bounds in a generic network configuration, they only work for small utilization factors.

Authors in [86] compare network calculus and the trajectory approaches on a real avionics AFDX configuration and shows that the trajectory approach computes tighter upper bounds compared to network calculus. However, delay bounds derived from network calculus are calculated by the summation of per-node delay bounds, expectedly resulting in a loose total delay bound.

There are different works which compute delay bound through network calculus in feed-forward networks under arbitrary multiplexing [87–89]. Authors in [89] aim to derive the worst-case end-to-end delay bound for a target flow in any feed-forward network under blind multiplexing, with concave arrival curves and convex service curves. They present a first algorithm for this problem. However, since it is a difficult (NP-hard) problem, the paper shows some cases, such as tandem networks with cross-traffic interfering along intervals of servers, in which the complexity becomes polynomial. [90] improves the proposed method in [89] to consider networks with a fixed priority service policy. Authors in this work try to take into account the pay multiplexing only once (PMOO) phenomenon. These works consider networks with arbitrary or blind multiplexing in which there is no assumption about service policy while an explicit assumption on multiplexing scheme, like FIFO, results in tighter bounds.

A related stream of works is concerned to the proposed methodology in [91–93]. Authors in these works calculate delay bounds in tandem networks of rate-

latency nodes traversed by leaky bucket shaped flows in the FIFO order. They also implement algorithms employed in their methodology and present them as a tool called DEBORAH. These works deal with networks only in tandem or sink trees and are not able to compute end-to-end delay in a generic topology. All aforementioned works compute delay bound considering only average behavior of flows and not peak behavior, which arrives at less accurate bounds.

In [94], it is assumed that each server is shared by two flows and the authors try to model shaping for an end-to-end delay under such a system. They shape an applicative token bucket $\gamma_{r,b}$ by the bit-rate of the link λ_R, which lead to a two-slopes affine arrival curve. This arrival curve is similar to one models double leaky bucket and considers traffic peak behavior. However, the paper investigates a simple type of nested contention in a simple topology consists of a sequence of rate-latency servers shared by two flows with a FIFO policy. Moreover, as stated in their paper, the proposed model is incomplete since they model only the shaping on the considering flow, not on the interfering ones when computing the worst-case traversal time of a flow. That is why they entitled their own paper as "half-modeling of shaping".

All reviewed works in the subject of aggregate scheduling propose a methodology for deriving delay bounds in off-chip networks of different nature but not on-chip networks. The analytical models are very close to the reality of the system in on-chip networks. As an example, a router in on-chip networks can be modeled in pure hardware which means the micro-architecture is feasible for analysis. Therefore, network calculus can analyze more accurate models in on-chip networks. Authors in [95] propose a network-calculus based approach for modeling flow control and resource sharing and analyzing per-flow communication delay bounds in wormhole networks. They then extend their analytical models under strict priority queueing in [96] and compare it with weighted round robin scheduling in terms of the service behavior. Like most of reviewed works, [95] and [96] do not deal with peak behavior of flows, which results in less accurate bounds. Besides analysis of deterministic performance bounds, authors in [97] analyze "soft" performance bounds in NoCs using stochastic network calculus.

2.5 Optimization Problems

Optimization problems are common in many disciplines and various domains. From the communication management perspective, there exists a huge search space to explore at the network, resulting from the high number of nodes in current and future systems. Thus, designers need to investigate topology, switching, routing and flow control schemes. They should be also able to support QoS requirements by optimizing resource allocation and flow characterizations. Moreover, they need to examine the impact of flow control schemes on performance metrics. Each of the design parameters also has a number of options to consider. Thus, to design an efficient on-chip network, besides performance analysis, developing optimization

problems and making appropriate decisions are of significant importance.

Design decisions are grouped into two categories: architecture-level decisions such as topology, switching, and routing algorithm; application-level decisions such as task-to-node mapping, task scheduling, traffic reshaping. Optimization problems allow designers to investigate the impact of design parameters and performance-cost tradeoffs among these parameters. Obviously, more accurate tradeoffs can be made based on more complex decision models.

Commonly, inputs for optimization problems in this subject include both the architecture specifications A_s such as the bandwidth of channels, buffer space, routing policy, and topology, and application parameters A_p such as communications bandwidth, latency requirements, and traffic specifications. The optimization goals and constraints reflect different metrics belonging to performance and cost parameters. Performance metrics include average/maximum packet latency, bisection bandwidth, and network throughput; and cost metrics include average/peak energy/power consumption, network area overhead, total area, average/peak temperature.

These metrics can be employed as objective functions or constraints defined as a function of the architecture and application parameters as $O(A_s, A_p)$ and $C(A_s, A_p)$, respectively. The general problem is defined as below:

General Problem Description:
Given Architecture specifications A_s and application parameters A_P;
Find A set of decision variables;
Such that the objective function $O(A_s, A_p)$ is optimized,
subject to the constraints specified by $C(A_s, A_p)$.

In this formulation, decision variables can include finding an efficient application mapping to processing cores, statistical traffic parameters (e.g. mean, peak, and variance), a routing algorithm, a resource allocation strategy (e.g., size of buffers, bandwidth of channels, etc.), packet injection rates in the network and buffer size for each channel at each router. $O(A_s, A_p)$ and $C(A_s, A_p)$ can be subsets of the performance and cost metrics. For instance, decision variables can be finding packet injection rates in each source node and the objective can be minimizing the communication latency, such that the bandwidth constraints for each link are satisfied, as defined later in Section 3.1. Depending on the cost, constraints, and flexibility allowed in the design, the optimization problem may have different forms and solution complexities.

There are so many research studies in various subjects considering optimization problems to obtain different goals like minimizing power consumption/ packet latency or maximizing throughput under the corresponding constraints. For example, the authors in [98] present a mapping algorithm to minimize the communication energy subject to bandwidth and latency constraints. A multi-objective mapping algorithm for mesh based NoC architectures is presented in [99]. In [100], a genetic algorithm is proposed to produce a thermally balanced design while minimizing

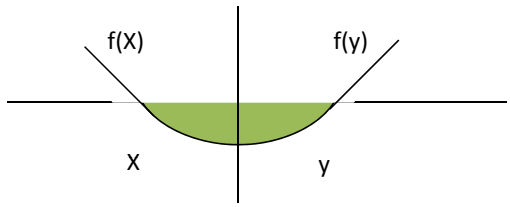


Figure 2.5: An example of convex function

the communication cost via placement. The proposed algorithm in [101] minimizes the overall energy consumption of the system, while guaranteeing the hard deadlines imposed on tasks. There are some works which minimize the average distance traveled by packets in the network, with a constraint on the maximum distance between any pair of nodes [102–104]. Authors in [105–107] focus on designing a router to minimize the latency through it while meeting different constraints such as bandwidth requirements.

As there may exist different solution methods for a specific optimization problem, it is highly important to find appropriate solution approaches. Depending on the types of optimization problems, the solution methods or algorithms that can be used for optimization may find one global optimal solution or near-optimal solutions. Mathematical relationships between the objective and constraints and the decision variables determine the difficulty of an optimization problem that is going to be solved. A key issue for solving optimization problems is whether the problem functions are convex or non-convex. To briefly describe what convexity is, it needs to introduce basic concepts as follows.

Definition 3. *Convex Function* [108]: Algebraically, f is *convex* if, for any x and y, and any t between 0 and 1, $f(tx+(1-t)y) \leq tf(x)+(1-t)f(y)$. Geometrically, a function is *convex* if a line segment drawn from any point $(x, f(x))$ to another point $(y, f(y))$, which is called the *chord* from x to y, lies *on or above* the graph of f, as in Figure 2.5.

Definition 4. *Concave Function* [108]: A function is *concave* if $-f$ is convex, namely, if the chord from x to y lies *on or below* the graph of f.

It is obvious that linear functions are both convex and concave.

Definition 5. *Non-convex Function* [108]: A *non-convex* function is neither convex nor concave.

A common example is the sine function as depicted in Figure 2.6:

It is very important to note that the bounds on the variables may restrict the domain of the objective and constraints to a specific region. For instance, the sine function is convex from $-\pi$ to 0, and concave from 0 to $+\pi$. If the domain of the

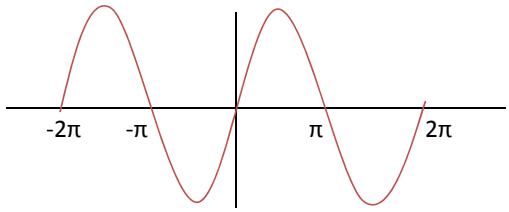


Figure 2.6: An example of nonconvex function

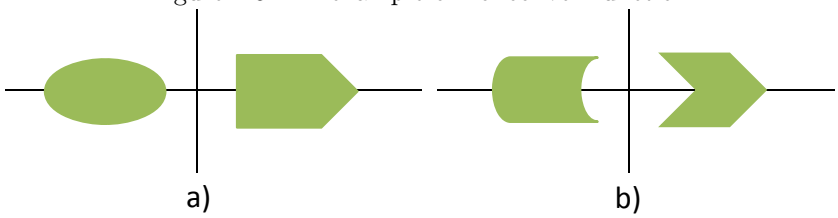


Figure 2.7: The feasible region in a a) convex and b) nonconvex optimization problem

objective and constraints is restricted to a region where the functions are convex, then the overall problem is convex.

Definition 6. *Feasible region* [108]: In mathematical optimization, a feasible region of an optimization problem is the intersection of constraint functions, namely, the set of all possible points that satisfy the constraints.

Definition 7. *Convex optimization problem* [108]: A convex optimization problem is a problem in which all constraints are convex functions and the objective is a convex function if minimizing, or a concave function if maximizing. Linear programming problems are an example of convex problems.

The feasible region in a convex optimization problem is a convex region, as shown in Figure 2.7a).

When both objective and feasible region are convex, there is either only one global optimal solution or no feasible solution to the problem. Convex problems can be solved efficiently up to very large size. Several methods such as Interior Point methods can solve convex problems.

Definition 8. *Non-convex optimization problem* [108]: A non-convex optimization problem is a problem in which the objective or any of the constraints are non-convex, as shown in Figure 2.7b).

Non-convex optimization problems may have multiple feasible regions and multiple locally optimal points within each region. Solution methods for such problems often find near-optimal solutions since it can take time exponential in the number of

Figure 2.8: The ice cream cone

variables and constraints to determine a global optimal solution across all feasible regions.

Optimization problems can be grouped into five categories as follows, arranged in order of increasing difficulty for the solution methods.

- **Linear and Quadratic Programming Problems [108]**

  In a linear programming (LP) problem the objective and all constraints are *linear functions* of the variables. Since all linear functions are convex, LP problems are also convex.

  A quadratic programming (QP) problem is one in which the objective is *quadratic function* (may be convex or non-convex) and all of the constraints are *linear functions*. It is notable that the convexity of the objective function makes the QP problem easier to solve. QP problems, like LP problems, have only one feasible region with "flat faces" on its surface due to the linear constraints, but the optimal solution may be found anywhere within the region or on its surface.

- **Quadratic Constraints and Conic Optimization Problems [108]**

  Conic optimization problems are a class of convex nonlinear optimization problems which can be written as an LP plus one or more cone constraints. A *cone constraint* specifies that the vector formed by a set of decision variables is constrained to lie within a *closed convex pointed cone*. A simple type of closed convex pointed cone is the *second order cone (SOC)* or "ice cream cone" which looks like Figure 2.8.

  A convex quadratic constraint can be converted to an SOC constraint by several steps of linear algebra. Consequently, convex quadratic programming (QP) and quadratically constrained programming (QCP) problems can be formulated as conic optimization problems.

- **Mixed-Integer and Constraint Programming Problems [109] [110]**

  In a mixed-integer programming (MIP) problem some of the decision variables are constrained to be *integer values* at the optimal solution. Since integer variables make an optimization problem non-convex, it becomes more difficult to solve it such that solution time may rise exponentially.

Constraint programming defines "higher-level" constraints that apply to integer variables. The *alldifferent constraint* is one of the most common higher-level constraints in which, for a set of n decision variables, non-repeating orderings of integers from 1 to n are considered. The *traveling salesman problem* is an example of a constraint programming problem. Constraint programming problems not only, like mixed-integer programming problems, are non-convex but also have the extra requirements such as "alldifferent" which make them even harder to solve.

- **Smooth Nonlinear Optimization Problems [111]**

  In a smooth nonlinear programming (NLP), the objective or at least one of the constraints is a *smooth nonlinear function* of the decision variables. A nonlinear function is smooth where its gradients that are its derivatives with respect to each decision variable are continuous.

  Nonlinear functions may involve variables raised to a power or multiplied or divided by other variables, or may use transcendental functions such as log, exp, sine and cosine.

  When the objective and all constraints in an NLP problem are convex functions, the problem can be solved efficiently by interior point methods to find global optimality. In the other hand, if the objective or any constraints are non-convex, the solution methods can find near-optimal solutions as the problem may have multiple feasible regions and multiple locally optimal points.

- **Non-smooth Optimization Problems [111]**

  Non-smooth optimization problems (NSP) are the most difficult type of optimization problem to solve. These problems are non-convex and have multiple feasible regions and multiple locally optimal points. On the other hand, having one possible solution in such problems gives very little information about finding a better solution because some of the functions are non-smooth and gradient information cannot be used to determine the increasing or decreasing direction of the function.

Table 2.2 summarizes different categories of optimization problems. This thesis formulates and solves optimization problems in these categories as stated in the last column of the table.

Table 2.2: Categories on optimization problems

Problem Type	Convex or non-convex?	Notable solution methods	How optimal is the solution?	Paper
Linear and Convex Quadratic Programming Problems	LP: convex; QP: may be convex or non-convex	Simplex method, Interior Point method, Newton-Barrier method, Subgradient method, Projected Gradient method	Global optimal solution	Paper 2,4
Quadratic Constraints and Conic Optimization Problems	Convex	Specialized Interior Point methods, Projected Gradient method, Newton's method	Global optimal solution	Paper 1,3,11
Mixed-Integer and Constraint Programming Problems	Non-convex	Branch and Bound, Genetic and Evolutionary algorithms	Depends on the problem size and solution method, may have global optimal or local optimal solutions	Paper 15,16
Smooth Nonlinear Optimization Problems	May be convex or non-convex	No single method is best for all problems. The most widely used methods, are Interior Point methods and active-set methods including the Generalized Reduced Gradient (GRG) and Sequential Quadratic Programming (SQP) methods	Depends on the convexity of the problem, may have global optimal or local optimal solutions.	Paper 8,12
Non-smooth Optimization Problems	Non-convex	Genetic or Evolutionary Algorithms	Good solutions	Paper 14

Chapter 3

Contributions

The Contributions in the present thesis are summarized in this chapter which are in the subject of communication management with respect to QoS and resource constraints for both BE and GS traffic flows in on-chip networks.

Communication management as a critical means of providing QoS in traditional data networks is a widely studied issue. However, it is still a challenge in on-chip networks. It is natural to think of NoC's nodes as competing for available resources. In this respect, managing communications to satisfy such limited resources to be in accordance to a specified notion of fairness will be of great concern.

The papers and contributions in this thesis are divided into two main categories as belows:

- Communication management for BE traffic flows (Papers 1-7 and 11)
- Communication management for real-time systems with guaranteed service (Papers 8-10 and 12-14)

The present thesis analyzes BE traffic flows based on average-case performance metrics while GS connections are modeled based on the worst-case performance analysis. The author of the thesis is the main contributor to some papers in the first category and the main contributor to all papers of the second one. Here is a short list of publications and submissions.

Accepted:

1. M. S. Talebi, **F. Jafari**, and A. Khonsari, *A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization*, *MASCOTS 2007*.
2. M. S. Talebi, **F. Jafari**, A. Khonsari, and M. H. Yaghmaee, *A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization*, *ICCSA 2007*.
3. M. S. Talebi, **F. Jafari**, A. Khonsari, and M. H. Yaghmaee, *Proportionally-Fair Best Effort Flow Control in Network-on-Chip Architectures*, *PMEO 2008*.

4. **F. Jafari**, M. S. Talebi, A. Khonsari, and M. H. Yaghmaee,A Novel Congestion Control Scheme in Network-on-Chip Based on Best Effort Delay-Sum Optimization, *ISPAN 2008*.
5. **F. Jafari**, M. H. Yaghmaee, M. S. Talebi, and A. Khonsari, *Max-Min-Fair Best Effort Flow Control in Network-on-Chip Architectures*, *ICCS 2008*.
6. **F. Jafari** and M. H. Yaghmaee, *A Novel Flow Control Scheme for Best Effort Traffics in Network-on-Chip Based on Weighted Max-Min-Fairness*, *IST 2008*.
7. **F. Jafari**, M. S. Talebi, M. H. Yaghmaee, and A. Khonsari, *Throughput-fairness tradeoff in Best Effort flow control for on-chip architectures*, *PMEO 2009*.
8. **F. Jafari**, Z. Lu, A. Jantsch, and M. H. Yaghmaee, *Optimal Regulation of Traffic Flows in Network-on-Chip*, *DATE 2010*.
9. **F. Jafari**, A. Jantsch, and Z. Lu, *Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling*, *ICCD 2011*.
10. **F. Jafari**, A. Jantsch, and Z. Lu, *Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling*, *DATE 2012*.
11. M. S. Talebi, **F. Jafari**, A. Khonsari, and M. H. Yaghmaee, *Proportionally Fair Flow Control Mechanism for Best Effort Traffic in Network-on-Chip Architectures*, *International Journal of Parallel, Emergent, and Distributed Systems (IJPEDS), 2010*.
12. **F. Jafari**, Z. Lu, and A. Jantsch, *Buffer Optimization in network-on-Chip through Flow Regulation*, *IEEE Transactions on CAD, 2010*.

Submitted:

13. **F. Jafari**, Z. Lu, and A. Jantsch, *Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels*, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*.
14. **F. Jafari**, A. Jantsch, and Z. Lu, *Weighted Round Robin Configuration for Worst-Case Delay Optimization in Network-on-Chip*, *IEEE Transactions on CAD*.

3.1 Communication management for BE traffic flows

The first part focuses on the flow control for the NoCs with best-effort service as the solution to optimization problems. In on-chip networks, flow control provides a smooth traffic flow by avoiding packet drop and buffer overflow. The flow control can also restrict the packet injection to the network to regulate the packet population in the network [28]. This is precisely the main objective of the first part of the thesis. The present thesis quantitatively measure QoS by consideration of throughput and delay. The general QoS and congestion control problem defined in this thesis formulated as below:

Problem Definition
Given a NoC architecture and an application graph
Find packet injection rates in the network;

Such that network utility is maximized or the network cost is minimized and QoS/resource constraints are satisfied. For example, the aggregate BE source rates passing thorough each link l cannot exceed link capacity c_l. This threshold can be further specified as a function of different service classes like $c_{(l,GS)}$ and $c_{(l,BE)}$, where $c_{(l,BE)}$ represents the portion of the link capacity which has not been allocated to GS sources.

The emphasis of this part of thesis is on understanding and structuring different flow control schemes and considering their specific effects on the network via definition, description, and proposed solutions of various optimization scenarios. The present thesis proposes iterative algorithms as the solution to the optimization problems which have the benefit of low complexity and fast convergence. In order to have a better insight about the behavior of proposed algorithms, the relative error with respect to optimal source rates, which is averaged over all active sources, is calculated. Optimal values are obtained using CVX which is MATLAB toolbox for solving disciplined convex optimization problems. A synthetic case study in Paper 11 exhibits less than 10% average relative error just after running about 13 iteration steps of the proposed iterative flow control algorithm and less than 5% after 20 steps, which confirms the fast convergence of the algorithm.

3.1.1 Utility-Maximization Problem [Paper 1]

In the case of maximizing a network utility, the abovementioned general problem is called a utility-maximization problem as referred in the economics literature. There are many options for utility functions with various features and specific behavior. However, in Paper 1, the general utility function is considered with no restriction on a specific form. The paper transforms the constrained optimization problem into an unconstrained one according to the *Duality Theory*, to reduce the computational complexity. Then, the dual of the problem is solved using simple iterative algorithms. In what follows, the effect of other utility functions on the BE rates and fairness provision is investigated.

- **Identity Function [Paper 2]**:

  The simplest form of the utility function is the *Identity Function* in which the utility function is equal to decision variables. Therefore, the objective function of the optimization problem turns into a sum-rate maximization problem. Since the constraints of this problem are coupled across the network, it has to be solved using centralized methods like interior point methods. As such methods may pose a great overhead on the system; the subgradient method for constrained optimization problems is used to present an iterative algorithm with simpler operations. The convergence analysis of the algorithm reveals that *square-summable but not summable* stepsizes can lead to lower relative error compared to constant stepsizes. The experimental results confirm that

the proposed algorithm converges very fast and the computational overhead of the congestion control algorithm is small.

- **Proportional Fairness [Papers 3 and 11]**:

  One of the famous forms of utility functions is *weighted logarithmic* which satisfies proportional fairness in which resources are shared in proportion to the resource usage of each source. The weight assigned to each source indicates the priority of that source in resource sharing. The problem is indirectly solved through its dual using Newton's method which is led to a flow control algorithm obtaining optimal BE source rates. The performance of the proposed algorithm is evaluated in several aspects:

  - Investigating the convergence behavior of the algorithm indicates the significant role of a stepsize on the convergence speed.
  - The convergence analysis of the algorithm in a dynamic scenario shows that the proposed algorithm needs just as few as 20 iteration steps to move towards the new optimal source rates in a synthetic case study.
  - Regarding the effect of weights, experimental results confirm that using larger weight factors lead to larger rates for corresponding sources while reduce the rate of some other nodes passing through the same channel. Moreover, such an asymmetric case adversely influences the speed of convergence.

- **Max-Min Fairness [Papers 5 and 6]**:

  In networks with Max-Min fairness, resources are mainly shared in favor of weak users. The contribution here is to present a flow control algorithm which satisfies Max-Min fairness criterion. To solve the proposed optimization problem with max-min objective function, it should be converted to a form of disciplined optimization problems and solved by a simple and famous algorithm, known as "*progressive filling*". The effect of max-min fairness on BE rate allocation is considered by comparing several parameters including least source rate, sum of source rates, variance of source rates with respect to mean value, Jain's fairness index, and min-max ratio in both sum-rate maximization and max-min fair flow control schemes.

  Moreover, the author of the thesis formulates weighted max-min problem through the analysis of mathematical model and simulation in Paper 6 for the NoC architecture. To have better insight about the impact of weights, the experimental results with various weights are obtained and the rate region for the weighting scheme is introduced and analyzed.

- **Throughput-Fairness Tradeoff [Paper 7]**:

  Here, two proposed flow control schemes rate-sum maximization and max-min fairness are compared and analyzed in terms of tradeoff between conflicting metrics. With a slight abuse in the definition of throughput in lossless

scenarios, sum-rate maximization scheme is construed as one with the aim of maximizing throughput in the network and max-min scheme guarantees max-min fairness among source rates.

Since there is no rate allocation that can satisfy optimal allocation in terms of both of fairness and throughput, a mechanism for providing a tradeoff between throughput and fairness metrics is of significant importance. To this end, the author of the thesis takes into account weight factors in the underlying optimization problems to define tradeoff between conflicting metrics. The weight assigned to each source determines its priority of rate allocation than other sources. Paper 7 compares underlying flow control schemes and analyze the influence of weight factors on both schemes by the same parameters defined in Paper 6.

3.1.2 Delay-Minimization Problem [Paper 4]

In the case of minimizing the network cost, the defined general problem is converted into a flow control problem which allocates BE source rates so that to minimize the sum of delays of all BE traffic flows while maintaining the required QoS. In addition to satisfying link capacity constraints, the sum of BE source rates must be greater than a specified threshold which is construed as minimum expected throughput in the network. The optimization problem is solved using *Projected Gradient Method* for constrained problems and analyzed in terms of convergence behavior.

3.1.3 Implementation Aspects

The proposed algorithms can be implemented as a centralized flow control mechanism by a centralized controller shown in Figure 3.1. The controller can be set up as a separate hardware module or a part of the operating system and must be able to carry out simple mathematical and logical operations needed for running the algorithms. To communicate the algorithm's output (source rate information) to BE sources without delay and loss, a control bus is designed by GS links in conjunction with all sources with light traffic load.

The proposed algorithms have capability of tracking the dynamic conditions. If the network traffic is dynamic in the sense that flows may be dynamically created or deleted, or cores may dynamically join or leave communication tasks, the corresponding source node sends control information to the controller through the control bus and then the algorithm obtains new optimal rates because the addition or subtraction of a flow from the existing set of flows requires re-computation of the rates for all involved flows. Then, the controller sends new optimal rates to corresponding source nodes, if the new value of the rate differs from the previous one. The proposed iterative algorithms do not need to be rerun from the initial phase because they are able to track such a dynamic changes and move towards the new optimal source rates by a few more iteration steps from the previous optimal point. For instance, Paper 11 looks into the behavior of the proposed flow control

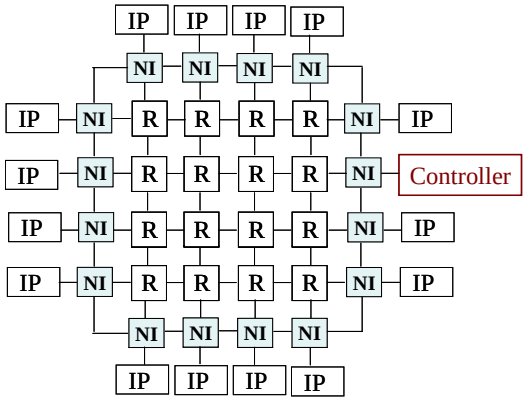


Figure 3.1: The structure of implementation

algorithm in a dynamic condition through a synthetic case study. It is assumed that a source node is activated and starts sending data at the iteration step 140 of the algorithm. The results exhibit that the algorithm can reach the new optimal source rates just after 20 iteration steps, without restarting from its initial points. In this respect, the proposed algorithms are well enough in a time-varying environment with real-time changes. Control packets are responsible for sending control information to the controller and rates to source nodes. They are considered as GS traffic with higher priority than GS data packets to be first served in each node.

3.1.4 Where does the underlying idea come from?

It is worth mentioning that the idea of proposing the flow control algorithms as solutions of optimization problems is inspired by window-based flow control schemes employed in data networks such as flow control in TCP. In this method, each source node maintains a window of packets transmitted but not acknowledged. Since packets in data networks may be lost, destination node should acknowledge the ordered receipt of them in the current window. After receiving the acknowledgment, the source node modifies the window size due to the acknowledgment and thereby avoids the network congestion. Since the source rate during each round trip from source to destination node is the ratio of the window size to the duration of that trip (Round Trip Time), window size can be updated by updating the corresponding rate. Although the proposed flow control algorithms in this section are very similar to rate update in TCP scheme, they have not devised any window-based transmission and acknowledgment mechanism because the NoC architecture is lossless and all packets will be delivered successfully in the correct order and therefore no acknowledgement is needed. To the best of our knowledge, this is the first study which deals with the flow control problem in NoCs using optimization approaches and considers policies to maintain fairness among sources.

3.2 Communication management for real-time systems with guaranteed service

This part of thesis proposes formal approaches to analyze and guarantee QoS for real-time systems with guaranteed service, which can be efficiently utilized to reason about delay and backlog bounds of traffic flows. The proposed approaches apply network calculus to construct an analysis framework. Based on this framework, a contract-based flow regulation is introduced which shapes incoming traffic according to a contract between a client and the network. This contract defines traffic specifications which are the maximum transfer size, peak rate, burstiness and average rate. It also can be used to optimize architectural design decisions such as buffer sizes, arbitration policies etc.

3.2.1 Flow regulation and Performance analysis regardless of VC effects (Papers 8 and 12)

The author of the present thesis uses flow regulation to optimize buffers because buffers are a major source of cost and power consumption [112]. To this end, it is necessary to first derive per-flow bounds on delay and backlog in order to formulate optimization problems.

Applying network calculus, it is possible to derive per-flow delay bound and buffer requirements at each router. Deriving a per-flow delay bound needs an Equivalent Service Curve (ESC) per entire route while deriving a backlog bound requires an ESC per router. The total required number of buffers for a flow can be obtained by summing up the required numbers of buffers at all routers passed by that flow. Then, three timing-constrained buffer optimization problems are defined which called as *buffer size minimization*, *buffer variance minimization*, and *multi-objective optimization* which has both buffer size and variance as minimization objectives. Minimizing buffer variance is formulated to can reuse IP modules and design similar switches as far as possible.

The optimization problems are solved by the interior point method. A realistic case study from Ericsson Radio Systems shows 62.8% reduction of total buffers, 84.3% reduction of total latency, and 94.4% reduction on the sum of variances of buffers. The experimental results also obtain similar improvements for a synthetic traffic pattern. The optimization algorithm is enabling of quick exploration in large design space because of its low run-time complexity.

There are different buffer-dimensioning cases counting on if buffers are finite or infinite (large enough), and if they are shared or not. The underlying system model assumes that the number of VCs in each PC is the same as the number of flows passing through that channel, which means that at most one flow can traverse through each VC. In other words, there is no buffer sharing in the network which is a limitation of this work.

3.2.2 Performance analysis of flows regarding VC effects in network based on aggregate scheduling (Papers 9, 10, and 13)

The previously presented analysis has assumed one virtual channel per flow. This limitation is lifted to address routers in which multiple flows can share a VC and investigate VC effects in the analytical models. To this end, the worst-case performance per flow is analyzed in a FIFO multiplexing and aggregate scheduling network.

To calculate a tight delay bound per flow, it is necessary to obtain the end-to-end ESC which the tandem of routers provides to the flow. As required propositions for calculating performance metrics of traffic characterized with TSPEC and transmitted in the FIFO order and scheduled as aggregate have not been so far represented, papers 9 and 10 apply network calculus to present and prove the required propositions under the mentioned system model. Applying these propositions, Paper 13 proposes a formal approach to calculate end-to-end ESC and then delay bound for a tagged flow in the underlying system. The approach defines two steps which are intra-router ESC and inter-router ESC and employs the results from these steps to calculate the end-to-end ESC. Analysis steps are listed as follows:

1. Different resource sharing scenarios, namely, *channel sharing*, *buffer sharing*, and *channel&buffer sharing*, are defined and the corresponding analysis models are built.

2. Based on these models, the intra-router ESC for an individual flow is derived.

3. Regarding the intra-router ESCs extracted in each router, a mathematical method based on the algebra of sets is presented to classify and analyze flow contention patterns which a flow may experience along its routing path.

4. A formal method applies these analytical models to derive inter-router ESC and in turn end-to-end ESC.

5. Finally, delay bound for the tagged is computed according to the corresponding proposed proposition in paper 10 and the end-to-end ESC calculated for the tagged flow.

The recursive algorithm presented in paper 13 follows steps described for calculating end-to-end ESC. The thesis author has developed algorithms to automate the analysis flow.

To validate the approach, the per-flow delay bounds from the analysis are derived for VOPD, as a real-time application, and compared with per-flow observed maximum delay from detailed simulations. The experimental results exhibit that the maximum relative error with respect to simulation result is about 12.1% which verify the accuracy of the proposed approach. It also provides quick per-flow delay bounds in comparison with detailed simulations. Moreover, compared to previous

models with two parameters, the proposed method improves the accuracy of the delay bounds up to 46.9% and more than 37% on average over all flows. In the case of synthetic traffic patterns, the experimental results show similar improvements.

It is worth mentioning that all previously related works in this subject, as reviewed in Section 2.4.2, compute delay bounds only for average behavior of flows without investigating peak behavior, expectedly resulting in a less tight delay bound. Although the proposed formal method has a good degree of accuracy, it is limited to networks with buffers being larger than the thresholds determined in Paper 13.

3.2.3 Design optimization based on analytical performance models (Paper 14)

In the previous step, the thesis author presumes the routers employ round robin scheduling to share the link bandwidth. Scheduling policy is a critical aspect to ensure sufficient bandwidth and avoid starvation. In order to consider the effect of different scheduling policies, the analysis approach is extended to support weighted round robin policy in the routers. As different values of weights result in different per-flow delay bounds, selecting an appropriate set of values plays an important role in control of the delay bounds in the network.

Since real-time systems need to immediately send data packets, the weights in WRR policy are optimized such that minimize different functions of delay bounds subject to performance constraints. Based on extended analytical models, the thesis defines and formulates two optimization problems, namely, *Minimize-Delay* and *Multi-objective* optimization. *Multi-objective* optimization minimizes both total delay bounds and their variances as an objective function. The proposed problems are solved using genetic algorithm. The thesis author particularly investigates and compares the optimal solutions from four different methods including *Pure Random Search*, *Markov Monotonous Search*, *Adaptive Search*, and *Genetic Algorithm*. A realistic case study shows 15.4% reduction of total worst-case delays and 40.3% reduction on the sum of variances of delays when compared with round robin policy.

Table 3.1: The thesis author's contributions

Topic	My role	Paper	Problem	My contribution	Main limitation(s)
Communication management for BE traffic flows	Secondary contributor	Paper 1	Flow control to maximize a general function of network utility	Design and develop the flow control algorithm from the proposed mathematical solution method	The approach does not deal with all dynamic changes in the network
		Paper 2	Flow control in terms of sum-rate maximization	Design and develop the flow control algorithm from the proposed mathematical solution method	The approach does not deal with all dynamic changes in the network
		Paper 3, 11	Flow control to address proportional fairness in the network	Cooperate in solving the optimization problem and develop the corresponding control algorithm	The approach does not deal with all dynamic changes in the network
	Main contributor	Paper 4	Flow control as a solution of delay optimization	Formulate and solve the optimization problem and propose the flow control algorithm	The approach does not deal with all dynamic changes in the network
		Paper 5	Flow control in terms of max-min fairness in the network	Formulate the problem and propose the flow control algorithm to address max-min fairness in the network	The approach does not deal with all dynamic changes in the network
		Paper 6	Flow control in terms of weighted max-min fairness in the network	Extend the algorithm proposed in Paper 5 to support weighted max-min fairness in the network	The approach does not deal with all dynamic changes in the network
		Paper 7	Consideration of throughput-fairness tradeoffs	Comparison between rate-sum and max-min flow control to study throughput-fairness tradeoffs	The approach does not deal with all dynamic changes in the network

Continued from previous page

Topic	My role	Paper	Problem	My contribution	Main limitation(s)
Communication management for real-time systems with guaranteed service	Main contributor	Paper 8	Optimal traffic regulation based on buffer minimization	Derive per-flow latency and backlog bounds and find the optimized regulator parameters to minimize buffer requirements.	The approach is limited to static regulation and each flow has its own VC without sharing with other flows.
		Paper 12	Buffer optimizations through the traffic regulation	Extend paper 8 to consider more buffer optimizations as a multi-objective problem	The approach is limited to static regulation and each flow has its own VC without sharing with other flows.
		Paper 9	Output process analysis of VBR flows in NoCs with aggregate scheduling	Propose and prove a theorem for analyzing output process analysis	——
		Paper 10	Worst-case delay analysis of VBR flows in NoC with aggregate scheduling	Propose and prove the required theorems for worst-case delay analysis	——
		Paper 13	Evaluate least upper delay bounds for VBR flows in NoCs aggregate scheduling	Propose an analytical model to derive per-flow least upper delay bounds	The model does not consider back-pressure in the network.
		Paper 14	Worst-case delay optimization through weight regulation	Extend the analytical model in Paper 13 to support weighted round robin policy and find the optimized weights for delay minimization.	The model does not consider back-pressure in the network and the approach is limited to static regulation.

Chapter 4

Summary and Outlook

THE findings in the present thesis are briefly concluded in this chapter. The chapter also suggests related future works.

4.1 Summary

Future MPSoCs have to cope with increasingly QoS demands on NoCs including nanometer-scale internal components which compete for available communication resources. Therefore, one of the important questions which should be answered is how communications can be managed to satisfy QoS requirements. This thesis presented novel communication management methodologies for both GS and BE traffic while proposing analytical models for performance evaluation and optimization of on-chip networks.

The following lists the summary of findings in the present thesis:

- The thesis addressed the problem of flow control for BE traffic as solutions to optimization problems with different objectives and QoS constraints. The proposed problems have been solved through mathematical approaches led to iterative flow control algorithms. The experimental results exhibited the behavior of the proposed algorithms in static and dynamic conditions and confirmed their fast convergence. The thesis also looked into the effect of different objective functions on the BE rates in terms of different features like fairness and throughput.

- As it is necessary to ensure QoS under worst-case conditions for real-time systems with guarantee services, this thesis proposed formal approaches for worst-case performance analysis applying network calculus.

  - The author of the thesis derived the analysis procedure of per-flow delay and backlog bounds to reason about the optimal buffer size under traffic regulation.

- The author expanded the analytical approach for a FIFO multiplexing network with aggregate scheduling taking into account VC sharing in routers. To this end, we proposed and proved network calculus-based theorems and then applied them to present analytical models.
- Further extension on the proposed analytical models presented to support weighted round robin scheduling policy. This makes it possible to investigate the effect of weights on an efficient resource allocation in terms of minimizing objective functions of delay bounds.
- The author also automated the analysis steps of the proposed approaches to be used as a performance evaluation and optimization tool.

4.2 Outlook

- **Further extensions on analytical models**

  The proposed analytical approach has the potential to be extended to more features like other arbitration policies and routing algorithms. In the case of arbitration policy, the thesis has focused on the fair arbitration policies while unfair policies such as priority-based scheduling policies are also important since application traffic may be classified into different priority classes. The thesis looks into only deterministic routing algorithm; it is worth developing the analytical models for adaptive and partially adaptive routings.

- **Further optimization opportunities for flow regulation**

  Future works on the present thesis can be continued to consider further optimization opportunities offered by flow regulation such as optimal arbitration policy or routing selections. Optimizations for multiple objectives can be investigated for analyzing tradeoffs and preferences between different network metrics.

- **Mixing of real-time and best-effort traffic**

  The present thesis considers the management of shared communication fabric for real time and best-effort traffic, separately. Each of these services has its own requirements and differs somewhat in purpose. One direction for future work would be integrating analytical approaches to explicate mixed real-time and best-effort traffic. As best-effort traffic and non-critical real-time flows may desire stochastic guarantees, a stochastic regulation with stochastic parameters gives the ability of better dealing with networks with mixed real-time and best-effort traffic. Stochastic guarantees can give better utilization of resources in the network without putting performance at risk.

- **Dynamic regulation**

  The present thesis has considered a static flow regulator for traffic flows with guaranteed services. The static regulation may not employ network resources

efficiently. Moreover, it is not sufficient to support traffic dynamism like when a flow is dynamically created or deleted, or a core may dynamically join or leave NoC; because these changes cause different bounds for all involved flows. Thus, another direction for future work would be focusing on dynamic regulation mechanism with online feedback information.

- **QoS framework with the analysis of multiple resources sharing**

  Most existing approaches in the subject of QoS management work well on one or two individual resources. For example, the works that focus on cache management ignore contentions in the communication management. Since future MPSoC is expected to have architectures with tens of heterogeneous agents sharing cache, bandwidth and memory simultaneously, a guarantee of one individual resource is not sufficient to support the overall performance. Thus, it becomes important to find a QoS framework analyzing multiple resources sharing and coordinating the management of them.

- **Large-scale interconnection networks**

  The thesis has concentrated on small-scale NoCs of a few IP blocks. An interesting idea on this research is providing a general analysis framework for large-scale interconnection networks with both off-chip and on-chip environment, each with its own design constraints. Development of a general tool in this subject can be employed for different kinds of interconnection networks such as GALS SoCs, data centers, clusters, and supercomputers.

Bibliography

[1] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Öberg, M. Millberg, And D. Lindqvist, "Network on a Chip: An architecture for billion transistor era", Proceedings of the Norchip Conference.

[2] Li-Shiuan Peh, Stephen W. Keckler, Sriram Vangal, "On-Chip Networks for Multicore Systems", *Multicore Processors and Systems Integrated Circuits and Systems 2009*, pp 35-71.

[3] Hu. Jingcao and R. Marculescu, DyAD-smart routing for networks-on-chip, Automation Conference (2004), pp. 260-263.

[4] I. Saastamoinen and J. N. M. Alho, "Buffer implementation for proteo networks-on-chip", *in Proc. Int. Symp. Circuits and Syst.*, May 2003, pp. 113-116.

[5] R. Marculescu, U. Y. Ogras, L. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems archive (TCAD)*, Vol. 28, no. 1, pp. 3-21, 2009.

[6] R. Dick, "Embedded System Synthesis Benchmarks (E3S)," http://www.ece.northwestern. edu/ dickrp/e3s, 2007.

[7] G. De Micheli, L. Benini, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, 1st edition, 2006.

[8] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: concepts, architectures, and implementations," IEEE Trans. Design and Test, vol. 22, no. 5, pp. 414-421, 2005.

[9] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip", in Proc. Des., Autom. Test Eur. Conf., Feb. 2004, pp. 890-895.

[10] W.-D. Weber, J. Chou, I. Swarbrick, and D. Wingard, "A quality-of-service mechanism for interconnection networks in system-on-chips," in Proc. of Conference on Design, Automation and Test in Europe, 2005, pp. 1232-1237.

[11] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in Proc. Int. Symp.Comput. Architecture, 2008, pp. 89-100.

[12] B. Grot, S. W. Keckler, and O. Mutlu, "Preemptive virtual clock: a flexible, efficient, and cost-effective QoS scheme for networks-on-chip," in Proc. of International Symposium on Microarchitecture, 2009, pp. 268-279.

[13] S. Nasri, "New Approach of QoS Metric Modeling on Network on Chip", Int. J. Communications, Network and System Sciences (IJCNS), 4, pp. 351-355, 2011.

[14] N. Rameshan, M. Ahmed, M.S. Gaur, V. Laxmi, and A. Biyani, "QoS Aware Minimally Adaptive XY routing for NoC", In Proc. of 17th International Conference on Advanced Computing and Communication (ADCOM), Bangalore, India (2009)

[15] P. Vellanki, N. Banerjee and K. S. Chatha, "Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures", In Proc. of the 14th ACM Great Lakes symposium on VLSI (GLSVLSI), pp. 45-50, 2004.

[16] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Evaluation of current QoS mechanisms in network on chip", in Intl. Symposium on Soc, Nov. 2006, pp. 115-118.

[17] T. Bjerregaard and J. Sparso, "A router architecture for connectionoriented service guarantees in the MANGO clockless network-on-chip", in Proc. Des., Autom. Test Eur. Conf., Mar. 2005, pp. 1226-1231.

[18] K. Goossens et al., "A design flow for application-specific networks on chip with guaranteed performance to accelerate SoC design and verification", in Proc. Des., Autom. Test Eur. Conf., Mar. 2005, pp. 1182-1187.

[19] L. F. Leung and C. Y. Tsui, "Optimal link scheduling on improving best-effort and guaranteed services performance in network-on-chip systems", in Proc. Des. Autom. Conf., Jul. 2006, pp. 833-838.

[20] J. Liang, A. Laffely, S. Srinivasan, and R. Tessier, "An architecture and compiler for scalable on-chip communication", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 7, pp. 711-726, Jul. 2004.

[21] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework", in Proc. Int. Symp. Asynchronous Circuits Syst., May 2005, pp. 54-63.

[22] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip", J. Syst. Architecture: EUROMICRO J., vol. 50, no. 2/3, pp. 105-128, Feb. 2004.

[23] M. Harmanci, N. Escudero, Y. Leblebici, and P. Ienne, "Quantitative modeling and comparison of communication schemes to guarantee quality-of-service in networks-on-chip," in Proc. Int. Symp. Circuits Syst., May 2005, pp. 1782-1785.

[24] T. Marescaux and H. Corporaal, "Introducing the superGT network-onchip", in Proc. Des. Autom. Conf., Jun. 2007, pp. 116-121.

[25] J. W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-controlled best-effort communication for networks-onchip," in Proc. Des., Autom. Test Eur. Conf., Apr. 2007, pp. 948-953.

[26] J. Duato et al., "A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks," in Proc. Int. Symp. High-Performance Comput. Architecture, Feb. 2005, pp. 108-119.

[27] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network on chip," in Proc. Des., Autom. Test Eur. Conf., Mar. 2003, pp. 1126-1127.

[28] U. Y. Ogras and R. Marculescu, "Analysis and optimization of prediction-based flow control in networks-on-chip," ACM Trans. Des. Autom. Electron. Syst., vol. 13, no. 1, pp. 1-28, Jan. 2008.

[29] W. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In Proc. DAC, June 2001.

[30] A. Jalabert, et al. XpipesCompiler: A tool for instantiating application specific networks on chip. In Proc. DATE, March 2004.

[31] J. Hu and R. Marculescu, Energy- and performance-aware mapping for regular NoC architectures, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol.24, No.4, April 2005.

[32] E. Nilsson et. al. Load distribution with the proximity congestion awareness in a network on chip. In Proc. DATE, March 2003.

[33] A. Radulescu et. al. An efficient on-chip ni offering guaranteed services, shared-memory abstraction, and flexible network configuration, IEEE Trans. on CAD of ICs and Systems, 24(1) 2005.

[34] C. A. Zeferino et. al. Paris: a parameterizable interconnect switch for networks-on-chip, In Proc. Symp. on IC and Systems Design, 2004.

[35] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip", Journal of Systems Architecture, Volume 50, Issue 2-3 (Special Issue on Network on Chip), pp. 105-128, February 2004.

[36] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Cost considerations in Network on Chip", Integration: The VLSI Journal, no. 38, 2004, pp. 19-42.

[37] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip", IEEE Circuits and Systems Magazine, vol. 4, Issue 2, pp. 18-31, 2004.

[38] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip", IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 2, pp. 113-129, 2005.

[39] M. Dall'Ossa, G. Biccari, L. Giovannini, L. D. Bertozzi, and L. Benini, "XPIPES: A latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs", Proc. 21st International IEEE Conference on Computer Design, pp. 536-539, 2003.

[40] C. A. Zeferino and A. A. Susin, "SoCIN: A parametric and scalable network-on-chip", Proc. 16th Symposium on Integrated Circuits and Systems Design, pp. 169-175, 2003.

[41] U.Y. Ogras and R. Marculescu, Prediction-based flow control for network-on-chip Design Automation Conference (2006), pp. 839-844.

[42] J.W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, Congestion-controlled best-effort communication for networks-on-chip, Design, Automation and Test in Europe Conference (2007), pp. 948-953.

[43] Y. Gu, H.O. Wang, and Y. Hong, A predictive congestion control algorithm for high speed communication networks, Am. Control Conf. 5 (2001), pp. 3779-3780.

[44] F.P. Kelly, A. Maulloo, and D.K.H. Tan, Rate control for communication networks: Shadow prices, proportional fairness, and stability, Oper. Res. Soc. 49(3) (1998), pp. 237-252.

[45] S. Mascolo, Classical control theory for congestion avoidance in high-speed internet, Conf. Decis. Control 3 (1999), pp. 2709-2714.

[46] C. Yang and A.V.S. Reddy, A taxonomy for congestion control algorithms in packet switching networks, IEEE Netw. 9(4) (1995), pp. 34-45.

[47] D. Bertsekas and R. Gallager, Data Networks. Prentice Hall, 1992.

[48] A. Radulescu et. al. An efficient on-chip ni offering guaranteed services, shared-memory abstraction, and flexible network configuration, IEEE Trans. on CAD of ICs and Systems, 24(1) 2005.

[49] A. Pullini et. al. Fault tolerance overhead in network-on-chip flow control schemes. In Proc. Symp. on IC and System Design, September 2005.

[50] Z. Qian, D. Juan, P. Bogdan, C. Tsui, D. Marculescu, and R. Marculescu, "Svr-noc: A performance analysis tool for network-on-chips using learning-based support vector regression model," in ACM/IEEE Design Automation and Test in Europe (DATE), 2013.

[51] P. Gratz and S. W. Keclker, "Realistic Workload Characterization and Analysis for Networks-on-Chip Design," in The 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI), 2010.

[52] Y. Ben-Itzhak, I. Cidon, and A. Kolodny, "Delay analysis of wormhole based heterogeneous noc," in Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on, 2011, pp. 161-168.

[53] G. Varatkar and R. Marculescu, "On-chip traffic modeling and synthesis for mpeg-2 video applications," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 12, no. 1, pp. 108-119, 2004.

[54] P. Bogdan and R. Marculescu, "Non-stationary traffic analysis and its implications on multicore platform design," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 30, no. 4, pp. 508-519, 2011.

[55] P. Bogdan, R. Marculescu, "Statistical physics approaches for network-on-chip traffic characterization," in Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis (CODES+ISSS), 2009, pp. 461-470.

[56] BAKHOUYA, M., SUBOH, S., GABER, J.,EL-GHAZAWI, T., AND NIAR, S. 2011. Performance evaluation and design tradeoffs of on-chip interconnect architectures. Simulation Modelling Practice and Theory, Elsevier. 19, 6, 1496-1505.

[57] LE BOUDEC, J. Y. AND THIRAN, P. 2004. Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag.

[58] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, John Kim and William J. Dally. A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator. In Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). 86-96.

[59] http://research.cs.tamu.edu/codesign/nocsim/

[60] Z. Lu, R. Thid, M. Millberg, E. Nilsson, and A. Jantsch. NNSE: Nostrum network-on-chip simulation environment. In Swedish System-on-Chip Conference (SSoCC'03), pages 1-4. Citeseer, 2005.

[61] F. Fazzino, M. Palesi, and D. Patti. Noxim: Network-on-chip simulator, 2008.

[62] Worm_Sim: a cycle accurate simulator for networks-on-chip. http://www.ece.cmu.edu/ sld/software/worm_sim.php

[63] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Islam, and M. Akbar. Gpnocsim-A General Purpose Simulator for Network-On-Chip. In Information and Communication Technology, 2007. ICICT'07. International Conference on, pages 254-257. IEEE, 2007.

[64] A. Nayebi, S. Meraji, A. Shamaei, H. Sarbazi-Azad, "XMulator: A Listener-Based Integrated Simulation Platform for Interconnection Networks", in Proceedings of First Asia International Conference on Modelling & Simulation (AMS '07), pp. 128-132, 2007.

[65] L. Jain, B. Al-Hashimi, M. Gaur, V. Laxmi, and A. Narayanan. NIRGAM: a simulator for NoC interconnect routing and application modeling. In Workshop on Diagnostic Services in Network-on-Chips, Design, Automation and Test in Europe Conference (DATE'07), pages 16-20, 2007.

[66] M. Lis, K. Shim, M. Cho, P. Ren, O. Khan, and S. Devadas. DARSIM: a parallel cycle-level NoC simulator. In 6th Annual Workshop on Modeling, Benchmarking and Simulation. Citeseer, 2010.

[67] V. Puente, J. Gregorio, and R. Beivide. SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems. In Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on, pages 15-22. IEEE, 2002.

[68] P. Abad, P. Prieto, L. Menezo, A. Colaso, V. Puente, J.A. Gregorio, "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers", in Proceedings of IEEE/ACM International Symposium on Networks on Chip (NOCS), pp. 99-106, 2012.

[69] A. E. Kiasari, Z. Lu, and A. Jantsch, "An analytical latency model for networks-on-chip," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, pp. 113-123, Jan. 2013.

[70] D. Rahmati, S. Murali, L. Benini, F. Angiolini, G. De Micheli, and H. Sarbazi-Azad, "Method for calculating hard qos guarantees for networks-on-chip," in Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on, 2009, pp. 579-586.

[71] Yue Qian, Zhonghai Lu and Wenhua Dou, "Analysis of Worst-Case Delay Bounds for On-Chip Packet-Switching Networks", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 29 no. 5, pp. 802-815, 2010.

[72] U. Ogras, P. Bogdan, and R. Marculescu, "An analytical approach for network-on-chip performance analysis," Computer-Aided Design of IntegratedCircuits and Systems, IEEE Transactions on, vol. 29,no. 12, pp. 2001-2013, 2010.

[73] G. Min and M. Ould-Khaoua, "A performance model for wormhole-switched interconnection networks under self-similar traffic," Computers, IEEE Transactions on, vol. 53, no. 5, pp. 601-613, 2004.

[74] Hansson, M. Wiggers, A. Moonen, K. Goossens, and M. Bekooij. Applying dataflow analysis to dimension buffers for guaranteed performance in networks on chip. NOCS Proc., pages 211-212, 2008.

[75] Rahmati, D., Murali, S., Benini, L., Angiolini, F., Demicheli, G., and Sarbazi-Azad, H. A method for calculating hard QoS guarantees for Networks-on-Chip. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'09). pp. 579-586.

[76] Rahmati, D., Murali, S., Benini, L., Angiolini, F., Demicheli, G., and Sarbazi-Azad, H. Computing Accurate Performance Bounds for Best Effort Networks-on-Chip. IEEE Transactions on Computers (IEEE-TC). Vol. 62,No. 3, pp. 452-467

[77] M. Bakhouya, S. Suboh, J. Gaber, T. El-Ghazawi, and S. Niar. Performance evaluation and design tradeoffs of on-chip interconnect architectures. Simulation Modelling Practice and Theory, Elsevier, 19(6):1496-1505, 2011.

[78] F. Ciucu and J. Schmitt, Perspectives on Network Calculus - No Free Lunch but Still Good Value, ACM Sigcomm 2012.

[79] Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu. Mathematical formalisms for performance evaluation of networks-on-chip. ACM Computing Surveys, 45(3), 6 2013.

[80] J. Turner. New Directions in Communications (or Which Way to the Information Age?), IEEE Communications Magazine, vol. 24 no. 10, pp. 8-15, Oct 1986.

[81] M. Bakhouya, S. Suboh, J. Gaber, T. ElGhazawi, and S. Niar. Performance evaluation and design tradeoffs of on-chip interconnect architectures. Simulation Modelling Practice and Theory, Elsevier, 19(6):1496-1505, 2011.

[82] Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An architecture for differentiated services", IETF RFC 2475, 1998.

[83] J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, J.-Y. Le Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding", *IEEE/ACM Transactions on Networking* vol. 10, no. 4, pp. 529-540, 2002.

[84] A. Charny, J.-Y. Le Boudec, Delay bounds in a network with aggregate scheduling, in Proceedings of QoFIS'00, 25-26 September 2000, Berlin, Germany, in: LNCS, vol. 1922, Springer-Verlag, 2000, pp. 1-13.

[85] Y. Jiang, "Delay bounds for a network of guaranteed rate servers with FIFO aggregation", *Computer Networks*, vol. 40, no. 6, pp. 683-694, 2002.

[86] Bauer H., Scharbarg J.-L., Fraboul C., "Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach", IEEE Transactions on Industrial Informatics 6(4), Nov. 2010, pp. 521-533

[87] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch ... In Proceedings of INFOCOM'2008, 2008.

[88] Kiefer A., Gollan N., Schmitt J.B., "Searching for Tight Performance Bounds in Feed-Forward Networks", in Proc. of MMB/DFT 2010: 227-241.

[89] A. Bouillard, L. Jouhet, and E. Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In Proceedings of Infocom'2010, 2010.

[90] A. Bouillard and A. Junier. Worst-case delay bounds with fixed priorities using network calculus,. In Proceedings of Valuetools'11, 2011.

[91] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks", *Performance Evaluation*, vol. 63, no. 9, pp. 956-987, 2006.

[92] L. Lenzini, E. Mingozzi, G. Stea, "A Methodology for Computing End-to-end Delay Bounds in FIFO-multiplexing Tandems" Elsevier Performance Evaluation, 65 (2008) 922-943.

[93] L. Bisti, L. Lenzini, E. Mingozzi, G. Stea, "DEBORAH: A Tool for Worst-case Analysis of FIFO Tandems", ISoLA 2010, Special Track on Worst-case Traversal Time, Crete, GR, October 18, 2010

[94] M. Boyer, "Half-modelling of shaping in FIFO net with network calculus", RTNS 2010

[95] Y. Qian, Z. Lu, W. Dou, "Analysis of Worst-case Delay Bounds for Best-effort Communication in Wormhole Networks on Chip", Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp. 44-53, 2009.

[96] Y. Qian, Z. Lu, and Q. Dou, "QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin", *In Proceedings of the 28th International Conference on Computer Design (ICCD'10)*, pp. 52-59, Amerstedam, the Netherlands, 2010.

[97] Z. Lu, Y. Yao, and Y. Jiang, "Towards Stochastic Delay Bound Analysis for Network-on-Chip", In Proceedings of the Eighth ACM/IEEE International Symposium on Networks-on-Chip (NoCS'2014), Ferrara, Italy, September 2014.

[98] K. Srinivasan and K. S. Chatha, "A technique for low energy mapping and routing in network-on-chip architectures," in Proc. Int. Symp. Low Power Electron. Des., Aug. 2005, pp. 387-392.

[99] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," in Proc. Int. Conf. Hardware-Softw. Codesign Syst. Synthesis, Sep. 2004, pp. 182-187.

[100] W. Hung et al., "Thermal-aware IP virtualization and placement for networks-on-chip architecture," in Proc. Int. Conf. Comput. Des., 2004, pp. 430-437.

[101] J. Hu and R. Marculescu, "Communication and task scheduling of application-specific networks-on-chip," Proc. Inst. Elect. Eng.-Comput. Digit. Tech., vol. 152, no. 5, pp. 643-651, Sep. 2005.

[102] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network on chip," in Proc. Des., Autom. Test Eur. Conf., Mar. 2003, pp. 1126-1127.

[103] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," IEEE Trans. Comput.-Aided Design Integr.Circuits Syst., vol. 24, no. 4, pp. 551-562, Apr. 2005.

[104] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in Proc. Des., Autom. Test Eur. Conf.,Feb. 2004, pp. 896-901.

[105] L. Peh and W. J. Dally, "Flit-reservation flow control," in Proc. Int. Symp. High-Performance Comput. Architecture, Jan. 2000, pp. 73-84.

[106] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in Proc. Int. Symp. Comput. Architecture, Jun. 2004, pp. 188-197.

[107] L. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in Proc. Int. Symp. High-Performance Comput. Architecture, Jan. 2001, pp. 255-266.

[108] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press.

[109] I. E. Grossmann and Z. Kravanja, "Mixed-integer nonlinear programming: A survey of algorithms and applications", *In Biegler, Coleman, Conn, Santosa (Eds.), The IMA volumes in mathematics and its applications: Large-scale optimization with applications*, Vol. 93, Part II, Optimal design and control, pp. 73-100, Berlin: Springer-Verlag, 1997.

[110] F. Rossi, P. van Beek, and T. Walsh. Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA, 2006.

[111] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[112] Erik Jan Marinissen, B. Prince, D. Keitel-Schulz, and Y. Zorian, "Challenges in Embedded Memory Design and Test", In Proceeding of the conference on Design, Automation and Test in Europe (DATE), pp. 722-727, March 2005.

Part II

Included Papers

Paper 1

A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization

M. S. Talebi
F. Jafari
A.Khonsari

In the Proceedings of the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 381-386, Istanbul, Turkey, October 2007.

A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization

Mohammad S. Talebi
School of Computer Science, IPM
mstalebi@ipm.ir

Fahimeh Jafari
School of Computer Science, IPM
Ferdowsi University of Mashhad
jafari@ipm.ir

Ahmad Khonsari
School of Computer Science, IPM
ECE Department, University of Tehran
ak@ipm.ir

***Abstract*—Advances in semiconductor technology, has enabled designers to put complex, massively parallel multiprocessor systems on a single chip. Network on Chip (NoC) that supports high degree of reusability and scalablity, is a new paradigm for designing core based System-on-Chip. NoCs provide efficient communication services to IPs: communication services with guarantees on throughput and latency (GS) and communication services with no guarantees on them (BE). However, the run-time management of communication in NoC, especially congestion control mechanism is a challenging task. This paper considers a congestion control scenario which models flow control as a utility-based optimization problem. Since BE traffic is prone to congestion, we assume that GS traffic requirements are being preserved at the desired level and regulate BE source rates with the solution of the optimization problem. We propose an iterative algorithm to solve the optimization problem based on Newton's method. The proposed algorithm can be implemented by a centralized controller with low computation and communication overhead.**

Keywords-congestion control; Network-on-Chip; utility-based optimization; iterative algorithm

I. INTRODUCTION

Network on Chip (NoC) is a new paradigm structure for designing future *System on Chips* (SoCs) [1,2], where various IP resource nodes are connected to the router based square network of switches using Resource Network Interfaces, and network is used for packet switched on-chip communication among cores [3]. A typical NoC architecture will provide a scalable communication infrastructure for interconnecting cores. Since the communication infrastructure as well as the cores from one design can be easily reused for a new product, NoC provides maximum possibility for reusability. NoC-based system will be much easily used for design, test and production. NoCs are efficient communication architectures. However the run-time management of their communication, especially congestion avoidance is a challenging task. Congestion control has been already the subject of research in the field of NoC. Furthermore, minimizing the network cost (or maximizing network utility) while maintaining the required *Quality of Service* (QoS) is one of the considerable factors in NoC architecture design.

NoCs provide two types of communication services to IPs: *Guaranteed Service* (GS) and *Best-Effort* (BE) [4]. Guaranteed Service requires reservation of resources so as to insure data integrity, lossless and ordered data delivery, while Best-Effort service does not require any reservation of resources and no assurance are meant to be given. BE services are easy to use, while GS services require careful programming to reserve the required resources in the network

During the past two decades, several strategies for congestion control have been proposed for data networks [5-8]. However, this issue for Network-on-Chip systems is still novel and only a few works exist. [9] has proposed a a flow control strategy for on-chip networks based on prediction of future congestion problems by routers. In [10], a controller has been proposed to determine the appropriate loads for the Sources with Best Effort traffic. Dyad [11] control the congestion by employing adaptive routing during congestion phase.

The aforementioned works in this issue for NoC ([9]-[11]) mainly used prediction-based method to control the flow of sources which are prone to congestion. In contrast, we have applied a different approach. In this paper, we model the flow control as a utility-based maximization problem which is constrained by link capacities. We assume GS services are being preserved at the desired level and rate allocation of BE sources is the main role of the optimization problem. We mainly adopt the framework provided by [8] for data networks.

The rest of the paper is organized as follows: in the next section we present the system model and flow control problem. In section III, we obtain the dual of the optimization problem that motivates our approach. In Section IV, we solve the dual problem using Newton's Method present the resultant congestion control algorithm. The simulation results are given in section V and finally, section VI concludes the paper.

II. SYSTEM MODEL

Our NoC architecture is based on a two dimensional mesh topology and wormhole routing. In wormhole networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in pipeline fashion. A hop-to-hop credit mechanism assures that a flit is transmitted only when the receiving port has free space in its input buffer. Our

382 NoC architecture is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing [3].

High performance wormhole based interconnect systems often include *virtual channels* (VCs) which increase NoC throughput. Furthermore, virtual channels must be included when links have different capacities to allow the multiplexing of several slow streams over a high bandwidth link. Flits of different VCs that contend for the same link bandwidth are time-multiplexed according to some arbitration policy. Our architecture employs a simple policy in which flits of the active outgoing VCs are transmitted in a round-robin manner over the physical link.

We model the congestion control problem in NoC as the solution to an optimization problem. To have more convenience, we turn the aforementioned NoC architecture into a mathematical model as in [8]. In this respect, we consider NoC as a network with a set of links L and a set of sources S. A source consists of Processing Elements (PEs) and Input/Output ports. Each link $l \in L$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of c_l packets/sec. We also denote the set of sources that share link l by $S(l)$. Similarly, the set of links that source s passes through, is denoted by $L(s)$.

As previously stated, there are two types of traffic in a NoC: Guaranteed service (GS) and Best Effort (BE) traffic. For notational convenience, we divide the set of sources, S, into two parts, each one representing sources with the same traffic. In this respect, we denote the set of sources with BE and GS traffic by S_{BE} and S_{GS}, respectively. Each link l is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE sources.

Our objective is to choose source rates, x_s, of BE traffics so that to maximize the sum of utilities of all BE traffics. Hence the maximization problem can be formulated as [8]:

$$\max_{x_s} \sum_{s \in S_{BE}} U_s(x_s) \tag{1}$$

subject to:

$$\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{2}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{3}$$

where U_s is a positive, concave and strictly increasing function of source rate. Optimization variables are BE source rates, i.e. $(x_s,\ s \in S_{BE})$. U_s is monotonic and we also assume that the curvatures of U_s satisfy the following condition:

$$-U_s^{"}(x_s) \ge \frac{1}{\alpha_s} > 0 \qquad \forall s \in S_{BE} \tag{4}$$

The constraint (2) states that the sum of BE source rates passing through link l cannot exceed its free capacity, i.e. the portion of c_l which hasn't been allocated to GS traffic.

U_s in the economics literature is referred to as *utility function*, hence problem (1) is called a utility-maximization problem. There are many choices for utility function with specific features and behavior. The simplest form of the utility function is the *Identity Function*, i.e. $U_s(x_s) = x_s$, for which the problem (1) turns into a sum-rate maximization. One of the popular forms of utility functions is logarithmic one, which satisfy *Proportional Fairness* [15]. In this paper, we will consider a general utility function and will not restrict ourselves to a specific form. The investigation of the features of popular utility functions on the rates chosen is one of the directions of our future work.

With the above assumptions, problem (1) is a convex optimization problem with linear constraints. Hence it admits a unique maximizer [12][13], i.e. there exists an optimal source rate vector, $x^* = (x_s^*, s \in S_{BE})$ so that to maximize the sum of utilities in problem (1) while satisfying capacity constraints.

Although problem (1) is separable among sources, its constraints will remain coupled across the network. The coupled natured of such constrained problems, necessitate usage of centralized methods like interior point methods which pose great computational overhead onto the system [12][13].

One way to reduce the computational complexity is to transform the constrained optimization problem into an unconstrained one, for which several methods can be used. According to the Duality Theory [12][13], each convex optimization problem has a dual whose optimal solution can lead to the optimal solution of the main problem. In this respect, the main problem retroactively called *Primal Problem*. As the dual problem can be defined in such a way to be unconstrained, solving the dual is much simpler than the primal. In the sequel, we will obtain the dual of problem (1) and solve it using simple iterative algorithms.

For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in S_{GS}(l)} x_s \tag{5}$$

Using the standard optimization methods [12], the Lagrangian of the problem (1) can be written as:

$$L = \sum_{s \in S_{BE}} U_s(x_s) - \sum_{l=1}^{L} \lambda_l \left(\sum_{s \in S_{BE}(l)} x_s - \hat{c}_l \right) \tag{6}$$

where $\lambda_l > 0$ is the Lagrange Multiplier associated with constraint (2) for link l. Usually, λ_l is called *shadow price* [15] for the economic interpretation of its role in solving the primal problem through dual.

Regarding the Lagrangian of problem (1), the dual function is defined as [12]:

$$g(\lambda) = \max_{x_s} L(x, \lambda) \tag{7}$$

where λ is the vector of positive Lagrange multipliers. Thus the dual function is given by:

$$\begin{aligned} g(\lambda) &= \max_{x_s} \sum_{s \in S_{BE}} U_s(x_s) - \sum_{l=1}^{L} \lambda_l \left(\sum_{s \in S_{BE}(l)} x_s - \hat{c}_l \right) \\ &= \max_{x_s} \sum_{s \in S_{BE}} \left(U_s(x_s) - x_s \sum_{l \in L(s)} \lambda_l \right) + \sum_{l=1}^{L} \lambda_l \hat{c}_l \end{aligned} \tag{8}$$

By Karush-Kuhn-Tucker (KKT) Theorem [12][13], we can obtain optimal source rates, i.e. $x^* = (x_s^*, s \in S_{BE})$. In doing so, we should find the roots of $\nabla_x L(x, \lambda) = 0$. By taking the derivative of (6) with respect to x_s, we have

$$\frac{\partial L}{\partial x_s} = \frac{\partial U_s(x_s)}{\partial x_s} - \sum_{l \in L(s)} \lambda_l \tag{9}$$

Duality theory states that the optimal source rate vector, x^*, corresponds to the optimal Lagrange multiplier vector, λ^* [12][13]. In other words, if x is a feasible point of the primal problem and x is primal-optimal, the corresponding λ will be dual-optimal and vice versa. Therefore, at optimality we have

$$\nabla_x L(x, \lambda)\Big|_{(x^*, \lambda^*)} = \mathbf{0} \tag{10}$$

where $\mathbf{0}$ is a vector with all zero. From (9), we have

$$\frac{\partial L}{\partial x_s}\Big|_{(x_s^*, \lambda^*)} = \frac{\partial U_s(x_s^*)}{\partial x_s^*} - \sum_{l \in L(s)} \lambda_l^*$$

Hence, the optimal source rate is given by

$$x_s^* = f\left(\sum_{l \in L(s)} \lambda_l^* \right) \tag{11}$$

where f is the inverse function of $U_s^{'}$ whose existence is guaranteed by monotonicity of U_s in strict sense.

Substituting x_s^* into (8) yields

$$g(\lambda) = \sum_s \left(U_s(x_s^*) - x_s^* \sum_{l \in L(s)} \lambda_l \right) + \sum_{l=1}^{L} \lambda_l \hat{c}_l \tag{12}$$

where x_s^* is given by (11).

The dual problem is defined as [13]:

$$\min_{\lambda \geq 0} g(\lambda) \tag{13}$$

The dual problem is always convex regardless of convexity or non-convexity of the primal problem. Moreover, the dual problem can be defined to be unconstrained or constrained with simple constraints. Thus, the primal problem has been transformed into an unconstrained convex optimization problem.

Convexity of the primal problem (1) guarantees strong duality. Thereby the duality gap is zero and solving the dual problem leads to optimal point of the primal [12]. Since dual problem is convex, it admits a unique optimum, i.e. a unique minimizer, which can be obtained using optimization algorithms. As the dual problem is unconstrained; solving (13) using search methods is much simpler than the primal.

There exist several methods to search the optimal point of an unconstrained optimization problem iteratively [12]. One famous and simple ones is Gradient Projection Method [12] which uses simple mathematical operations. Another famous one is Newton Method that has better convergence behavior at the expense of higher computational complexity [12]. Due to need for faster convergence, in this paper we use the Newton's Method to solve problem (13).

For notational convenience in solving the problem using the Newton's Method, in the rest of the paper we may use matrix notation. To this end, we define Routing matrix, i.e. $R = [R_{ls}]_{L \times S}$, as following:

$$R_{ls} = \begin{cases} 1 & \text{if } s \in S_{BE}(l) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

We also define the source rate vector (for BE traffic) and link capacity vector as $x = (x_s, s \in S_{BE})$ and $\hat{c} = (\hat{c}_l, l \in L)$, respectively.

III. Flow Control For Best Effort Sources

In this section, we will solve the dual problem using Newton's Method [12] and present a congestion control mechanism to be run for BE traffic by a controller in NoC systems.

The Newton's Method adjusts shadow prices, i.e. Lagrange multiplier vector, in opposite direction to the scaled version of gradient of the dual function as follows [12]:

$$\lambda(t+1) = \left[\lambda(t) - \gamma(t) [\nabla^2 g(\lambda(t))]^{-1} \nabla g(\lambda(t)) \right]^+ \tag{15}$$

where $\lambda(t+1) = \left(\lambda_l(t+1),\ l \in L\right)$, $\gamma(t) > 0$ is a stepsize, $[x]^+ \triangleq \max\{x, 0\}$ and $\nabla^2 g(\lambda(t))$ is the Hessian of $g(\lambda)$. Since U_s is strictly concave, $g(\lambda)$ is continuously differentiable [13], hence $\nabla g(\lambda)$ exists. Using (14), the l-th element of the gradient vector is given by:

384

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \frac{\partial}{\partial \lambda_l} \sum_s \left(U_s(x_s^*) - x_s^* \sum_{l \in L(s)} \lambda_l \right) + \hat{c}_l \tag{16}$$

Regarding the system model, we have

$$\sum_s x_s^* \sum_{l \in L(s)} \lambda_l = \sum_l \lambda_l \sum_{s \in S_{BE}(l)} x_s^*$$

Therefore,

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \hat{c}_l - \sum_{s \in S_{BE}(l)} x_s^* \tag{17}$$

or equivalently in the matrix form

$$\nabla g(\lambda) = \hat{c} - Rx \tag{18}$$

To obtain the Hessian of $g(\lambda)$, we have

$$\nabla^2 g(\lambda) = -R \nabla_\lambda x \tag{19}$$

or equivalently,

$$\frac{\partial^2 g(\lambda)}{\partial \lambda_k \partial \lambda_l} = -\frac{\partial}{\partial \lambda_k} \sum_s R_{ks} x_s^* \tag{20}$$

Substituting (11) into above equation, yields

$$\begin{aligned} \frac{\partial^2 g(\lambda)}{\partial \lambda_k \partial \lambda_l} &= -\frac{\partial}{\partial \lambda_k} \sum_s R_{ks} x_s^* \\ &= -\frac{\partial}{\partial \lambda_k} \sum_s R_{ks} f\left(\sum_{l \in L(s)} \lambda_l^* \right) \\ &= -\frac{\partial}{\partial \lambda_k} \sum_s R_{ks} f\left(\sum_l R_{ls} \lambda_l^* \right) \end{aligned} \tag{21}$$

Using the rule of derivation for inverse function, we have

$$\frac{\partial^2 g(\lambda)}{\partial \lambda_k \partial \lambda_l} = -\sum_s \sum_l R_{ls} R_{ks} \left(\frac{1}{U_s''(x_s^*)} \right)$$

Defining $F(t)$ as the following

$$F(t) = \mathrm{diag}(-1/U_s''(x_s(t)), s \in S_{BE}) \tag{22}$$

we have

$$\nabla^2 g(\lambda) = RF(t)R^T \tag{23}$$

and the update equation is given by:

$$\lambda(t+1) = \left[\lambda(t) - \gamma(t) \left(RF(t)R^T \right)^{-1} (\hat{c} - Rx) \right]^+ \tag{24}$$

where $x_s(\lambda(t))$ is the approximate of x_s^* in time t.

(18)

The abovementioned update equation necessitates matrix inversion in each iteration which imposes very large computational complexity to the system. One remedy to this problem is to consider the main diagonal elements of the Hessian and to ignore cross terms. Regarding this simplification, we only need to calculate the main diagonal elements of $RF(t)R^T$. By defining

$$E(t) = [E_{ij}(t)]_{L \times L}$$

$$E_{ij}(t) = \begin{cases} [RF(t)R^T]_{ii} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

The update equation using the simplified method can be rewritten as:

$$\lambda(t+1) = \left[\lambda(t) - \gamma(t) E^{-1}(t) (\hat{c} - Rx) \right]^+ \tag{26}$$

where $E(t)$ is a diagonal matrix and its inverse calculation poses very light computational load onto the system. It is worthnoting that (26) admits a very simple scalar form as:

$$\lambda_l(t+1) = \left[\lambda_l(t) - \frac{\gamma(t)}{[RF(t)R^T]_{ll}} \left(\hat{c}_l - \sum_{s \in S_{BE}(l)} x_s \right) \right]^+ \tag{27}$$

which in turns implies that the necessary mathematical operations using the simplified method only involve simple operations and admit very low computational complexity overhead.

(11) and (26) together form an iterative algorithm as the solution to the problem (13) and thereby problem (1). In this respect, optimal source rates for BE sources can be found while satisfying capacity constraints and preserving GS traffic requirements. Thus, the aforementioned algorithm can be used to control the flow of the BE sources in the NoC. The aforementioned iterative solution can be addressed in distributed scenarios. However, due to well-formed structure of the NoC, we focus on a centralized scheme; we consider a controller to be mounted in the NoC to implement this algorithm. The necessary requirement of such a controller is the ability to accommodate mathematical operations especially performing matrix inversion as in (11) and (26) and the allocation of few dedicated links to communicate flow control information to nodes with a light GS load. We summarize the proposed algorithm for Best Effort traffic as follows.

Algorithm 1: Congestion Control for BE Traffics in NoC

Initialization:

1. **Initialize $\hat{c}_l$ of all links.**
2. **Set link price vector to zero.**

Loop:

Do until $(\max|x_s(t+1)-x_s(t)| < Error)$

1. $\forall l \in L$ **: Compute new link prices:**

$$\lambda(t+1) = \left[\lambda(t) - \gamma(t)E^{-1}(t)(\hat{c} - Rx)\right]^{+}$$

where $\gamma(t)$ **can be selected as** $\gamma(t) = a/(b+t)$.

2. **Compute new BE source rates as follows**

$$x_s(t+1) = f(\sum_{l \in L(s)} \lambda_l(t+1))$$

where $f^{-1} = U_s^{'}(x_s)$

Output:

Communicate BE source rates to the corresponding nodes.

Stepsize has an important role on the convergence behavior of the update equation. There are several choices for stepsize, each one belonging to a predefined category and having certain advantages and drawbacks (see [14] and references herein).

In the family of iterative algorithms for distributed scenarios, stepsize is usually chosen to be a small enough constant so that to guarantee the convergence of the algorithm. Constant stepsize benefits from robustness against propagation delay and errors in estimation especially in asynchronous schemes[1]. However, it mainly suffers from slow convergence rate. On the contrary, time-varying stepsizes can be adapted to vary to achieve faster convergence rate. Due to well-formed structure of the NoC and its unified administration, in this paper we use a time-varying stepsize. Several categories for time-varying stepsize exists [14]. In this paper, we focus on a specific category known as *square-summable but not-summable* which satisfy the following conditions [13][14]:

$$\gamma(t) \geq 0 \qquad \forall t \tag{28}$$

$$\sum_{k=1}^{\infty} \gamma^2(t) < \infty \tag{29}$$

$$\sum_{k=1}^{\infty} \gamma(t) = \infty \tag{30}$$

One typical example is of the form $\gamma(t) = a/(b+t)$, where $a > 0$ and $b \geq 0$, which we will use in our simulations.

[1] Note that (24) presents a synchronous scheme, and may diverge in asynchronous cases, e.g. real world conditions with large delays, etc.

IV. Simulation Results

In this section we examine the proposed congestion control algorithm, listed above as Algorithm 1, for a typical NoC architecture. We have simulated a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. We assume packets traverse the network on a shortest path using a deadlock free XY routing. Each packet consists of 500 flits and each flit is 16 bit long.

In order to simulate our scheme, some nodes are considered to have a Guaranteed Service data (such as Multimedia, etc.) to be sent to a destination while other nodes, which maybe in the set of nodes with GS traffic, have a Best Effort traffic to be sent. As stated in section II, GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE traffics.

In our simulation we have chosen logarithmic utility functions. In this respect, for source s, we choose $U_s(x_s) = \log x_s$. Such a utility function satisfies fair conditions among sources and is said to be *Weighted Proportionally-Fair* which is an important property in economics [15]. Due to this property, such utility functions exhibit fair behavior across all nodes.

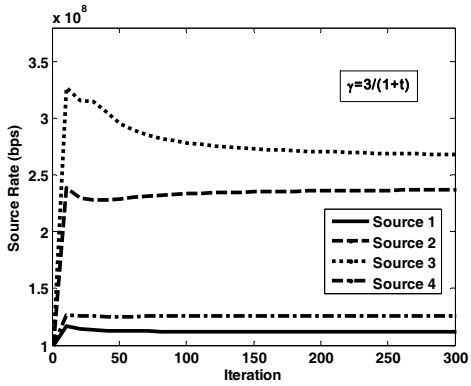
On of the most significant issues of our interest, is the convergence behavior of the source rates. We used two different scenarios for step-size; both of them are chosen to be *square-summable but not summable.* In this regard, step sizes are chosen as $\gamma = 3/(1+t)$ and $\gamma = 1/(1+t)$ which satisfy (28)-(30).

Variation of source rates for some nodes using aforementioned step sizes are shown in Fig. 1(a)-(b). Regarding Fig. 1(a), it's apparent that after about 80 iterations, all source rates will be in the vicinity of the steady state point of the algorithm. However, for the second case, Fig. 1(b) reveals that at least 100 iterations needed to have source rates in the vicinity of the optimal point. Comparing Fig. 1(a) and 1(b), we realize that the initial value of the step size, directly influences the rate of convergence.

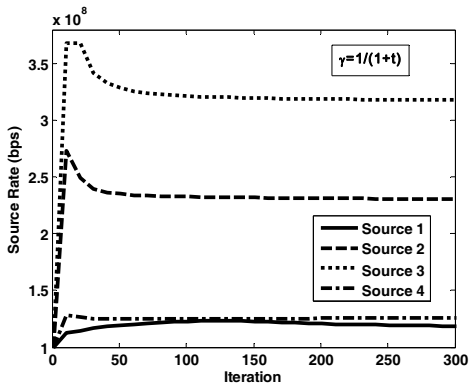
In order to have a better insight about the algorithm behavior, the relative error with respect to optimal rates which averaged over all sources, is also shown in Fig. 2. It is worthnoting that optimal source rates are obtained using CVX [16] which is a MATLAB-based software for solving disciplined convex optimization problems.

V. Conclusion and Future Work

In this paper we addressed the problem of congestion control for BE traffic in NoC systems. Congestion control was considered as the solution to the source rate utility maximization problem which was solved indirectly through its dual using Newton's method. This was led to an iterative algorithm which can be used to determine optimal BE source rates and thereby as a means to control the congestion of the NoC. The algorithm can be implemented by a controller which admits a light communication and communication overhead. Further investigation about convergence behavior of the algorithm and the effect of different utility functions on the



(a)



(b)

Figure 1. Source rates for (a) $\gamma = \frac{3}{1+t}$ and (b) $\gamma = \frac{1}{1+t}$.

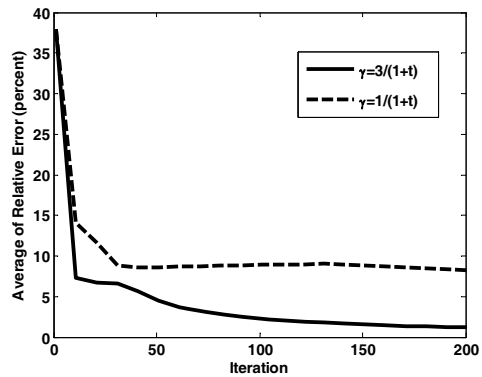


Figure 2. Average of relative error with respect to optimal solution for the three cases.

BE rates and fairness provision is the main directions of our future studies.

REFERENCES

[1] L. Benini, and G. De Micheli, “Network on Chips: A New SoC Paradigm”, IEEE Computer, pp. 70-78, January 2002.

[2] A. Jantsch, and H. Tenhunen, "Networks on Chip", Kluwer Academic Publishers, Boston, USA, January 2003.

[3] W. J. Dally, and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks”, Proceedings of the 38th Design Automation Conference, ACM/IEEE, Las Vegas, Nevada, USA, pp. 684-689, June 2001.

[4] B. Gebremichael, F.W. Vaandrager and M. Zhang, “Formal Models of Guaranteed and Best-Effort Services for Networks on Chip”, Technical Report ICIS-R05016, ICIS, Radboud University Nijmegen, March 2005.

[5] G. Almes et al. RFC2581: “TCP congestion control”. Technical report, Network Working Group, 1999.

[6] Y. Gu et al. “A predictive congestion control algorithm for high speed communication networks”. Proceedings of American Control Conference, 2001.

[7] C. Yang et al. “A taxonomy for congestion control algorithms in packet switching networks”. IEEE Network, 9, 1995.

[8] S. H. Low. “Optimization Flow Control, I: Basic Algorithm and Convergence”, IEEE/ACM Transactions on Networking, 7(6): 861-874, 1999.

[9] U. Ogras et al. “Prediction-based flow control for network-onchip traffics”. Proceedings of Design Automation Conference (DAC) , 2006.

[10] J. W. van den Brand, C. Ciordas, K. Goossens1 and T. Basten. “Congestion-Controlled Best-Effort Communication for Networks-on-Chip”. Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), April 2007.

[11] J. Hu et al. “DyAD - smart routing for networks-on-chip”. Proceedings of Design Automation Conference (DAC), 2004.

[12] S. Boyd and L. Vandenberghe, “Convex Optimization”. Cambridge, U.K, Cambridge Univ. Press, 2004.

[13] Dimitri P. Bertsekas, ”Nonlinear Programming”, Athena Scientific. 1999.

[14] S. Boyd, “Convex Optimization II Lecture Notes”, Stanford University, 2006.

[15] F. P. Kelly. “Charging and rate control for elastic transaction”. European Transactions on Telecommunications, 8:33{37, 1997.

[16] Michael Grant, Stephen Boyd and Yinyu Ye., “CVX (Ver. 1.0RC3): Matlab Software for Disciplined Convex Programming”, Download available at: http://www.stanford.edu/~boyd/cvx.

Paper 2

A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization

M. S. Talebi
F. Jafari
A.Khonsari
M. H. Yaghmaee

In the Proceedings of the International Conference on Computational Science and its Applications (ICCSA), pp. 398-409, Kuala Lumpur, Malaysia, August 2007.

A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization

Mohammad S. Talebi[1], Fahimeh Jafari[1,3], Ahmad Khonsari[2,1], and Mohammad H. Yaghmae[3]

[1] IPM, School of Computer, Tehran, Iran.,
[2] ECE Department, University of Tehran
[3] Ferdowsi University of Mashhad,
mstalebi@ipm.ir, jafari@ipm.ir, ak@ipm.ir, hyaghmae@ferdowsi.um.ac.ir

Abstract. Network-on-Chip (NoC) has been proposed as an attractive alternative to traditional dedicated busses in order to achieve modularity and high performance in the future System-on-Chip (SoC) designs. Recently, end-to-end congestion control has gained popularity in the design process of network-on-chip based SoCs. This paper addresses a congestion control scenario under traffic mixture which is comprised of Best Effort (BE) traffic or elastic flow and Guaranteed Service (GS) traffic or inelastic flow. We model the desired BE source rates as the solution to a rate-sum maximization problem which is constrained with link capacities while preserving GS traffic services requirements at the desired level. We proposed an iterative algorithm as the solution to the maximization problem which has the advantage of low complexity and fast convergence. The proposed algorithm may be implemented by a centralized controller with low computation and communication overhead.

1 Introduction

The *Systems-on-Chip* (SoC) was first designed as a tightly interconnected set of cores, where all components share the same system clock, and the communication between components is via shared-medium busses. With the advance of the semiconductor technology, the enormous number of transistors available on a single chip allows designers to integrate dozens of IP blocks together with large amounts of embedded memory. Such IPs consist of CPU or DSP cores, video stream processors, high-bandwidth I/O, routers, *etc*. As more and more cores are integrated into a single chip, it is becoming increasingly difficult to meet the design constraints while still using the old design methodologies for SoC designs. Shared-medium busses do not scale well, and do not fully utilize potentially available bandwidth. As the features sizes shrink, and the overall chip size relatively increases, interconnects start behaving as lossy transmission lines. Crosstalk, electromagnetic interference, and switching noise cause higher incidence of data errors. Line delays have become very long as compared to gate delays causing synchronization problems between cores. A significant amount of power is dissipated on long interconnects and in clocking network. This trend only

O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, LNCS 4707, pp. 398 – 409, 2007.

worsens as the clock frequencies increase and the features sizes decrease. Lowering the power supplies and designing smaller logic swing circuits decreases the overall power consumption at the cost of higher data errors.

One solution to these problems is to treat SoCs implemented using *micro-networks*, or *Networks on Chips* (NoCs). Networks have a much higher bandwidth due to multiple concurrent connections. They have regular structure, so the design of global wires can be fully optimized and as a result, their properties are more predictable. Regularity enables design modularity, which in turn provides a standard interface for easier component reuse and better interoperability. Overall performance and scalability increase since the networking resources are shared.

Networking model decouples the communication layers so that design and synthesis of each layer is simpler and can be done separately. In addition, decoupling enables easier management of power consumption and performance at the level of communicating cores. Generally, the concept of NoC which was introduced in [1][2], suggests that different modules would be connected by a simple network of shared links and routers. Examples of NoCs are Æthereal [3], Mango [4] and Xpipes [5]. NoCs provide communication services to IPs. Communication services with guarantees on throughput and latency enable predictable system design. Guarantees are given by reserving communication resources in the NoC (e.g. wires and buffers). Although necessary for hard real-time applications, this results in poor resource utilization for applications that require Variable-Bit Rate (VBR) communication. According to the nature of such kind of traffic, this is also called *Inelastic Flow*. Best Effort service (BE) is another kind of communication services which doesn't need any guarantees on latency and bandwidth. According to the nature of this kind of traffic, this is also called *Elastic Flow*. Elastic Flow or BE service can give high resource utilization by using unreserved or unused resources. However, BE traffic is prone to network congestion. Æthereal [3] and Mango [4] are examples of NoCs that provide both GS and BE services.

Networks with BE services should have a strategy to avoid congestion. The congestion control in NoCs is a novel problem for the resource constrained on-chip designs. During the past two decades, many strategies for congestion control have been proposed for off-chip networks [6, 7, 8]. Congestion control for on-chip networks is still a novel issue, however this problem has been investigated by several researchers [9]-[11]. In [9], a prediction-based flow control strategy for on-chip networks has been proposed where each router predicts future buffer fillings to detect future congestion problems. The buffer filling predictions are based on a router model. Dyad [10] solves congestion problem by switching from deterministic to adaptive routing when the NoC gets congested. In [11] the link utilization has been used as congestion measure and the controller determines the appropriate loads for the BE sources. All of the aforementioned work has dealt with this issue using the predictive control approach to overcome the congestion in the network. As the NoC architecture is similar to a regular data network, in this paper we have used an optimization approach over Best Effort source rates to control the flow.

The main purpose of this paper is to present a congestion control as the solution to a sum-rate maximization problem for choosing the rate of BE sources. We present an algorithm as the solution to the optimization problem and prove its convergence. To evaluate the performance of the proposed approach, we simulate the congestion

control algorithm under a NoC-based scenario. Similar to [10], we have used a controller to implement the proposed algorithm; however our approach is completely different from [10].

This paper is organized as follows. In section 2 we present the system model and formulate the underlying optimization problem for BE flow control. In section 3 we solve the optimization problem using an iterative algorithm and propose the solution as a centralized congestion control algorithm to be implemented as a controller. In section 4 we analyze the convergence behavior of the proposed algorithm and prove the underlying theorem of its convergence. In section 5 we present the simulation results. Finally, the section 6 concludes the paper and states some future work directions.

2 System Model

We consider a NoC with two dimensional mesh topology and wormhole routing. In wormhole networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in a pipeline fashion. The NoC architecture is assumed to be lossless, and packets traverse the network on a shortest path using a deadlock free XY routing.

We model the congestion control problem in NoC as the solution to an optimization problem. For more convenience, we turn the aforementioned NoC architecture into a mathematical model as in [8]. In this respect, we consider NoC as a network with a set of bidirectional links L and a set of sources S. A source consists of Processing Elements (PEs), routers and Input/Output ports. Each link $l \in L$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of c_l packets/sec. We denote the set of sources that share link l by $S(l)$. Similarly, the set of links that source s passes through, is denoted by $L(s)$. By definition, $l \in S(l)$ if and only if $s \in L(s)$ [8].

As previously stated, there are two types of traffic in a NoC: Guaranteed Service traffic (GS) or inelastic flow and Best Effort (BE) traffic or elastic flow. For notational convenience, we divide S into two parts, each one representing sources with the same traffic. In this respect, we denote the set of sources with BE and GS traffic by S_{BE} and S_{GS}, respectively. Each link l is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE sources.

Our objective is to choose source rates (PE loads) of BE traffics so that to maximize the sum of rates of all BE traffics. Hence the maximization problem can be formulated as:

$$\max_{x_s} \sum_{s \in S_{BE}} x_s \tag{1}$$

subject to:

$$\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{2}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{3}$$

where source rates, i.e. x_s, $s \in S$, are optimization variables.

The constraint Eq. (2) says the aggregate BE source rates passing thorough link l cannot exceed its free capacity, i.e. the portion of the link capacity which has not been allocated to GS sources. The abovementioned problem is in fact *constrained sum-rate maximization.* Such a problem, in general belongs to the class of *Utility-Maximization Problems* for which the utility function of all sources is considered to be *Identity Function*, i.e. $U_s(x_s) = x_s$. Although the general form of Eq. (1) with a general utility function has been investigated by the authors in another work [12], the approach of this paper to solve problem Eq. (1) is completely different from the previous work. In this paper we focus on the primal problem while in [12] the problem is solved via its dual function. For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in S_{GS}(l)} x_s \tag{4}$$

Hence, Eq. (2) can be rewritten as:

$$\sum_{s \in S_{BE}(l)} x_s \leq \hat{c}_l \qquad \forall l \in L \tag{5}$$

Although problem Eq. (1) is separable across sources, its constraints will remain coupled across the network. Due to coupled nature of such constrained problems, they have to be solved using centralized methods like interior point methods [13]-[15]. Such computations may pose a great overhead on the system. Instead of such methods, we seek to obtain the solution with simpler operations. One way is to use the subgradient method for constrained optimization problems [16] which will be briefly reviewed in the next section.

For notational convenience in solving the problem, we use matrix notation. In this respect, we define Routing matrix, i.e. $R = [R_{ls}]_{L \times S}$, as following:

$$R_{ls} = \begin{cases} 1 & \text{if } s \in S_{BE}(l) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We also define the source rate vector (for BE traffic) and link capacity vector as $x = (x_s,\ s \in S_{BE})$ and $\hat{c} = (\hat{c}_l,\ l \in L)$, respectively. Therefore problem Eq. (1) can be rewritten in the matrix form as follows:

$$\max_x \ \mathbf{1}^T x \tag{7}$$

subject to:

$$Rx \leq \hat{c} \tag{8}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{9}$$

where $\mathbf{1}$ is a vector with all one.

3 Congestion Control Algorithm

In this section, we will solve the sum-rate optimization problem Eq. (7) using subgradient method for constrained optimization problems [13][16] and present a flow control scheme for BE traffic – or elastic flows - in NoC systems to overcome the congestion. Convergence analysis of the algorithm is to be discussed in the next section.

The subgradient method for constrained optimization problems is very similar to *Poljak's Method* [17]. In this method, for a maximization problem like Eq. (1), the optimization variable vector will be adjusted in the direction to the gradient of the objective function. We briefly review this method in lemma 1 as follows.

Lemma 1. *Consider the constrained maximization problem,*

$$\max_{x} f(x) \tag{10}$$

subject to:

$$f_i(x) \geq 0, \quad i = 1..M \tag{11}$$

with the maximal x^* *and the sequence* $x(t)$ *as*

$$x(t+1) = x(t) + \gamma(t)u(x(t)) \tag{12}$$

where

$$u(x(t)) = \begin{cases} \nabla f(x(t)) & \text{if } x(t) \text{ satisfies (11)} \\ \nabla f_j(x(t)) & \exists j \text{ s.t } f_j(x(t)) < 0 \end{cases} \tag{13}$$

where $\gamma(t)$ *is a diminishing step-size rule [13]-[15]. If the* l_2 *-norm of the* $u(x(t))$ *is bounded (Lipschitz Continuity), i.e. there exist* G *such that*

$$\|u\|_2 \leq G \tag{14}$$

and the Euclidian distance of the initial point to the optimal point is bounded, i.e.

$$\left\|x(1) - x^*\right\| \leq D \tag{15}$$

then the sequence $\{x(t)\}_{t=1}^{\infty}$, *as* $t \to \infty$ *will converge to* x^*.

Proof: See [16].

In the sequel, we will solve the optimization problem Eq. (7) using subgradient method for constrained optimization problems as stated in Lemma 1. Regarding Eq. (12), we should calculate $u(x(t))$. According to Eq. (13), if $x(t)$ is feasible, i.e. $Rx \leq \hat{c}$, we have:

$$u = \nabla \mathbf{1}^T x = \nabla x^T \mathbf{1} = \mathbf{1} \tag{16}$$

otherwise at least one of the constraints should be violated. Assuming the corresponding constraint for link $l^{'}$ is violated, i.e. $\sum_{s \in S_{BE}(l^{'})} x_s > \hat{c}_{l^{'}}$. We can represent this constraint in matrix form as:

$$f_{l^{'}}(x) = \mathbf{e}_{l^{'}}^T(\hat{c} - Rx) < \mathbf{0} \tag{17}$$

where $\mathbf{e}_{l^{'}}$ is the $l^{'}$th unit vector of $\mathbb{R}^L$ space which is zero in all entries except the $l^{'}$th at which it is 1. Therefore, u is given by:

$$u = -\nabla \mathbf{e}_{l^{'}}^T(Rx - \hat{c}) = -R^T \mathbf{e}_{l^{'}} \tag{18}$$

Using Eq. (16) and Eq. (18), the update equation to solve problem Eq. (7) is given by:

$$x(t+1) = x(t) + \gamma(t)u(x(t)) \tag{19}$$

where $u(x(t))$ is given by:

$$u(t) = \begin{cases} \mathbf{1} & \sum_{s \in S_{BE}(l)} x_s(t) \le \hat{c}_l, \forall l \\ -R^T \mathbf{e}_{l^{'}} & \sum_{s \in S_{BE}(l^{'})} x_s(t) > \hat{c}_{l^{'}}, \exists l^{'} \end{cases} \tag{20}$$

Stepsize has an important role on the convergence behavior of the update equation. There are several choices for stepsize, each one belonging to a predefined category and having certain advantages and drawbacks (see [14] and references herein).

In the family of gradient algorithms, for distributed scenarios stepsize is usually chosen to be a small enough constant so that to guarantee the convergence of the algorithm. Constant stepsize is robust in the sense of convergence in time-varying conditions and asynchronous schemes[1]. However, it mainly suffers from slow convergence rate. On the contrary, time-varying stepsizes are defined in such a way to adapt to the error with the desired point, i.e. optimal point of the optimization problem, and hence benefit from much more faster convergence. However, they should be constrained to guarantee that the iterative algorithm will converge.

In our scheme, the algorithm is to be centralized in implementation, and thus we use a time-varying stepsize to take advantage of fast convergence. To this end, we choose $\gamma(t)$ as a time-varying stepsize, to be *square-summable but not summable* [14][16]. In this respect, $\gamma(t)$ satisfies

$$\gamma(t) \ge 0 \qquad \forall t \tag{21}$$

$$\sum_{k=1}^{\infty} \gamma^2(t) < \infty \tag{22}$$

[1] Note that Eq. (19) proposes a synchronous scheme, and may diverge in asynchronous ones, e.g. real world conditions with large delays, etc.

$$\sum_{k=1}^{\infty} \gamma(t) = \infty \tag{23}$$

One typical example that satisfies Eq. (21)-(23), is $\gamma(t) = a/(b+t)$, where $a > 0$ and $b \geq 0$, which we have used in this paper.

Eq. (19) and Eq. (20) together propose an iterative algorithm as the solution to problem Eq. (7). In this respect, optimal source rates for BE sources can be found while satisfying capacity constraints and preserving GS traffic requirements. Thus, the aforementioned algorithm can be employed to control the congestion of the BE traffic in the NoC. The above iterative algorithm is decentralized in the nature and can be addressed in distributed scenarios. However, due to well-formed structure of the NoC, we focus on a centralized scheme; a simple controller can be mounted in the NoC to implement this algorithm. The necessary requirement of such a controller is the ability to accommodate simple mathematical operations as in Eq. (19) and Eq. (20) and the allocation of few wires to communicate congestion control information to nodes with a light GS load. Algorithmic realization of the proposed Congestion-Controller for BE traffic is listed as Algorithm 1.

Algorithm 1. Congestion Control for BE Traffics in NoC

Initialization:

1. **Initialize $\hat{c}_l$ of all links.**
2. **Set source rate vector to zero.**

Loop:

Do until $(\max |x_s(t+1) - x_s(t)| < Error)$

1. $\forall s \in S$ **: Compute new source rate:**

$$x(t+1) = x(t) + \gamma(t) u(x(t))$$

where $\gamma(t)$ **can be selected as** $\gamma(t) = a/(b+t)$ and

$$u(t) = \begin{cases} \mathbf{1} & \sum_{s \in S_{BE}(l)} x_s(t) \leq \hat{c}_l, \ \forall l \\ -R^T \mathbf{e}_{l'} & \sum_{s \in S_{BE}(l')} x_s(t) > \hat{c}_{l'} \end{cases}$$

Output:

Communicate BE source rates to the corresponding nodes.

4 Convergence Analysis

In this section, we investigate the convergence analysis of the proposed algorithm using a time-varying stepsize in Eq. (19). As stated in the previous section, in this paper the stepsize is selected to be *square-summable but not summable* [16].

Theorem 1. *The iterative congestion control scheme proposed by Eq. (19) and Eq. (20) with a time-varying stepsize which satisfies Eq. (21)-(23), will converge to the optimal point of problem Eq. (1).*

***Proof*:** By lemma 1, it is clear that if its assumptions hold, the proof of Theorem is done. First, $u(x(t))$ should admit an upper bound in l_2-norm. In doing so, it suffices to show that its gradient is upper bounded in l_2-norm. Considering Eq. (16) and (18), we have

$$\begin{aligned} \|u\|_2 &\le \max\{\|\mathbf{1}\|_2, \left\|-R^T \mathbf{e}_{l'}\right\|_2\} \\ &= S \end{aligned} \tag{24}$$

hence u in l_2-norm is bounded with at least S.

In the next step, we show that the Euclidian distance of the initial point to the optimal point is bounded at least with D, i.e.

$$\exists D > 0 \qquad \text{s.t.} \qquad \left\|x(1) - x^*\right\|_2 \le D$$

According to Eq. (3), we have $x_s > 0, \ \forall s \in S_{BE}$. On the other hand, optimal source rates are bounded at most with maximum value of link capacities, i.e.

$$\max_s x_s^* \le \max_l \hat{c}_l \le \max_l c_l \tag{25}$$

therefore,

$$\begin{aligned} \left\|x(1) - x^*\right\|_2 &\le \left\|\max x^* - \min x(1)\right\|_2 \\ &= \left\|\max_l c_l - 0\right\|_2 \\ &= L c_{l\max} \end{aligned} \tag{26}$$

and hence the initial Euclidian distance is bounded and Eq. (26) with Eq. (24) completes the proof.

5 Simulation Results

In this section we examine the proposed congestion control algorithm, listed above as Algorithm 1, for a typical NoC architecture. We have simulated a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. In our scenario, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bit long.

In order to simulate our scheme, some nodes are considered to have a Guaranteed Service data (such as Multimedia, etc.) to be sent to a destination while other nodes,

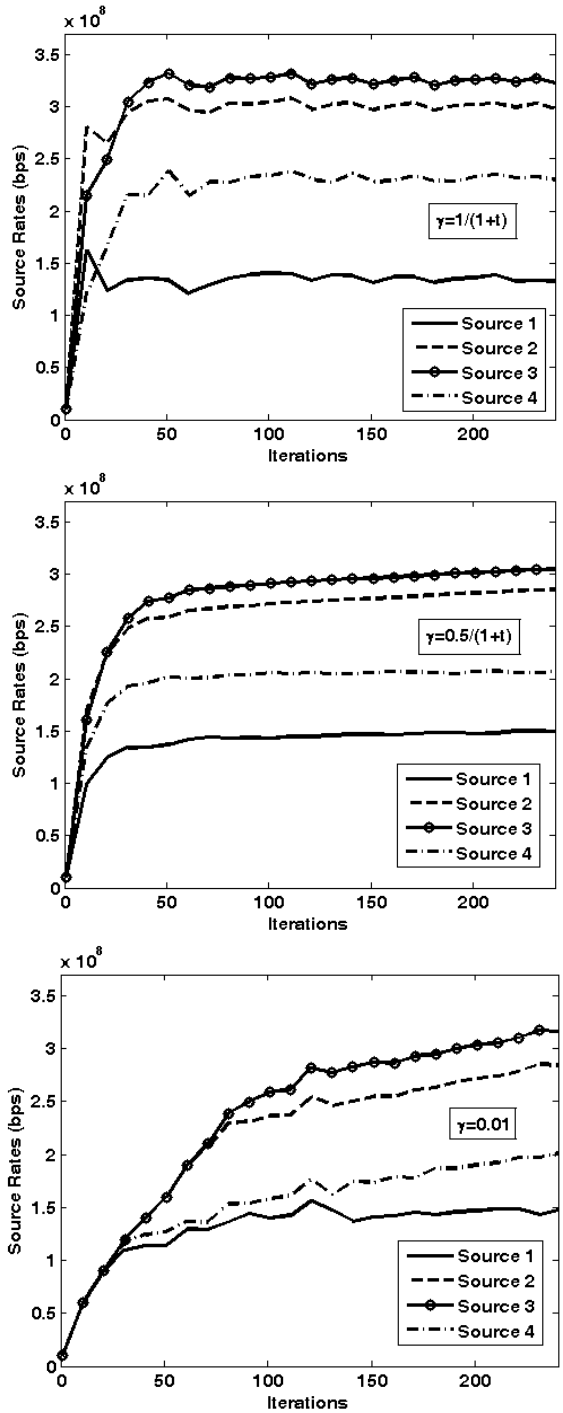


Fig. 1. Source rates for (a) $\gamma = \dfrac{1}{1+t}$, (b) $\gamma = \dfrac{0.5}{1+t}$ and (c) $\gamma = 0.01$

which maybe in the set of nodes with GS traffic, have a Best Effort traffic to be sent. As stated in section 2, GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE traffics.

One of the most significant issues of our interest is the convergence behavior of the source rates. We used three different scenarios for step-size; two of them are chosen to be *square-summable but not summable* and the third is set to be constant. For the first two cases, stepsizes are chosen as $\gamma = 1/(1+t)$ and $\gamma = 0.5/(1+t)$ which satisfy Eq. (21)-(23). For the constant case, stepsize is set to be $\gamma = 0.01$. The first and second cases will be comparable with the constant stepsize after about 99 and 49 iterations, respectively.

Variation of source rates for some nodes using aforementioned stepsizes are shown in Fig. 1(a)-(c). Regarding Fig. 1(a), it's apparent that after about 50 iterations, all source rates will be in the vicinity of the steady state point of the algorithm. However, for the second case, Fig. 1(b) reveals that at least 80 iterations needed to have source rates in the vicinity of the optimal point. For the third case, the rate of convergence is even less and at least 150 iterations are needed to fall within the neighborhood of the steady state point of the algorithm. It is clear that compared to the *square-summable but not summable* stepsizes, constant stepsize has much slower rate of convergence. Comparing Fig. 1(a) and 1(b), we realize that the initial value of the stepsize, directly influences the rate of convergence.

In order to have a better insight about the algorithm behavior, the relative error with respect to optimal rates which averaged over all sources, is also shown in Fig. 2. Optimal source rates are obtained using CVX [18] which is MATLAB-based software for solving disciplined convex optimization problems. This figure reveals that *square-summable but not summable* stepsizes can lead to lower relative error in

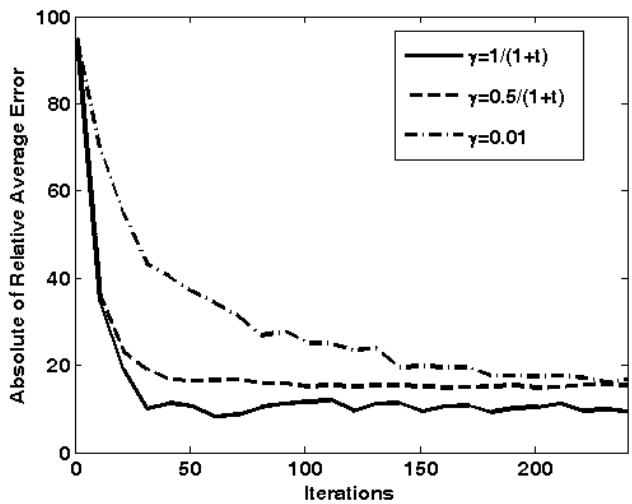


Fig. 2. Average of relative error with respect to optimal solution for the three cases

average with regard to constant stepsize. The faster rate of convergence of the first two cases than the third, can also be verified and it is apparent that the first case slightly acts better from the second in terms of averaged relative error.

6 Conclusion and Future Work

In this paper we addressed the problem of congestion control for BE traffic in NoC systems. Congestion control was modeled as the solution to the sum-rate maximization problem which was solved using subgradient method for constrained optimization problems. This was led to an iterative algorithm which determine optimal BE source rates. We have also studied the realization of the algorithm as a centralized congestion controller and presented a theorem to prove the convergence of the proposed congestion control scheme. Simulation results confirm that the proposed algorithm converges very fast and the computational overhead of the congestion control algorithm is small. Fast convergence of the algorithm also justifies that the delay incurred by the algorithm is very small. Further investigation about the effect of other utility functions on the BE rates and fairness provision is the main direction of our future studies.

References

1. Guerrier, P., Greiner, A.: A Generic Architecture for On-Chip Packet-Switched Interconnections. In: Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE) (2000)
2. Dally, W.J., Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks. In: Proc. DAC 2001 (2001)
3. Goossens, K., et al.: The Æthereal network on chip: Concepts, architectures, and implementations. IEEE Design and Test of Computers 22(5) (2005)
4. Bjerregaard, T., et al.: A router architecture for connection oriented service guarantees in the MANGO clockless Network-on-Chip. In: Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE) (2005)
5. Bertozzi, D., et al.: Xpipes: A network-on-chip architecture for gigascale systems-on-chip. IEEE Circuits and Systems Magazine (2004)
6. Kelly, F.P., Maulloo, A., Tan, D.: Rate control for communication networks: Shadow prices, proportional fairness, and stability. J. Oper. Res. Soc. 49(3), 237–252 (1998)
7. Yang, C., et al.: A taxonomy for congestion control algorithms in packet switching networks. IEEE Network 9 (1995)
8. Low, S.H., Lapsley, D.E.: Optimization Flow Control, I: Basic Algorithm and Convergence. IEEE/ACM Transactions on Networking 7(6), 861–874 (1999)
9. Ogras, U., et al.: Prediction-based flow control for network-onchip traffic. In: Proc. DAC (2006)
10. Hu, J., et al.: DyAD - smart routing for networks-on-chip. In: Proc. DAC (2004)
11. van den Brand, J.W., Ciordas, C., Goossens, K., Basten, T.: Congestion- Controlled Best-Effort Communication for Networks-on-Chip. In: Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE) (April 2007)

12. Talebi, M.S., Jafari, F., Khonsari, A.: Utility-Based Congestion Control for Best Effort Traffic in Network-on-Chip Architecture, (Submitted to MASCOTS 2007) (2007)
13. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge Univ. Press, Cambridge, U.K (2004)
14. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1999)
15. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and distributed computation. Prentice-Hall, Englewood Cliffs (1989)
16. Boyd, S.: Convex Optimization II Lecture Notes. Stanford University (2006)
17. Poljak, B.T.: A General Method of Solving Extremum Problems. Soviet Math Doklady 8(3), 593–597 (1967)
18. Grant, M., Boyd, S., Ye, Y.: CVX (Ver. 1.0RC3): Matlab Software for Disciplined Convex Programming, Download available at: http://www.stanford.edu/ boyd/cvx

Paper 3

Proportionally-Fair Best Effort Flow Control in Network-on-Chip Architectures

M. S. Talebi
F. Jafari
A.Khonsari
M. H. Yaghmaee

In the Proceedings of the International Workshop on Performance Modeling, Evaluation, and Optimization of Ubiquitous Computing and Networked Systems (PMEO UCNS), in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Miami, Florida, USA, April 2008.

Proportionally-Fair Best Effort Flow Control in Network-on-Chip Architectures

Mohammad S. Talebi[1], Fahimeh Jafari[1,2], Ahmad Khonsari[3,1], and Mohammad H. Yaghmaee[2]
[1] *IPM, School of Computer, Tehran, Iran*
[2] *Ferdowsi University of Mashhad, Mashahhad, Iran*
[3] *ECE Department, University of Tehran,Tehran, Iran*
mstalebi@ipm.ir, jafari@ipm.ir, ak@ipm.ir, hyaghmae@ferdowsi.um.ac.ir

Abstract

The research community has recently witnessed the emergence of Multi-Processor System on Chip (MPSoC) platforms consisting of a large set of embedded processors. Particularly, Interconnect networks methodology based on Network-on-Chip (NoC) in MP-SoC design is imminent to achieve high performance potential. More importantly, many well established schemes of networking and distributed systems inspire NoC design methodologies. Employing end-to-end congestion control is becoming more imminent in the design process of NoCs. This paper presents a centralized congestion scheme in the presence of both elastic and streaming flow traffic mixture. In this paper, we model the desired Best Effort (BE) source rates as the solution to a utility maximization problem which is constrained with link capacities while preserving Guaranteed Service (GS) traffics services requirements at the desired level. We proposed an iterative algorithm as the solution to the maximization problem which has the benefit of low complexity and fast convergence. The proposed algorithm may be implemented by a centralized controller with low computation and communication overhead

1. Introduction

The high level of system integration characterizing Multi-Processor Systems-on-Chip (MPSoCs) is raising the scalability issue for communication architectures. Towards this direction, traditional system interconnects based on shared busses are evolving both from the protocol and the topology viewpoint. Advanced bus protocols acts in favor of better exploitation of available bandwidth, while more parallel topologies are instead being introduced in order to provide more bandwidth [1].

In the long run, many researchers and SoC designers agree on the fact that this trend approaches the Network-on-Chip (NoC) as a solution to the lack of SoCs' Scalability [2].

A NoC system fundamentally consists of three components: switches, Network Interfaces (NIs) and links. The switches can be arbitrarily connected to each other and to NIs, based on a specified topology. They are responsible for routing, switching and flow control logic, as well as error control handling. NIs are responsible for packetization/depacketization and implement the service levels associated with each transaction.

Recently, Quality-of-Service (QoS) provisioning in NoC's environment has attracted many researchers and currently it is the focus of many literatures in NoC research community. NoCs are expected to serve as multimedia servers and are required not only to carry Elastic Flows, i.e. BE traffic, but also Inelastic Flows, i.e. GS traffic which requires tight performance constraints such as necessary bandwidth and maximum delay boundaries.

It's obvious that a network with data services needs some mechanisms to avoid congestion. Congestion Control in data networks is known as a widely-studied issue over the past two decades. However, it is still a novel problem in NoCs and to the best of our knowledge only few works has been carried out in this field. Congestion control, or equivalently, flow control in NoCs mainly focuses on the resource constrained on-chip designs, with the aim of minimizing the network cost or maximizing network utility while maintaining the required Quality-of-Service (QoS).

2. Related Works

Flow control for data networks is a widely-studied issue [3]-[6]. A wide variety of flow control mechanisms in data network belongs to the class of End-to-End control schemes, like TCP/IP, which is mainly based on the window-based scheme. In this methods, routers and intermediate nodes avoid the network from becoming congested by means of packet dropping deterministically (as in DropTail) or

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

randomly (as in RED). Therefore, sent packets are subject to loss and the network must aim to providing an acknowledgement mechanism. On the other On-chip networks pose different challenges. The reliability of on-chip wires and more effective link-level flow-control allows NoCs to be loss-less. Therefore, there is no need to utilize acknowledgment mechanism and we face to slightly different concept of flow control.

So far, several works have focused on this issue for NoC systems. In [7], a prediction-based flow-control strategy for on-chip networks is proposed in which each router predicts the buffer occupancy to sense congestion. This scheme controls the packet injection rate and regulates the number of packets in the network. In [8] link utilization is used as a congestion measure and a Model Prediction-Based Controller (MPC), determines the source rates. Dyad [9] controls the congestion by using adaptive routing when the NoC faces congestion.

In this paper, we focus on the flow control for BE traffic as the solution to a utility-based optimization problem. To the best of our knowledge, none of the aforementioned works have dealt with the flow control problem through utility optimization approach. In our seminal work [10], we have modeled desired BE source rates as the solution to a utility-based optimization problem with general form utility function and aimed at the issue with solving the proposed problem using Newton method. In [11], we also have considered this issue via sum-rate optimization problem and used a different approach to solve the problem. This paper we address the performance analysis of our seminal work [10] with a special utility function which satisfies Proportional Fairness feature and solve the flow control problem using a different approach which leads to low complexity flow control algorithm for BE traffic in NoCs.

This paper is organized as follows. In Section 3 we present the system model and formulate the underlying optimization problem for BE flow control. In section 4 we proceed to the proposed algorithm and discuss about some remarks. In section 5 we solve the optimization problem using an iterative algorithm over its dual and analyze the convergence behavior of it and present the underlying theorem of its convergence. Section 6 presents the simulation results. Finally, the section 7 concludes the paper and states some future work directions.

3. System Model and Flow Control Problem

We consider a NoC architecture which is based on a two dimensional mesh topology and wormhole routing. In wormhole networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in a pipeline fashion. A hop-to-hop credit mechanism assures that a flit is transmitted only when the receiving port has free space in its input buffer. We also assume that the NoC architecture is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing [2].

We model the flow control in NoC as the solution to an optimization problem. For the sake of convenience, we turn the aforementioned NoC architecture into a mathematically modeled network, as in [12]. In this respect, we consider NoC as a network with a set of bidirectional links L and a set of sources S. A source consists of Processing Elements (PEs), routers and Input/Output ports. Each link $l \in L$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of c_l packets/sec. We denote the set of sources that share link l by $S(l)$. Similarly, the set of links that source s passes through, is denoted by $L(s)$. By definition, $l \in S(l)$ if and only if $s \in L(s)$.

As discussed in section I, there are two types of traffic in a NoC: Guaranteed Service (GS) and Best Effort (BE) traffic. For notational convenience, we divide S into two parts, each one representing sources with the same kind of traffic. In this respect, we denote the set of sources with BE and GS traffic by S_{BE} and S_{GS}, respectively. Each link l is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and BE sources benefit from the remainder.

Our objective is to choose source rates with BE traffic so that to maximize the weighted sum of the logarithm of the BE source rates. Hence the maximization problem can be formulated as [12]:

$$\max_{x_s} \sum_{s \in S_{BE}} a_s \log x_s \tag{1}$$

subject to:

$$\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{2}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{3}$$

Optimization variables are BE source rates, i.e. $(x_s,\ s \in S_{BE})$ and a_s is the weight for source s. We later on discuss how such a weight determines the priority of source s in resource allocation. The constraint (2) states that the sum of BE source rates passing thorough link l cannot exceed its free

capacity, i.e. the portion of c_l which has not been allocated to GS traffic.

In General, problem (1) belongs to the class of utility-based optimization problems, for which the utility function, U_s, is assumed to be logarithmic, i.e. $U_s(x_s) = a_s \log x_s$. Such utility functions, are positive, concave and strictly increasing, as logarithmic function does. There are many choices for utility function, other than logarithmic, with specific features and behavior. We discuss in section V, that logarithmic utility function have nice properties in terms of economic terminology, known as proportional fairness [3].

It is worth to mention that despite the restriction of ourselves to a specific utility function, our work can be easily generalized to arbitrary utility functions, as in our seminal work [10].

With the model above, problem (1) is a convex optimization problem with linear constraints. Hence it admits a unique maximizer [13][14], i.e. there exists an optimal source rate vector, $x^* = (x_s^*, s \in S_{BE})$ that maximizes the objective of problem (1) while satisfying capacity constraints.

Problem (1) is coupled across the network through its constraints. Such a coupled nature, necessitate usage of centralized methods like Interior Point method which poses great computational overhead onto the system [13][14] and hence is of little interest.

In contrast, there are several low-complexity and distributive methods to solve unconstrained problems. Hence, one way to reduce the computational complexity is to transform the constrained optimization problem into its *Dual*, which can be defined to be unconstrained. According to the Duality Theory [13][14], each convex optimization (maximization) problem has a dual, whose optimal solution, called *Dual-Optimal*, leads to best bound (upper bound) of the optimal solution of the main problem. In this respect, the main problem is retroactively called *Primal Problem*. As the dual problem can be defined in such a way to be unconstrained, solving the dual is much simpler than the primal.

For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in S_{GS}(l)} x_s \tag{4}$$

We also define the source rate vector (for BE traffic) and link capacity vector as $x = (x_s, s \in S_{BE})$ and $\hat{c} = (\hat{c}_l, l \in L)$, respectively. To avoid confusing with summations indices, we define Routing matrix, i.e. $R = [R_{ls}]_{L \times S}$, as following:

$$R_{ls} = \begin{cases} 1 & \text{if } s \in S_{BE}(l) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Using the abovementioned definitions, problem (1) can be rewritten as:

$$\max_{x_s} \sum_s a_s \log x_s \tag{6}$$

subject to:

$$Rx \le \hat{c} \tag{7}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{8}$$

4. Optimal Flow Control Algorithm

In this section, we present a centralized flow control algorithm for BE traffic in NoC systems which controls the BE source rates in favor of problem (1). Later, in section V, we show that solving problem (1) leads to the proposed algorithm, and therefore the algorithm is an iterative optimal solution to it. The proposed flow control algorithm is listed below as algorithm 1.

In the sequel, we make some worth-mentioning remarks. Performance analysis of the algorithm is to be discussed in the next section.

Remarks:

1. Considering algorithm 1 as a centralized algorithm, we consider a simple controller that can be mounted in the NoC, whether as a separate hardware module or a part of the operating system, which is responsible for running of the algorithm. From computational aspect, such a controller must have the ability of carrying out simple mathematical operations, as in Algorithm 1. Another necessary requirement of the controller, as Output section of the algorithm 1 suggests, is some links e.g. a control bus, to communicate the algorithm output to the BE sources.

Although Algorithm 1 is centralized, it can be easily casted into a distributive one upon introducing low communication overheads. Thus it can be addressed in decentralized scenarios, too. However, due to well-formed structure of NoC Systems, such a centralized algorithm suits for the system and thereafter we only focus on the centralized scheme.

2. The proposed flow control algorithm is very similar to End-to-End congestion control schemes in data networks, also known as TCP which are widely used to control BE data flow in the internet. End-to-End schemes use window-based method, i.e. each

Algorithm 1: Flow Control for BE in NoC

Initialization:

1. **Initialize c_l of all links.**
2. **Set link shadow price vector to zero.**
3. **Set the ε as the stopping criteria.**

Loop:

Do until $(\max|x_s(t+1)-x_s(t)|<\varepsilon)$

1. $\forall l \in L$ **: Compute new link prices:**

$$\lambda_l(t+1)=\left[\lambda_l(t) -\gamma\left(c_l(t)-\sum_{s\in S_{GS}(l)}x_s(t)-\sum_{s\in S_{BE}(l)}x_s(\lambda(t))\right)\right]^+$$

2. **Compute new BE source rates as follows**

$$x_s(t+1)=\frac{a_s}{\sum_l R_{ls}\lambda_l(t+1)}$$

Output:

Communicate BE source rates to the corresponding nodes.

source maintains a window of packets which are transmitted, but not acknowledged. Because the packets in data networks may be lost due to dropping at the routers or link failure, destination should acknowledge the ordered receipt of each packet in the current window. Each source changes its window size in response to congestion signals, i.e. negative acknowledges or duplicates ones, and thereby avoids the network to face congestion. Roughly, the source rate in each round trip (i.e. the way from source to destination and back to the source for acknowledgment), is the ratio of window size to the round trip time (i.e. duration of the trip).

Although flow control in TCP is carried out by means of window updates, however we can derive the corresponding rate updates, too. The proposed flow control algorithm is very similar to rate update in TCP scheme. Such a similarity stems from the similarity in the underlying flow control problem in both schemes. However, it is worth noting that unlike TCP, in algorithm 1 we have not considered any window based transmission and acknowledgement mechanism. This is due to the fact that NoC architecture is lossless, as previously stated in section III, and hence all packets will be delivered successfully and no acknowledgment is needed.

5. Performance Analysis: Optimal Solution and Convergence Analysis

In this section, we discuss that solving problem (1) through its Dual, leads to Algorithm 1. Towards this end, we first obtain the Dual of problem (1) and then solve it using Gradient Projection Method [14][15] and derive the abovementioned flow control algorithm. Then, we focus on the convergence behavior and other aspects of the proposed algorithm.

5.1. Dual Problem

In this part, we will obtain the dual of problem (1). Using the standard optimization methods [12], the Lagrangian of the problem (1) can be written as:

$$L(x,\lambda)=\sum_s a_s\log x_s-\sum_l\lambda_l(\sum_s R_{ls}x_s-\hat{c}_l) \qquad (9)$$

where $\lambda_l>0$ is the Lagrange Multiplier associated with constraint (2) for link l. Usually, λ_l is called *shadow price* [12] for the economic interpretation of its role in solving the primal problem through dual.

Regarding the Lagrangian of problem (1), the dual function is defined as [13]:

$$g(\lambda)=\sup_{x_s}L(x,\lambda) \qquad (10)$$

where λ is the vector of positive Lagrange multipliers. Thus the dual function is given by:

$$\begin{aligned} g(\lambda)&=\max_{x_s}\sum_s a_s\log x_s-\sum_l\lambda_l(\sum_s R_{ls}x_s-\hat{c}_l)\\ &=\max_{x_s}\sum_s\left(a_s\log x_s-x_s\sum_l R_{ls}\lambda_l\right)+\sum_l\lambda_l\hat{c}_l \end{aligned} \qquad (11)$$

By Karush-Kuhn-Tucker (KKT) Theorem [13], we can obtain optimal source rates, i.e. $x^*=(x_s^*, s\in S_{BE})$. Duality theory states that when the primal problem is convex, strong duality holds and thereby the duality gap is zero [13]. In this respect, the optimal source rate vector, x^*, corresponds to the optimal Lagrange multiplier vector, λ^* [13]. In other words, if x is a feasible point of the primal problem, which is primal-optimal the corresponding λ will be dual-optimal and vice versa. Therefore, at optimality we have

$$\nabla_x L(x,\lambda)\Big|_{(x^*,\lambda^*)}=\mathbf{0} \qquad (12)$$

where $\mathbf{0}$ is a vector with all zero. By taking the derivative of (9) with respect to x, we have

$$\frac{\partial L}{\partial x_s}\Big|_{(x_s^*,\lambda^*)} = \frac{a_s}{x_s^*} - \sum_l R_{ls}\lambda_l^* = 0 \tag{13}$$

$$x_s^* = \frac{a_s}{\sum_l R_{ls}\lambda_l^*} \tag{14}$$

Substituting x_s^* into (11) yields

$$g(\lambda) = \sum_s \left(a_s(\log a_s - 1) - a_s \log(\sum_l R_{ls}\lambda_l) \right) + \sum_l \lambda_l \hat{c}_l \tag{15}$$

The dual problem is defined as [13]:

$$\min_{\lambda \geq 0} g(\lambda)$$

therefore, we have

$$\min_{\lambda \geq 0} \sum_s \left(a_s(\log a_s - 1) - a_s \log(\sum_l R_{ls}\lambda_l) \right) + \sum_l \lambda_l \hat{c}_l \tag{16}$$

It is proven that the dual is always convex regardless of convexity or non-convexity of the primal problem [13]. Moreover, it is apparent from (16) that, by ignoring the mild condition on the positivity of λ, the dual problem is unconstrained. As dual problem is convex, it admits a unique optimal, i.e. a unique minimizer, which can be obtained using iterative algorithms. As the dual problem is unconstrained; solving (16) using iterative methods is much simpler than the primal.

5.2. Solving The Dual Problem

In this part, we will solve the dual problem using Projected Gradient Method [13] and derive algorithm 1.

The Projected Gradient Method adjusts shadow prices, i.e. Lagrange multiplier vector, in opposite direction to the gradient of the dual function, i.e. $\nabla g(\lambda)$, as follows:

$$\lambda(t+1) = \left[\lambda(t) - \gamma \nabla g(\lambda(t)) \right]^+ \tag{17}$$

where $\gamma > 0$ is a constant stepsize, and $[x]^+ \triangleq \max\{x, 0\}$. Since the objective of problem (1) is strictly concave, $g(\lambda)$ is continuously differentiable [13], hence $\nabla g(\lambda)$ exists. Using (15), the l-th element of the gradient vector is given by:

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \frac{\partial}{\partial \lambda_l} \left[\sum_s \left(a_s(1 - \log a_s - \log(\sum_l R_{ls}\lambda_l)) \right) + \sum_l \lambda_l \hat{c}_l \right] \tag{18}$$

Therefore,

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \hat{c}_l - \sum_s \frac{R_{ls} a_s}{\sum_l R_{ls}\lambda_l} \tag{19}$$

Regarding (14), (19) can be rewritten as:

$$\begin{aligned} \frac{\partial g(\lambda)}{\partial \lambda_l} &= \hat{c}_l - \sum_s R_{ls} x_s(\lambda) \\ &= \hat{c}_l - \sum_{s \in S(l)} x_s(\lambda) \end{aligned} \tag{20}$$

and the update equation is given by:

$$\lambda_l(t+1) = \left[\lambda_l(t) - \gamma \left(\hat{c}_l - \sum_{s \in S_{BE}(l)} x_s(\lambda(t)) \right) \right]^+ \tag{21}$$

where $\lambda(t+1) = (\lambda_l(t+1),\ l \in L)$ and $x_s(\lambda(t))$ is the approximate of x_s^* in time t. (14) and (21) together forms the proposed algorithm. Therefore, algorithm 1 is the iterative solution to problem (1).

5.3. Convergence Analysis

In this part, we investigate the convergence behavior of the proposed algorithm. As stepsize has an important role in the convergence behavior of the update equation, we mainly focus on the effect of stepsize. The conditions under which Algorithm 1 converges and performance analysis of the algorithm will be obtained with respect to the choice of stepsize.

There are several choices for stepsize, each one belonging to a predefined category and having certain advantages and drawbacks (see [16] and references herein). In the family of gradient algorithm for distributed scenarios, stepsize is usually chosen to be a small enough constant so that to guarantee the convergence of the algorithm. Constant stepsize is robust in the sense of convergence in time-varying conditions and asynchronous schemes. However, it usually has slower convergence rate than time-varying ones. Due to its simplicity and robustness, in this paper we have used a constant step-size.

Before proceeding to the theorem, we first present the fundamental lemma for the gradient optimization algorithms.

Lemma 1 [14]: *Consider the unconstrained minimization problem,*

$\min_x f(x)$

with the minimal x^*. *If* $\nabla f(x)$ *has Lipschitz Continuity property, i.e. there exist* L *such that*

$$\left|\nabla f(x_1) - \nabla f(x_2)\right| \le L\left\|x_1 - x_2\right\|_2 \quad (22)$$

then the sequence $x(t)$ *defined as*

$$x(t+1) = x(t) - \gamma \nabla f(x(t))$$

converges to the neighborhood of x^* *provided that*

$$\varepsilon \le \gamma \le \frac{2-\varepsilon}{L} \quad (23)$$

for some $\varepsilon > 0$,

Proof: *See [14].*

The following theorem, determines the condition on the stepsize, under which the Algorithm 1 converges to the neighborhood of the optimal of the problem (16) and thereby that of problem (1).

Theorem 1: *The iterative flow control scheme proposed by (14) and (21) converges to a neighborhood of the optimal point of the primal problem (1) provided that*

$$0 < \gamma \le \frac{2\underline{a}}{\overline{c}^2 \overline{L}\overline{S}} \quad (24)$$

where $\overline{S}$ *is the length of the longest path used by the sources,* $\overline{L}$ *is the number of sources sharing the most congested link,* $\underline{a}$ *is the minimum weight of sources and* $\overline{c}$ *is the upper bound on link capacities.*

Proof: Omitted due to space limit.

5.4. Proportional Fairness

Utility function directly influences the policy by which system resources, i.e. bandwidth, are shared among the competing sources. In this respect, in terms of economics terminology, utility function controls the *fairness* among users or sources. Several fairness criteria have been defined in the economics which are applicable to problem (1). Among them are *Max-Min Fairness* and *Proportional Fairness* [3]. In a system with Max-Min fairness, the resources are mainly shared in favor of weak users while in system with Proportional Fairness the resources are shared in proportion to the resource usage of each source. In the latter case, given an optimal source rate allocation $x^* = \left(x_s^*,\ s \in S\right)$ satisfying Proportional Fairness, with any other feasible source rate, say $x = \left(x_s,\ s \in S\right)$, the total proportional net benefit gained by the new source rates is decreased [3], i.e.:

$$\sum_s \frac{x_s - x_s^*}{x_s^*} \le 0 \quad (25)$$

It is proven, systems with proportional fairness that satisfies (25), must have logarithmic utility functions [3], i.e.

$$U_s(x_s) = \log x_s \quad (26)$$

Thus the proposed flow control algorithm, with equal weight factors will be proportionally fair. It is worth to note that the case of heterogeneous weight factors corresponds to another implementation of fairness, the so-called Weighted Proportionally Fair, for which (25) turns to be

$$\sum_s a_s \left(\frac{x_s - x_s^*}{x_s^*}\right) \le 0 \quad (27)$$

In the sequel, we briefly discuss about the effect of weight factors. As previously stated, a_s is the weight for source s in the optimization problem which controls the priority of source s in resource sharing. To gain more insights on the role of a_s in the flow control, we consider a simple network with a single bottleneck link, say link l'. Since all other links doesn't saturate, we have $\lambda_l = 0,\ \ l \ne l'$. Using (2) and (14) we have:

$$x_s = \frac{a_s}{\sum_{l \in L(s)} \lambda_l} = \frac{a_s}{\lambda_{l'}}, \qquad s \in S(l') \quad (28)$$

$$\frac{x_i}{a_i} = \frac{x_j}{a_j} = \ldots = \frac{x_n}{a_n} = \frac{1}{\lambda_{l'}} \quad i,j,n \in S(l') \quad (29)$$

$$\sum_{s \in S(l')} x_s = c_{l'} \Rightarrow \lambda_{l'} \sum_{s \in S(l')} a_s = c_{l'} \Rightarrow \lambda_{l'} = \frac{c_{l'}}{\sum_{s \in S(l')} a_s} \quad (30)$$

combining (28)-(29), leads to

$$x_i = \frac{a_i c_{l'}}{\sum_{s \in S(l')} a_s} \qquad \forall i \in S(l') \quad (31)$$

Therefore, (31) shows that in a network with single congested link, the sources passing through the congested link, achieve their rates in proportion to their weights. For networks with multiple congested

links, such an insight might not be easily seen, however weight factors influence the capacity sharing at bottle neck links. In this respect, we can allocate more resources, i.e. link capacity, to some specified sources by assigning larger weights to them.

6. Simulation Results

In this section we examine the proposed flow control algorithm, listed above as Algorithm 1, for a typical NoC architecture. In our scenario, we have used a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. In our scheme, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bit long.

In order to simulate our scheme, some nodes are considered to have a GS data (such as Multimedia, etc.) to be sent while other nodes have a BE traffic. As stated before, GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE traffics. Routing policy for BE sources is shown in Fig. 1. We assume that all sources have logarithmic utility function of the form $U_s(x_s) = a_s \log x_s$ where a_s represents the weight factor for source *s*. In the sequel, we present our results in the following parts as below.

One of the most significant issues of our interest is the convergence behavior of the source rates. In this part, we have simulated our scheme using 2 different values for step-size, 1.05 and 0.2, respectively. Weight factor for all sources is assumed to be unity. The convergence behavior of source rates for after 150 iterations is depicted in Fig. 2(a)-(b). Regarding Fig. 2(a), it's apparent that for $\gamma = 1.05$, after 20 iteration steps the source rates will have very little variations, however, from Fig. 2(b) , i.e. for $\gamma = 0.2$, these threshold of iterations will be at least 85 steps.

In order to have a better insight about the algorithm behavior, the relative error with respect to optimal source rates which is averaged over all active sources, is also shown in Fig. 3. Optimal values are obtained using CVX [17] which is MATLAB toolbox for solving disciplined convex optimization problems. Fig. 3 reveals the first step size leads to less than 10% error in average just after about 13 iteration steps, and after 20 steps the average error lies below 5%. However, the second step size would reach the two aforementioned error margins at the expense of iterating for about 60 and 75 steps, respectively. Although not shown in Fig. 3, with much more iteration steps simulation results verify that the average error curve for the smaller step size lies below that of larger step size. However, for practical implementations and real world applications, due to faster convergence speed, larger step size is more appropriate.

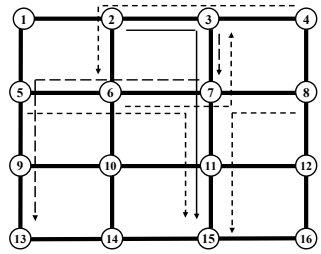
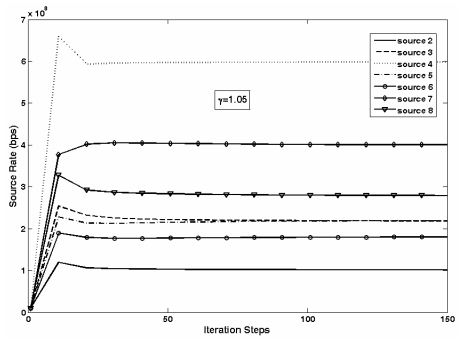
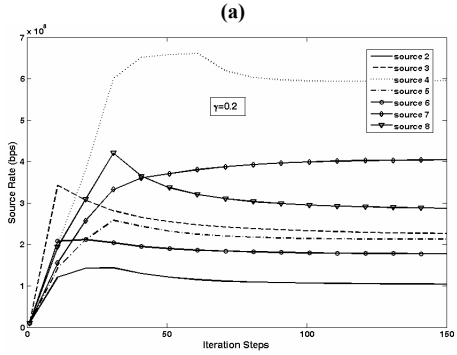


Fig. 1. Network Topology and Routing Policy



(a)



(b)

Fig. 2. Source rates convergence with symmetric weight factors for (a) $\gamma = 1.05$ **and (b)** $\gamma = 0.2$

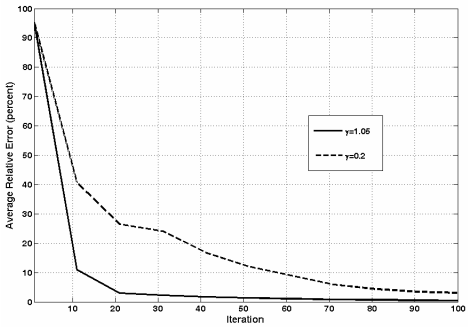


Fig. 3. Average Relative Error

7. Conclusion and Future Works

In this paper we addressed the problem of flow control for BE traffic in NoC systems. Flow control was considered as the solution to the utility maximization problem which was solved indirectly through its dual using gradient projection method. This was led to an iterative algorithm which can be used to determine optimal BE source rates.

The algorithm can be implemented by a controller which admits a light communication and communication overhead to the system. We have also investigated the convergence behavior of the algorithm. Further investigation about the effect of delay incurred by the proposed algorithm is the main direction of our future studies.

8. References

[1] L. Benini, and G. DeMicheli, "Networks on Chips: A New SoC Paradigm." *Computer*, 2002, vol. 35, no. 1, pp. 70-78.

[2] W. J. Dally, and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks." *Design Automation Conference*, 2001, pp. 684-689.

[3] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness, and stability." *Operational Research Society*, 1998, vol. 49, no. 3, pp. 237–252.

[4] S. Mascolo, "Classical control theory for congestion avoidance in high-speed internet" *Decision and Control IEEE Conference*, 1999, vol. 3, pp. 2709-2714.

[5] Y. Gu, H. O. Wang and L.G. Yiguang Hong Bushnell, "A predictive congestion control algorithm for high speed communication networks." *American Control Conference*, vol. 5, pp. 3779-3780, 2001.

[6] C. Yang, and A. V. S. Reddy, "A taxonomy for congestion control algorithms in packet switching networks." *IEEE Network*, 1995, vol. 9, no. 4, pp. 34-45.

[7] U. Y. Ogras, and R. Marculescu, "Prediction-based flow control for network-on-chip traffic." *In Proceedings of the Design Automation Conference,* 2006.

[8] J. W. van den Brand, C. Ciordas, K. Goossens and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip." *Design, Automation and Test in Europe Conference*, 2007, pp. 948-953.

[9] Hu. Jingcao, and R. Marculescu, "DyAD - smart routing for networks-on-chip." *Design Automation Conference*, 2004, pp. 260- 263.

[10] M. S. Talebi, F. Jafari, A. Khonsari, "A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization." *In proceedings of the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2007.

[11] M. S. Talebi, F. Jafari, A. Khonsari, and M. H. Yaghmaee, "A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization." *International Conference on Computational Science and its Applications*, 2007, pp. 398-409.

[12] S. H. Low, and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence." *IEEE/ACM Transactions on Networking*, 1999, vol. 7, no. 6, pp. 861-874.

[13] Boyd, S., and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[14] Bertsekas, D. P., *Nonlinear Programming*, Athena Scientific, 1999.

[15] Bertsekas, D. P., and J. N. Tsitsiklis, *Parallel and distributed computation*, Prentice-Hall, 1989.

[16] Boyd, S., *Convex Optimization II Lecture Notes*, Stanford University, 2006.

[17] Grant, M., S. Boyd, and Y. Ye, *CVX (Ver. 1.0RC3): Matlab Software for Disciplined Convex Programming*. Download available at: http://www.stanford.edu/~boyd/cvx.

Paper 4

A Novel Congestion Control Scheme in Network-on-Chip Based on Best Effort Delay-Sum Optimization

F. Jafari
M. S. Talebi
A.Khonsari
M. H. Yaghmaee

In the Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), pp. 191-196, Sydney, NSW, Australia, May 2008.

A Novel Congestion Control Scheme in Network-on-Chip Based on Best Effort Delay-Sum Optimization

Fahimeh Jafari[1,2], Mohammad S. Talebi[2], Ahmad Khonsari[3,2], Mohammad H. Yaghmaee[1]
[1]Ferdowsi University of Mashhad, Mashhad, Iran.,
[2]IPM, School of Computer, Tehran, Iran.,
[3]ECE Department, University of Tehran, Tehran, Iran.,
jafari@ipm.ir, mstalebi@ipm.ir, ak@ipm.ir, hyaghmae@ferdowsi.um.ac.ir

Abstract

With the advances of the semiconductor technology, the enormous number of transistors available on a single chip allows designers to integrate dozens of IP blocks together with large amounts of embedded memory. This has been led to the concept of Network on a Chip (NoC), in which different modules would be connected by a simple network of shared links and routers and is considered as a solution to replace traditional bus-based architectures to address the global communication challenges in nanoscale technologies. In NoC architectures, controlling congestion of the best effort traffic will continue to be an important design goal. Towards this, employing end-to-end congestion control is becoming more imminent in the design process of NoCs. In this paper, we introduce a centralized algorithm based on the delay minimization of Best Effort sources. The proposed algorithm can be used as a mechanism to control the flow of Best Effort source rates by which the sum of propagation delays of network is to be minimized.

1. Introduction

With the emergence of complex VLSI chips, the designers are facing several new challenges. Nowadays, *application-specific integrated circuits* (ASICs) have evolved into *systems-on-chip* (SoCs), where dozens of predesigned IP cores are assembled together to form large chips with complex functionality.

A recently proposed platform for the on-chip interconnects is the *network-on-chip* (NoC) architecture, where the IPs are usually placed on a grid of tiles and networking protocols governs the communication between tiles. Such a regular structures are very attractive because they can offer well-controlled electrical parameters, which enable high-performance circuits by reducing the latency and increasing the bandwidth. In fact, NoCs provide enhanced performance and scalability, in comparison with previous communication architectures. The advantages of NoC are achieved thanks to efficient sharing of wires and a high level of parallelism [1].

The provision of *Quality-of-Service* (QoS) in NoC's environment is currently the focus of much discussion in research community. NoCs are expected to serve as multimedia servers and are required not only to carry Best Effort (BE) traffic, but also Guaranteed Service (GS) traffic which requires tight performance constraints such as necessary bandwidth and maximum delay boundaries. Networks with BE services must choose a mechanism to avoid congestion. Congestion control in NoCs is a novel issue and usually studied regarding minimizing the network cost (in delay, area and power) or maximizing network utility while maintaining the required QoS, as we will focus on it in more detail later.

2. Related Works

During the past few years, many strategies for congestion control have been proposed for off-chip networks [2-5]. Congestion control for on-chip networks is still a novel issue, however this problem has been investigated by several researchers [6]-[8]. In [6], a prediction-based congestion control strategy for on-chip networks has been proposed where each router predicts buffer occupancies to detect future congestion problems. In [7] the link utilization has been used as congestion measure and the controller determines the appropriate loads for the BE sources. Dyad [8] overcomes the congestion by switching from

978-0-7695-3125-0/08 $25.00 © 2008 IEEE
DOI 10.1109/I-SPAN.2008.45

deterministic to adaptive routing when the NoC is going to be congested.

The main purpose of this paper is to present a congestion control as the solution to a delay minimization problem for choosing the rate of BE sources. Our approach is different from the aforementioned works, e.g. [6][7], in which no delay consideration were taken into account. We present an algorithm as the solution to the optimization problem. To evaluate the performance of the proposed approach, we simulate the congestion control algorithm under a NoC-based scenario.

This paper is organized as follows. In Section 3 we present the system model and formulate the underlying optimization problem for BE congestion control. In Section 4 we solve the optimization problem using an iterative algorithm and propose the solution as a centralized congestion control algorithm to be implemented as a controller. In Section 5 we analyze the convergence behavior of the proposed algorithm and prove the underlying theorem of its convergence. In Section 6 we present the simulation results. Finally, the section 7 concludes the paper.

3. System Model

We consider a NoC with two dimensional mesh topology and wormhole routing. In wormhole networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in a pipeline fashion. The NoC architecture is assumed to be lossless, and packets traverse the network on a shortest path using a deadlock free XY routing. We model the congestion control problem in NoC as the solution to an optimization problem. For more convenience, we turn the aforementioned NoC architecture into a mathematical model as in [9]. In this respect, we consider NoC as a network with a set of bidirectional links L and a set of sources S. A source consists of Processing Elements (PEs), routers and Input/Output ports. Each link $l \in L$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of c_l packets/sec. We denote the set of sources that share link l by $S(l)$. Similarly, the set of links that source s passes through, is denoted by $L(s)$. By definition, $l \in S(l)$ if and only if $s \in L(s)$.

As previously stated, there are two types of traffic in a NoC: GS and BE. For notational convenience, we divide S into two parts, each one representing sources with the same traffic. In this respect, we denote the set of sources with BE and GS traffic by S_{BE} and S_{GS}, respectively. Each link l is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE sources.

3.1. Delay Model

In recent years, researchers have presented different delay models in NoC (e.g. [10] and references therein). Due to simplicity of the model introduced in [10], we adopt its model in our framework.

Interconnects and network routers are two fundamental parts of the NoC which are subject to power consumption and communication latency. In our model, the delay of link $l \in L$ is denoted by d_l which represents the delay incurred to the system by packet propagation over this link. More precisely, d_l is given by

$$d_l = d_w + d_r \tag{1}$$

where d_w and d_r are delay of unit flow on interconnects and routers, respectively. In this respect, when a flow of amount f_l passes through link l, the total latency is:

$$D_l = f_l d_l \tag{2}$$

Interconnect or wire delay, d_w, is closely related to the wire styles. We assume that four types of wire styles are available for interconnects, namely, RC wires with repeated buffers with wire pitch varying from 1×, 2×, and 4× minimum global wire pitch, and on-chip transmission line with wire pitch equal to 16 micron. For RC wires with repeated buffers, we assume d_w is proportional to wire length, as below:

$$d_w = \text{per grid length delay} \times \text{wire length}$$

On the other hand, for on-chip transmission line, relatively large setup cost should be added to d_w. We use transmission line model proposed by Chen et al. [11] to estimate transmission line delay. Table 1 lists delay per grid length (2mm) of these four types of wire styles in 0.18 micron design technology. Setup cost of 50ps is added to d_w for transmission line.

We use the router delay model proposed by Peh et al. [12] to estimate NoC router delay. Table 2 shows

Table1: Delay Model of Wires

Wire Type	RC-1x	RC-2x	RC-4x	T-line
d_w (ns)	0.127	0.112	0.100	0.020

latency of routers in 0.18 micron technology node. When router input/output ports increase, d_r increases almost linearly.

Table 2: Model of Routers

Ports	2	3	4	5	6	7	8
d_r (ns)	0.599	0.662	0.709	0.756	0.788	0.819	0.835

3.2. Flow Control Model

Our objective is to choose source rates (IP loads) of BE traffics so that to minimize the sum of delays of all BE traffics. Hence the minimization problem can be formulated as:

$$\min_{x_s} \sum_{l=1}^{L} D_l \tag{3}$$

subject to:

$$\sum_{s\in S_{BE}(l)} x_s + \sum_{s\in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{4}$$

$$\sum_{s\in S_{BE}} x_s \ge f \tag{5}$$

$$x_s > 0 \qquad \forall s \in S_{BE}$$

where source rates, i.e. x_s, $s \in S$, are optimization variables.

Regarding (2), we rewrite (3) as below:

$$\min_{x_s} \sum_{l=1}^{L} \Big(\sum_{s\in S_{BE}(l)} x_s \Big) d_l \tag{6}$$

subject to:

$$\sum_{s\in S_{BE}(l)} x_s + \sum_{s\in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{7}$$

$$\sum_{s\in S_{BE}} x_s \ge f \tag{8}$$

$$x_s > 0 \qquad \forall s \in S_{BE}$$

The constraint (7) says that the aggregate BE source rates passing thorough link l cannot exceed its free capacity, i.e. the portion of the link capacity which has not been allocated to GS sources. The constraint (8) says that the sum of BE source rates must be at least f. For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s\in S_{GS}(l)} x_s \tag{9}$$

Hence, (7) can be rewritten as:

$$\sum_{s\in S_{BE}(l)} x_s \le \hat{c}_l \qquad \forall l \in L \tag{10}$$

Although problem (6) can be separated across sources, its constraints will remain coupled across the network. Due to coupled nature of such constrained problems, they have to be solved using centralized methods like interior point methods [13]. Such computations may pose great overheads on the system. Instead of such methods, we seek to obtain the solution with simpler operations. One way is to use the *Projected Gradient Method* for constrained optimization problems [13] which will be briefly reviewed in the next section.

For notational convenience in solving the problem, we use matrix notation. In this respect, we define Routing matrix, i.e. $R = [R_{ls}]_{L\times S}$, as following:

$$R_{ls} = \begin{cases} 1 & \text{if } s \in S_{BE}(l) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

We also define the source rate vector (for BE traffic), link delay and link capacity vectors as $x = (x_s, s \in S_{BE})$, $d = (d_l, l \in L)$ and $\hat{c} = (\hat{c}_l, l \in L)$, respectively. Therefore problem (6) can be rewritten in the matrix form as follows:

$$\min_{x_s} \; d^T R x \tag{12}$$

subject to:

$$Rx \le \hat{c} \tag{13}$$

$$\mathbf{1}^T x \ge f \tag{14}$$

$$x_s > 0 \qquad \forall s \in S_{BE}$$

where $\mathbf{1}$ is a vector with all one.

4. Congestion Control Algorithm

In this section, we will solve the optimization problem using Projected Gradient Method for constrained problems [13][14] and present a congestion control scheme for BE traffic in NoC systems to overcome the congestion.

The Projected Gradient Method for constrained minimization problems is very similar to the original one which only applies to unconstrained ones [13]. We briefly review this method in the following.

Consider the constrained minimization problem

$$\min_{x} f_o(x) \tag{15}$$

subject to:

$$f_i(x) \le 0, \qquad i = 1..m \tag{16}$$

in which $f_i : R^n \to R$ are convex functions.

In order to solve (15) iteratively, we define the following minimizing sequence

$$x^{(k+1)} = x^{(k)} - \gamma_k g^{(k)} \tag{17}$$

where

$$g^{(k)} \in \begin{cases} \nabla f_o(x) & f_j(x) \le 0, \quad i = 1,...,m \\ \nabla f_j(x(t)) & f_j(x) > 0 \end{cases} \tag{18}$$

and γ_k is the step size which satisfies:

$$\gamma_k > 0 \tag{19}$$

$$\gamma_k \to 0 \tag{20}$$

$$\sum_{k=1}^{\infty} \gamma_k = \infty \tag{21}$$

To quantify the performance of the method we define

$$f_{best}^k = \min\left\{ f_o(x^{(i)}) \middle| x^{(i)} \ feasible, \ i = 1,...,k \right\}$$

The following lemma, states the conditions on g under which the minimizing sequence (17) converges to the optimal point of (15), i.e. $f_{best}^k \to f^*$ and $x^k \to x^*$ as $k \to \infty$.

Lemma 1: *Consider the constrained minimization problem, as in (15). The minimizing sequence defined by (17) and (18) with the stepsize satisfying (19)-(21), converges to the optimal point of (15), i.e. x^*, if the following conditions hold*

$$\left\| g^{(k)} \right\|_2 \le G \tag{22}$$

$$\left\| x^{(1)} - x^* \right\|_2 \le E \tag{23}$$

Proof: *See [14].*

In the sequel, we will solve the optimization problem (12) using Projected Gradient Method for constrained problems as stated in Lemma 1. Regarding (17), we have to calculate $g^{(k)}$. According to (18), if $x^{(k)}$ is feasible, i.e. $Rx \le \hat{c}$ and $\mathbf{1}^T x \ge f$, we have:

$$g = \nabla d^T Rx = \nabla R^T dx = R^T d \tag{24}$$

otherwise at least one of the constraints must be violated. Assume link capacity constraint is violated for link l', i.e. $\sum_{s \in S_{BE}(l')} x_s > \hat{c}_{l'}$. Rewriting this in matrix form, yields:

$$\mathbf{e}_{l'}^T (Rx - \hat{c}) > \mathbf{0} \tag{25}$$

where $\mathbf{e}_{l'}$ is the l' th unit vector of R^L space which is zero in all entries except the l' th at which it is 1. Therefore, g is given by:

$$g = \nabla \mathbf{e}_{l'}^T (Rx - \hat{c}) = R^T \mathbf{e}_{l'} \tag{26}$$

Assuming that link capacity constraints are being satisfied, the sum-rate constraint is violated, i.e. $\mathbf{1}^T x < f$, or equivalently in the standard form as in (16), $f - \mathbf{1}^T x > 0$. Therefore g is given by:

$$g = -\nabla \mathbf{1}^T x = -\nabla x^T \mathbf{1} = -\mathbf{1} \tag{27}$$

Using (24), (26) and (27), the update equation to solve problem (12) is given by:

$$x_s^{(k+1)} = \left[x_s^{(k)} - \gamma_k g^{(k)} \right]^+ \tag{28}$$

where $[z]^+ = \max\{z, 0\}$ to satisfy non-negativity of source rate and $g^{(k)}$ is given by:

$$g^{(k)} = \begin{cases} R^T d & \sum_{s \in S_{BE}(l)} x_s(t) \le \hat{c}_l, \forall l \ \text{ and } \ \mathbf{1}^T x \ge f \\ R^T e_{l'} & \sum_{s \in S_{BE}(l')} x_s(t) > \hat{c}_{l'}, \quad \exists l' \\ -\mathbf{1} & \mathbf{1}^T x \le f \end{cases} \tag{29}$$

(28) and (29) together propose an iterative algorithm as the solution to problem (12). In this respect, optimal source rates for BE sources can be found while satisfying capacity constraints and preserving GS traffic requirements. Thus, the aforementioned algorithm can be employed to control the congestion of the BE traffic in the NoC. The iterative algorithm is decentralized in the nature and can be addressed in distributed scenarios. However, due to well-formed structure of the NoC, we focus on a centralized scheme; we consider a controller to be mounted in the NoC to implement the proposed algorithm. The necessary requirement of such a controller is the ability to accommodate simple mathematical operations as in (28) and (29) and the allocation of few wires to communicate congestion control information to nodes with a light GS load. Algorithmic realization of proposed Congestion-Controller for BE traffic is listed below as Algorithm 1.

5. Convergence Analysis

In this section, we investigate the convergence analysis of the proposed algorithm using a time-varying stepsize in (28). As stated in the previous section, in this paper the stepsize is selected as (19)-(21).

Theorem 1: *The iterative congestion control scheme proposed by (28) and (29) with a time-varying stepsize which satisfies (19)-(21), will converge to the optimal point of problem (6).*

Proof: By lemma 1, it is clear that if its assumptions hold, the proof of Theorem is done. In this respect, $g^{(k)}$ should admit an upper bound in l_2-norm. In doing so, it suffices to show that its gradient is upper bounded in l_2-norm. Considering (29), we have

$$\begin{aligned}\left\|g^{(k)}\right\|_2 &\le \max\{\|-\mathbf{1}\|_2, \left\|R^T d\right\|_2, \left\|R^T \mathbf{e}_{l'}\right\|_2\} \\ &= S\end{aligned} \tag{30}$$

Hence g in l_2-norm is bounded at least with S.

In the next step, we show that the Euclidian distance of the initial point to the optimal point is bounded at least with D, i.e.

$$\exists D > 0 \qquad \text{s.t.} \qquad \left\|x(1) - x^*\right\|_2 \le D \tag{31}$$

We have $x_s > 0, \ \forall s \in S_{BE}$. On the other hand, optimal source rates are bounded at most with maximum value of link capacities, i.e.

$$\max_s x_s^* \le \max_l \hat{c}_l \le \max_l c_l \tag{32}$$

Therefore,

$$\begin{aligned}\left\|x(1) - x^*\right\|_2 &\le \left\|\max x^* - \min x(1)\right\|_2 \\ &= \left\|\max_l c_l - 0\right\|_2 \\ &= L c_{l\max}\end{aligned} \tag{33}$$

Hence the initial Euclidian distance is bounded with at least $Lc_{l\max}$. (30) and (33) complete the proof.

6. Simulation Results

In this section we examine the proposed congestion control algorithm, listed above as Algorithm 1, for a typical NoC architecture. We have simulated a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. In our scenario, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bit long.

One of the most significant issues of our interest, is the convergence behavior of the source rates. The step size is chosen to be $\gamma_k = 3/(1+k)$ which apparently satisfies (19)-(21). Variation of source rates for some nodes using above parameters are shown in Fig. 1.

Regarding Fig. 1, it's apparent that after about 270 iteration steps, all source rates will be in the vicinity of the steady state point of the algorithm; however, for the second case, at least 600 iteration steps is needed that after which the source rates to be in the vicinity of the steady state point.

Algorithm 1: Congestion Control for BE Traffics in NoC

Initialization:

1. **Initialize $\hat{c}_l$ of all links.**
2. **Set source rate vector to zero.**

Loop:

Do until $(\max\left|x_s^{(k+1)} - x_s^{(k)}\right| < Error)$

1. $\forall s \in S$ **: Compute new source rate:**

$$x_s^{(k+1)} = \left[x_s^{(k)} - \gamma_k g^{(k)}\right]^+$$

where γ_k **can be selected as** $\gamma_k = a/(b+t)$ and

$$g^{(k)} = \begin{cases} R^T d & \sum_{s \in S_{BE}(l)} x_s(t) \le \hat{c}_l, \forall l \ \ and \ \ 1^T x \ge f \\ R^T e_{l'} & \sum_{s \in S_{BE}(l')} x_s(t) > \hat{c}_{l'}, \exists l' \\ -1 & 1^T x \le f \end{cases}$$

Output:

Communicate BE source rates to the corresponding nodes.

In order to have a better insight about the algorithm behavior, the relative error with respect to optimal rates which averaged over all sources, is also shown in Fig. 2. Optimal source rates are obtained using CVX [15] which is MATLAB-based software for solving disciplined convex optimization problems. As shown in Fig. 2, it is clear that after about 380 steps, the average of relative error of all sources falls below 20%, which is acceptable in practice. Thus, the proposed congestion control algorithm is computationally tractable.

Our final result is devoted to investigate the performance of algorithm 1 in terms of sum of delay in the network. In this respect, we have calculated sum of the delay for two cases; using Algorithm 1 and using uniform rate allocation. The result is depicted in Fig. 3. As a comparison, we conclude that the delay-sum is reduced at least by a factor of two which verifies the aim of the underlying optimization problem in source assignment in terms of delay-sum reduction.

7. Conclusion and Future Works

In this paper we addressed the problem of congestion control for BE traffic in NoC systems. Congestion control was modeled as the solution to the delay-sum minimization problem which was solved using gradient projection method for constrained

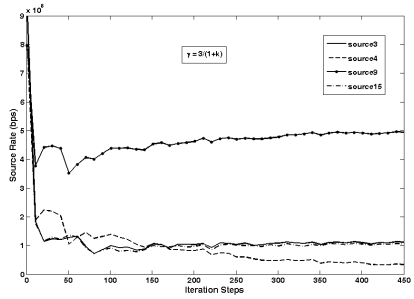


Fig. 1. Source rates for $\gamma_k = 3/(1+k)$

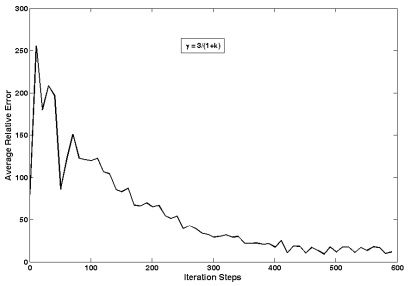


Fig. 2. Average of Error with respect to optimal solution for $\gamma_k = 3/(1+k)$

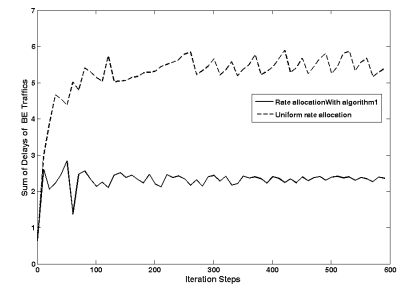


Fig. 3. Delay-Sum Comparison between proposed rate allocation and uniform rate allocation

optimization problems. This was led to an iterative algorithm which determine optimal BE source rates. We have also studied the realization of the algorithm as a centralized congestion controller. Simulation results confirm that the proposed algorithm converges and the computational overhead of the congestion control algorithm is small.

8. References

[1] L. Benini and G. DeMicheli, "Networks on Chips: A New SoC Paradigm." *Computer*, vol. 35, no. 1, pp. 70-78, 2002.

[2] F. P. Kelly, A. Maulloo and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness, and stability." *Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[3] S. Mascolo. "Classical control theory for congestion avoidance in high-speed internet." *Decision and Control IEEE Conference*, vol. 3, pp. 2709-2714, 1999.

[4] Y. Gu, H. O. Wang and L.G. Yiguang Hong Bushnell, "A predictive congestion control algorithm for high speed communication networks." *American Control Conference*, vol. 5, pp. 3779-3780, 2001.

[5] C. Yang and A. V. S. Reddy, "A taxonomy for congestion control algorithms in packet switching networks." *IEEE Network*, vol. 9, no. 4, pp. 34-45, 1995.

[6] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic." *In Proceedings of the Design Automation Conference,* 2006.

[7] J. W. van den Brand, C. Ciordas, K. Goossensl and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip*." Design, Automation and Test in Europe Conference*, pp. 948-953, 2007.

[8] J. Hu and R. Marculescu, "DyAD - smart routing for networks-on-chip." *Design Automation Conference*, pp. 260- 263, 2004.

[9] S. H. Low, D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861-874, 1999.

[10] Y. Hu, Y. Zhu, H. Chen, R. Graham and C. K. Cheng, "Communication latency aware low power NoC synthesis." *Annual ACM IEEE Design Automation Conference*, pp. 574 – 579, 2006.

[11] H. Chen, R. Shi, C.K. Cheng, and D. Harris, "Suriner: A Distortionless Electrical Signaling Scheme for Speed of Light On-Chip Communications." *IEEE Intl. Conf. on Computer Design*, pp.497-502, 2005.

[12] L.S. Peh, "Flow Control and Micro-Architectural Mechanisms for Extending the Performance of Interconnection Networks." *Ph.D. Thesis, Stanford University*, 2001.

[13] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[14] S. Boyd, Convex Optimization II Lecture Notes, Stanford University, 2006.

[15] M. Grant, S. Boyd and Y. Ye, CVX (Ver. 1.0RC3): Matlab Software for Disciplined Convex Programming. Download available at: http://www.stanford.edu/~boyd/cvx.

Paper 5

Max-Min-Fair Best Effort Flow Control in Network-on-Chip Architectures

F. Jafari
M. H. Yaghmaee
M. S. Talebi
A. Khonsari

In the Proceedings of the International Conference on Computational Science (ICCS), Part I, LNCS 5101, pp. 436-445, Krakow, Poland, June 2008.

Max-Min-Fair Best Effort Flow Control in Network-on-Chip Architectures

Fahimeh Jafari[1,2], Mohammad H. Yaghmaee[1], Mohammad S. Talebi[2], and Ahmad Khonsari[3,2]

[1] Ferdowsi University of Mashhad, Mashhad, Iran
[2] IPM, School of Computer, Tehran, Iran
[3] ECE Department, University of Tehran, Tehran, Iran
{jafari,ak}@ipm.ir, hyaghmae@ferdowsi.um.ac.ir, mstalebi@gmail.com

Abstract. *Network-on-Chip* (NoC) has been proposed as an attractive alternative to traditional dedicated busses in order to achieve modularity and high performance in the future *System-on-Chip* (SoC) designs. Recently, end to end flow control has gained popularity in the design process of network-on-chip based SoCs. Where flow control is employed, fairness issues need to be considered as well. In fact, one of most difficult aspects of flow control is that of treating all sources fairly when it is necessary to turn traffic away from the network. In this paper, we proposed a flow control scheme which admits Max-Min fairness criterion for all sources. In fact, we formulated Max-Min fairness criterion for the NoC architecture and presented implementation to be used as flow control mechanism.

Keywords: Network-on-Chip, flow control, Max-Min fairness.

1 Introduction

Network-on-Chip (NoC) is a new paradigm structure for designing future *System-on-Chips* (SoC) [1]**.** A typical NoC architecture provides a scalable communication infrastructure for interconnecting cores. Since the communication infrastructure as well as the cores from one design can be easily reused for a new product, NoC provides maximum possibility for reusability.

NoCs with their flexible and scalable interconnect provide high computational power to support computationally extensive multimedia applications, i.e. those that combine audio, video and data. In contrast to simple data applications, which can work without guarantees of timing of data delivery, multimedia applications require a guaranteed degree of service in terms of required bandwidth and timelines. According to the networking terminology, we refer to the traffic of simple data as elastic or *Best Effort* (BE) traffic and to multimedia traffic as inelastic or *Guaranteed Service* (GS) traffic.

Due to the rapid growth of the number of *processing elements* (PEs) in NoCs [2], employing efficient policy for flow control is inevitable in the design of NoCs to

M. Bubak et al. (Eds.): ICCS 2008, Part I, LNCS 5101, pp. 436–445, 2008.

provide the required *Quality of Service* (QoS). A NoC should support network level flow control in order to avoid congestion in the bottleneck links, i.e. link through which several sources pass [3]. The design and control of NoCs raises several issues well suited to study using techniques of operational research such as optimization and stochastic modeling. Recently, some novel researches have been embarked in studying congestion control in NoCs [4-5]. Congestion control schemes in NoCs mainly focus on utilizing NoC's resources, with the aim of minimizing network cost or maximizing network utility while maintaining the required QoS for Guaranteed Service traffics.

Many strategies for flow control have been proposed for off-chip networks, e.g. data networks, etc. [6-9]. On-chip networks pose different challenges. For instance, in off-chip environments, to overcome congestion in links, packet dropping is allowed. On the contrary, reliability of on-chip wires makes NoCs a loss-less environment.

So far, several works have addressed this problem for NoC systems. In [4], a prediction-based flow-control strategy for on-off traffic in on-chip networks is proposed where the prediction is used in router to be aware of buffer fillings. In [5] a flow-control scheme for Best Effort traffic based on Model Predictive Control is presented, in which link utilization is used as congestion measure. Dyad [10] controls the congestion by switching from deterministic to adaptive routing when system is going to be congested. [11] proposes a flow control scheme as the solution to rate-sum maximization problem for choosing the BE source rates. The solution to the rate-sum optimization problem is presented as a flow control algorithm.

Where flow control is employed, fairness issues need to be considered as well [3]. In fact, one of most difficult aspects of flow control is to choose a policy to accommodate a fair rate allocation. All of the abovementioned studies only regarded the flow control by taking into account the constraints of the system and to the best of our knowledge no policy to maintain fairness among sources was chosen.

The fairness of TCP-based flow control algorithms was first analyzed in [12]. The analysis in [12] was based on a single bottleneck link. Different flow control approaches can be classified with respect to the fairness criteria, in favor of which rate allocation is done. One of the famous forms of fairness criterion is Max-Min fairness, which has been discussed in earlier literature and described clearly in [13]. Our main contribution in this paper is to present a flow control scheme for Best Effort traffic in NoC which satisfies Max-Min fairness criterion. Our framework is mainly adopted from the seminal work [13] which presents a basic Max-Min fairness optimization problem. In this paper, we reformulate such a problem for the NoC architecture.

The organization of the paper is as follows. In Section 2 we present the system model, the concept of Max-Min fairness and formulation of the flow control as an optimization problem. In section 3 we present an iterative algorithm as the solution to the flow control optimization problem. Section 4 presents the simulation results and discussion about them. Finally, the section 5 concludes the paper and states some future work directions.

2 System Model

We consider a NoC with two dimensional mesh topology, a set S of sources and a set L of bidirectional links. Let c_l be the finite capacity of link $l \in L$. The NoC assumed to use wormhole routing. In wormhole-routed networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in a pipeline fashion. The NoC architecture is also assumed to be lossless, and packets traverse the network on a shortest path using a deadlock free XY routing. A source consists of Processing Elements (PEs), routers and Input/Output ports. Each link is a set of wires, busses and channels that are responsible for connecting different parts of the NoC. We denote the set of sources that share link l by $S(l)$. Similarly, the set of links that source s passes through is denoted by $L(s)$. By definition, $l \in S(l)$ if and only if $s \in L(s)$.

We assume that there are two types of traffic in the NoC: GS and BE traffic. For notational convenience, we divide S into two parts, each one representing sources with the same kind of traffic. In this respect, we denote the set of sources with BE and GS traffic by S_{BE} and S_{GS}, respectively. Each link l is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE sources.

2.1 Max-Min Fairness Concept

Any discussion of the performance of a rate allocation scheme must address the issue of fairness, since there exist situations where a given scheme might maximize network throughput, for example, while denying access for some users or sources. Max-Min fairness is one the significant fairness criteria. Crudely speaking, a set of rates is max-min fair if no rate can be increased without simultaneously decreasing another rate which is already smaller. In a network with a single bottleneck link, max-min fairness simply means that flows passing through the bottleneck link would have equal rates.

The following definition states the formal definition of Max-Min fairness.

Defination 1. A feasible rate allocation $x = (x_s, s \in S)$ is said to be "max-min fair" if and only if an increase of any rate within the domain of feasible allocations must be at the cost of a decrease of some already smaller rate. Formally, for any other feasible allocation y, if $y_s > x_s$ then there must exist some s' such that $x_{s'} \leq x_s$ and $y_{s'} < x_{s'}$ [13].

Depending on the network topology, a max-min fair allocation may or may not exist. However, if it exists, it is unique (see [14] for proof).

In what follows the condition under which the Max-Min rate allocation exists will be stated. Before we proceed to this condition, we define the concept of bottleneck link.

Defination 2. With our system model above, we say that link l is a bottleneck for source s if and only if

1. link l is saturated: $\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s = c_l$
2. source s on link l has the maximum rate among all sources using link l.

Intuitively, a bottleneck link for source s is a link which limits x_s.

Theorem 1. A max-min fair rate allocation exists if and only if every source has a bottleneck link (see [14] for proof).

2.2 Flow Control Model

Our focus will be on two objectives. First, choosing source rates (IP loads) of BE traffics so that to accomplish flow control in response to demands at a reasonable level. Second, maintaining Max-Min fairness for all sources. We model the flow control problem in NoC as the solution to an optimization problem. For more convenience, we turn the aforementioned NoC architecture into a mathematical model as in [5]. In this respect, the Max-Min fairness flow control problem can be formulated as:

$$\max_{x} \min_{s \in S} x_s \tag{1}$$

subject to:

$$\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{2}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{3}$$

where source rates, i.e. x_s, $s \in S$, are optimization variables.

The constraint (2) says the aggregate BE source rates passing thorough link l cannot exceed its free capacity, i.e. the portion of the link capacity which has not been allocated to GS sources. For notational convenience, we define

$$u = \min_{s \in S} x_s$$

$$\hat{c}_l = c_l - \sum_{s \in S_{GS}(l)} x_s ,$$

therefore the above mentioned problem can be rewritten as:

$$u = \min_{s \in S} x_s \tag{4}$$

$$\max \ u \tag{5}$$

subject to:

$$\sum_{s \in S_{BE}(l)} x_s \le \hat{c}_l \qquad \forall l \in L \tag{6}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{7}$$

To solve the above problem, it should be converted so as to be in the form of disciplined optimization problems [15] as follows:

$$\max u \tag{8}$$

subject to:

$$u \leq x_s \quad \forall s \in S \tag{9}$$

$$\sum_{s \in S_{BE}(l)} x_s \leq \hat{c}_l \qquad \forall l \in L \tag{10}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{11}$$

The above optimization problem can be solved using several methods. In the next section, we introduce a simple and famous algorithm, known as "progressive filling", to solve (8) iteratively.

In order to compare the results of progressive filling algorithm with the exact values, we solve problem (8) using CVX [16] which is a MATLAB-based software for disciplined convex optimization problems, whose results will be given in section 4.

3 Max-Min Fairness Algorithm

Theorem 1 is particularly useful in deriving a practical method for obtaining a max-min fair allocation, called "progressive filling". The idea is as follows: rates of all flows are increased at the same pace, until one or more links are saturated. The rates of flows passing through saturated links are then frozen, and the other flows continue to increase rates. All the sources that are frozen have a bottleneck link. This is because they use a saturated link, and all other sources using the saturated link are frozen at the same time, or were frozen before, thus have a smaller or equal rate. The process is repeated until all rates are frozen. Lastly, when the process terminates, all sources have been frozen at some time and thus have a bottleneck link. Using Theorem 1, the allocation is max-min fair.

Theorem 2. For the system model defined above, with fixed routing policy, there exists a unique max-min fair allocation. It can be obtained by the progressive filling algorithm. (see [14] for proof)

In the sequel, we derive the max-min rate allocation as the solution to problem (8) and based on this algorithmic solution, we present a flow control scheme for BE traffic in NoC systems.

Thus, the aforementioned algorithm can be employed to control the flow of BE traffic in the NoC. The iterative algorithm can be addressed in distributed scenario. However, due to well-formed structure of the NoC, we focus on a centralized scheme; we use a controller like [5] to be mounted in the NoC to implement the above algorithm. The necessary requirement of such a controller is the ability to accommodate simple mathematical operations and the allocation of few wires to communicate flow control information to nodes with a light GS load.

Algorithm 1. Max-Min Fair (MMF) Flow Control Algorithm for BE in NoC.

Initialization:

1. **Initialize** $\hat{c}_l$ **of all links.**
2. **Define:**
   a. T **as the set of sources not passing through any saturated link.**
   b. B **as the set of saturated links.**
   c. $\overline{B} = L - B$ **and** $\overline{T} = S_{BE} - T$ **.**
3. **Set source rate vector to zero.**
4. **Initialize** $T = S_{BE}$ **and** $B = \varnothing$ **.**

Loop:

Do until ($T = \varnothing$ **)**

1. $\Delta_s = \min_{l \in \overline{B}} \left[\left(c_l - \sum_{s \in \overline{T}} R_{ls} x_s(t) \right) \Big/ \sum_{s \in T} R_{ls} \right]$
2. $x_s(t+1) = x_s(t) + \Delta_s \qquad \forall s \in T$
3. **Calculate new bottleneck links and update** B **and** $\overline{B}$ **.**
4. $\forall s \in T$ **; if** s **passes through any saturated link then** $T \Leftarrow T - \{s\}$

Output:

Communicate BE source rates to the corresponding nodes.

4 Simulation Results

In this section we examine the proposed flow control algorithm, listed above as Algorithm 1, for a typical NoC architecture. We have simulated a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links, each one has a fixed capacity of 1 Gbps. In our scenario, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bit long.

In order to simulate our scheme, some nodes are considered to have a GS data, such as Multimedia, etc., to be sent to a destination while other nodes, which maybe in the set of nodes with GS traffic, have a BE traffic to be sent. As stated in section 2, GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE traffics.

We are mainly interested in investigating the fairness properties among source rates. In order to investigate the rate allocation in the optimal sense, we solved problem (8) using CVX [16], which is a MATLAB-based software for disciplined

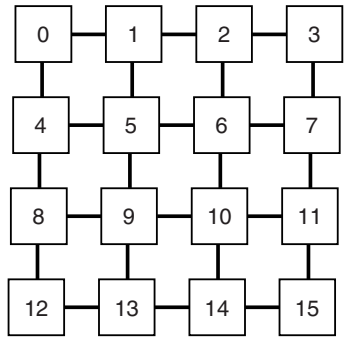


Fig. 1. Network topology

convex optimization problems. Optimal source rates, obtained by CVX, are shown in Fig. 2.

Source rates obtained from Algorithm 1 is depicted in Fig. 3. The main feature regarding Fig. 1 and Fig. 2 is that both yield equal values for the minimum source rate, i.e. 0.03 Gbps. The main difference is in the aggregate source rate which is greater for the result of Algorithm 1.

In order to compare the results of the proposed Max-Min fair flow control with other fairness criteria, we have accomplished rate allocation based on maximizing the sum of source rates, i.e. the so-called Rate-Sum Maximization, whose results are depicted in Fig. 4. Comparing Fig. 3 with Fig. 4, it's apparent that although Rate-Sum criterion aims at maximizing the sum of source rates, there is no guarantee for the rates of *weak* sources, i.e. sources which achieve very small rate. Indeed, in many scenarios with Rate-Sum criterion, such sources will earn as small as zero.

To compare the results of the three above mentioned schemes in more detail, we have considered five parameters featuring the merit of the different schemes as following:

1. least source rate
2. sum of source rates
3. Variance of source rates with respect to mean value.
4. Jain's fairness Index [17]
5. min-max ratio [17]

These parameters are presented in Table 1. Jain's fairness Index and max-min ratio, are defined by (12) and (13), respectively.

$$\text{Jain's Fairness Index} = \frac{\left(\sum_{s=1}^{S} x_s\right)^2}{S\sum_{s=1}^{S} x_s^2} \tag{12}$$

$$\text{Min-Max Ratio} = \frac{\min\limits_{s\in S} x_s}{\max\limits_{s\in S} x_s} \tag{13}$$

From table 1 we realize that rate allocation with Maximum Rate-Sum criteria, yield slightly greater rate-sum from Max-Min Fair criteria, i.e. Algorithm 1. However, as discussed above, Algorithm 1 guarantees that the rate allocation is max-min fair, and hence the minimum source rate wouldn't be greater with any other feasible rate allocation and hence rate allocation is carried out in favor of weak sources. On the contrary, Maximum Rate-Sum has no guarantee on such sources and as a result, the weakest source, has achieved his rate as low as zero. Another point which is worth mentioning is that similarity of the rate allocation to uniform rate allocation is further in the Max-Min scheme. To be more precise, we have calculated the variance of source rates in with respect to mean value of source rates in equilibrium. Table 1 shows that the variance of Max-Min rate allocation, obtained from Algorithm 1, is evidently less than that of Maximum Rate-Sum scheme, which in turn implies the inherent fairness in the Max-Min rate allocation.

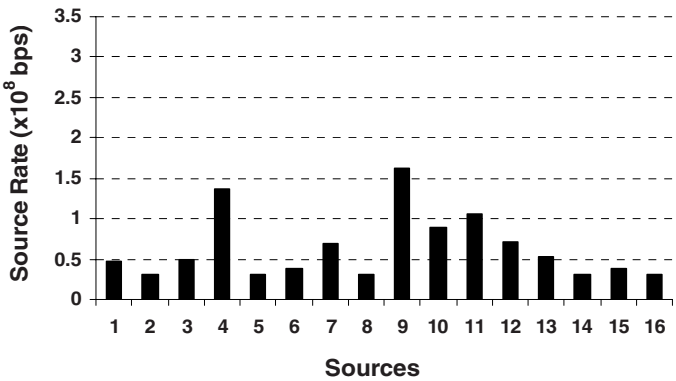


Fig. 2. Rate allocation using CVX results

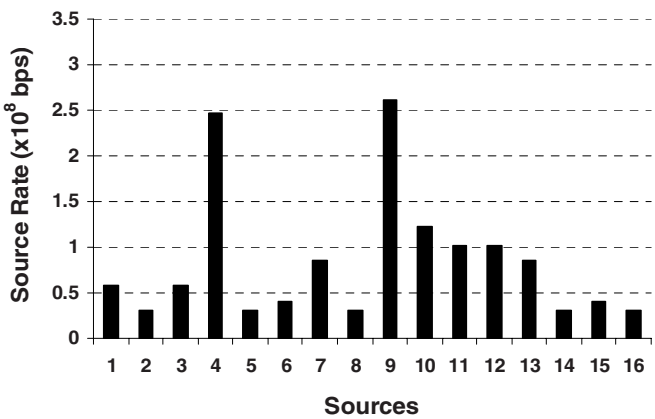


Fig. 3. Rate allocation using Algorithm 1

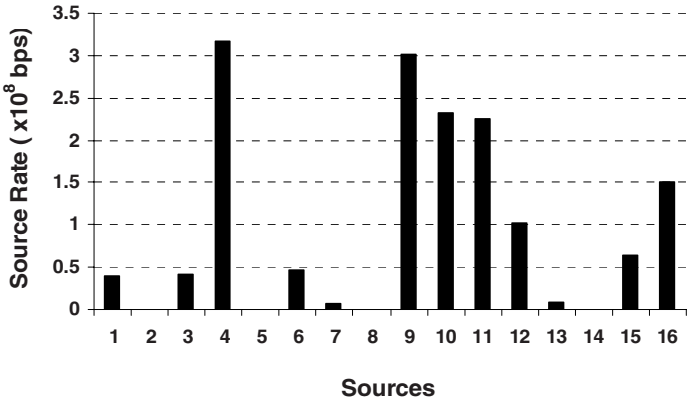


Fig. 4. Rate allocation using Rate-Sum Maximization

Table 1. Quantitative comparison between different rate allocation schemes

	Least Rate ($\times 10^8$ bps)	Sum of Source Rates ($\times 10^8$ bps)	Variance	Fairness Index	Min-max Ratio
Max-Min Fair (Mathematical Model)	0.310	10.079	0.1558	0.7181	0.1856
Max-Min Fair (Algorithm 1)	0.310	13.545	0.5004	0.5888	0.1148
Maximum Rate-Sum	0	15.349	1.1974	0.4346	0

5 Conclusion

In this paper we addressed the flow control problem for BE traffic in NoC systems. We considered two objectives. First, choosing source rates (IP loads) of BE traffics so that to accomplish flow control in response to demands at a reasonable level. Second, maintaining Max-Min fairness for all sources. Flow control was modeled as the solution to a simple algorithmic solution to an optimization problem. The algorithm can be implemented by a controller which admits a light communication and communication overhead. Finally, we compared the results of the proposed Max-Min fair flow control with Rate-Sum Maximization scheme based on several criteria such as Jain's fairness index, max-min ratio, and etc. comparison shows using the proposed flow control scheme, rate allocation has larger fairness index, which denotes that the aim of the proposed flow control scheme is to allocate NoC resources in a fair manner.

References

1. Benini, L., DeMicheli, G.: Networks on Chips: A New SoC Paradigm. Computer Magazine of the IEEE Computer Society 35(1), 70–78 (2002)
2. Dally, W.J., Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks. In: Design Automation Conference, pp. 684–689 (2001)

3. Cidon, I., Keidar, I.: Zooming in on Network on Chip Architectures. Technion Department of Electrical Engineering (2005)
4. Ogras, U.Y., Marculescu, R.: Prediction-based flow control for network-on-chip traffic. In: Proceedings of the Design Automation Conference (2006)
5. van den Brand, J.W., Ciordas, C., Goossens, K., Basten, T.: Congestion- Controlled Best-Effort Communication for Networks-on-Chip. In: Design, Automation and Test in Europe Conference and Exhibition, pp. 948–953 (2007)
6. Kelly, F.P., Maulloo, A., Tan, D.: Rate control for communication networks: Shadow prices, proportional fairness, and stability. J. Oper. Res. Soc. 49(3), 237–252 (1998)
7. Mascolo, S.: Classical control theory for congestion avoidance in high-speed internet. In: Decision and Control IEEE Conference, vol. 3, pp. 2709–2714 (1999)
8. Gu, Y., Wang, H.O., Hong, Y., Bushnell, L.G.: A predictive congestion control algorithm for high speed communication networks. In: American Control Conference, vol. 5, pp. 3779–3780 (2001)
9. Yang, C., Reddy, A.V.S.: A taxonomy for congestion control algorithms in packet switching networks. J. IEEE Network 9(4), 34–45 (1995)
10. Hu, J., Marculescu, R.: DyAD - smart routing for networks-on-chip. In: Design Automation Conference, pp. 260–263 (2004)
11. Talebi, M.S., Jafari, F., Khonsari, A., Yaghmae, M.H.: A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization. In: International Conference on Computational Science and its Applications, pp. 398–409 (2007)
12. Chiu, D.M., Jain, R.: Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. J. Computer Networks and ISDN Systems 17(1), 1–14 (1989)
13. Bertsekas, D.P., Gallager, R.: Data Networks. Prentice-Hall, Englewood Cliffs (1992)
14. Le Boudec, J.Y.: Rate adaptation, Congestion Control and Fairness: A Tutorial. Ecole Polytechnique Fédérale de Lausanne (EPFL) (2001)
15. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1999)
16. Grant, M., Boyd, S., Ye, Y.: CVX (Ver. 1.0RC3): Matlab Software for Disciplined Convex Programming, \url{http://www.stanford.edu/boyd/cvx}
17. Jain, R., Chiu, D., Hawe, W.: A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. DEC Research Report TR-301 (1984)

Paper 6

A Novel Flow Control Scheme for Best Effort Traffics in Network-on-Chip Based on Weighted Max-Min-Fairness

F. Jafari
M. H. Yaghmaee

In the Proceedings of International Symposium on Telecommunications (IST), pp. 458-463, Tehran, Iran, August 2008.

A Novel Flow Control Scheme for Best Effort Traffics in Network-on-Chip Based on Weighted Max-Min-Fairness

Fahimeh Jafari, Mohammad H. Yaghmaee
Ferdowsi University of Mashhad
Mashhad, Iran
fjafari@wali.um.ac.ir, hyaghmae@ferdowsi.um.ac.ir

***Abstract*- Network on Chip (NoC) has been proposed as an attractive alternative to traditional dedicated busses in order to achieve modularity and high performance in the future System-on-Chip (SoC) designs. Recently, end to end flow control has gained popularity in the design process of network-on-chip based SoCs. Where flow control is employed, fairness issues need to be considered as well. In fact, one of most difficult aspects of flow control is that of treating all sources fairly when it is necessary to turn traffic away from the network. In this paper, we propose a flow control scheme which admits Max-Min fairness criterion for all sources. In fact, we formulate Weighted Max-Min fairness criterion for the NoC architecture and presente implementation to be used as flow control mechanism.**

Keywords: Network on Chip; flow control; Weighted Max-Min fairness.

I. INTRODUCTION

The high level of system integration characterizing Multi-Processor Systems on Chip (MPSoCs) is raising the scalability issue for communication architectures. The problems emanating from the scalability issue in the MPSoCs have been remedied by the emergence of Network-on Chip (NoC) architectures [1]. Due to the rapid growth of the number of processing elements in NoCs [2], employing efficient policies for flow control has become an inevitable subject in the design of NoCs to provide the required Quality of Service (QoS). A NoC must have network level flow control in order to avoid congestion in the bottleneck links.

Recently, QoS provisioning in NoC's environment has attracted many researchers and currently is the focus of many literatures in NoC research community. NoCs are expected to serve as multimedia servers and are required to carry both Best Effort (BE) and Guaranteed Service (GS) traffics. It's trivial that such a networked architecture with data services should have some policies to avoid congestion. Congestion Control in data networks is known as a widely-studied issue over the past two decades. However, it is still a novel issue in NoC and to the best of our knowledge only a few works have been carried out in this field.

Many strategies for flow control have been proposed for off-chip networks, e.g. data networks, etc. [3-5]. On-chip networks pose different challenges. For instance, in off-chip environments, to overcome congestion in links, packet dropping is allowed. On the contrary, reliability of on-chip wires makes NoCs a loss-less environment.

So far, several works have focused on this issue for NoC architectures. In [6], a prediction-based flow-control strategy for on-chip networks has been proposed in which each router predicts the buffer occupancy to sense congestion. In [7] link utilization is used as a congestion measure and a Model Prediction-based Controller (MPC), determines source rates. Dyad [8] controls the congestion by using adaptive routing when the NoC faces congestion.

Where flow control is employed, fairness issues need to be considered as well [9]. In fact, one of the most difficult aspects of flow control is to choose a policy to accommodate a fair rate allocation. All of the abovementioned studies only regarded the flow control by taking into account the constraints of the system. To the best of our knowledge no policy to maintain fairness among sources has been chosen.

Different flow control approaches can be classified with respect to the fairness criteria, in favor of which rate allocation is done. One of the famous forms of fairness criterion is Max-Min fairness, which has been discussed in earlier literature and described clearly in [10]. Our main contribution in this paper is to present a flow control scheme for Best Effort traffic in NoC which satisfies Weighted Max-Min fairness criterion through the analysis of mathematical model and simulation. Our framework is mainly adopted from the seminal work [10] which presents a basic Max-Min fairness. In this paper, we formulate Weighted Max-Min problem for the NoC architecture.

The organization of the paper is as follows. In Section II we present the system model and flow control problem. In section III we present an iterative algorithm as the solution to the flow control optimization problem. Section IV

978-1-4244-2751-2/08/$25.00 ©2008 IEEE

presents the simulation results and discussion about them. Finally, section IV concludes the paper.

II. System Model and Flow Control Problem

We consider a NoC architecture which is based on a two dimensional mesh topology and wormhole routing. We also assume that the NoC architecture is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing [2].

We model the flow control in NoC as the solution to an optimization problem. For the sake of convenience, we turn the aforementioned NoC architecture into a mathematically modeled network. In this respect, we consider NoC as a network with a set of bidirectional links L and a set of sources S. Each source $s \in S$ consists of processing elements, routers and input/output ports. Each link $l \in L$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of c_l packets/sec. We denote the set of sources that share link l by $S(l)$. Similarly, the set of links that source s passes through, is denoted by $L(s)$. By definition, $l \in L(s)$ if and only if $s \in S(l)$.

As discussed in section 1, there are two types of traffic in a NoC: Guaranteed Service (GS) and Best Effort (BE) traffic. For notational convenience, we divide S into two parts, each one representing sources with the same kind of traffic. In this respect, we denote the set of sources with BE and GS traffic by S_{BE} and S_{GS}, respectively. Each link l is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and BE sources benefit from the remainder.

Our objective is to choose source rates with BE traffic so that to maximize the type of weighted α-Fair function in which $\alpha = \infty$. Weighted α-Fair function is define as below [11]:

$$U(x,\alpha,w) = \begin{cases} w\dfrac{x^{1-\alpha}}{1-\alpha} & \alpha \neq 1 \\ w\ln x & \alpha = 1 \end{cases} \tag{1}$$

where $\alpha > 0$ is a parameter. Therefore we define our flow control problem as below:

$$\lim_{\alpha\to\infty} \max_{x_s} \sum_s w_s \frac{x_s^{1-\alpha}}{1-\alpha} \tag{2}$$

subject to:

$$\sum_{s\in S_{BE}(l)} x_s + \sum_{s\in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{3}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{4}$$

Optimization variables are BE source rates, i.e. $(x_s,\ s \in S_{BE})$. The constraint (3) states that the sum of BE source rates passing thorough link l cannot exceed its free capacity, i.e. the portion of c_l which has not been allocated to GS traffic.

Problem (2) is a convex optimization problem with linear constraints. Hence it admits a unique maximizer [12]. Treating problem (2) using such an extreme case is not disobedient. However, the following theorem states that it can be reduced to a well-known type of disciplined optimization problem known as Weighted Max-Min problem. The following definition states the formal definition of WMMF.

THEOREM 1: *α-Fair maximization problem for $\alpha = \infty$ reduces to weighted max-min optimization problem, as below* [11]:

$$\max_x \min_{s\in S} w_s x_s \tag{5}$$

subject to:

$$\sum_{s\in S_{BE}(l)} x_s + \sum_{s\in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{6}$$

$$x_s \ge 0 \qquad \forall s \in S_{BE} \tag{7}$$

For notational convenience, we define:

$$u = \min_{s\in S} w_s x_s$$

$$\hat{c}_l = c_l - \sum_{s\in S_{GS}(l)} x_s$$

To solve the above problem, it should be converted so as to be in the form of disciplined optimization problems [13] as follows:

$$\max u \tag{8}$$

subject to:

$$u \le w_s x_s \quad \forall s \in S \tag{9}$$

$$\sum_{s\in S_{BE}(l)} x_s \le \hat{c}_l \qquad \forall l \in L \tag{10}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{11}$$

Weighted Max-Min optimization problem is a widely-studied problem formulation in the design of data networks. Weighted Max-Min problem has an important property which discriminates it from the others. The optimal solution to the weighted max-min problem exists, a specific type of fairness characteristic know as Weighted Max-Min Fairness (WMMF) is admitted which will formally be defined in the following.

DEFINITION 1: (Weighted Max-Min Fairness [14]). *Given some positive constants w_i (called the "weights"), $x = (x_s, s \in S)$ is "Weighted-Max-Min Fair", if and only if increasing one component x_s must be at the expense of decreasing some other component x_t such as $x_t/w_t \le x_s/w_s$.*

If we assume $w_s = 1 \ \forall s \in S$, WMMF will be known as Max-Min Fairness (MMF) which will formally be defined in the following.

DEFINITION 2: (Max-Min Fairness [14]). *A feasible rate allocation $x = (x_s, s \in S)$ is said to be "Max-Min Fair" (MMF) if and only if an increase of any rate within the*

domain of feasible allocations must be at the cost of a decrease of some already smaller rate. Formally, for any other feasible allocation y, *if* $y_s > x_s$ *then there must exist some* s' *such that* $x_{s'} \le x_s$ *and* $y_{s'} < x_{s'}$.

Depending on the network topology, a max-min fair allocation may or may not exist. However, if it exists, it is unique (see [14] for proof). In what follows the condition under which the Max-Min rate allocation exists will be stated. Before we proceed to this condition, we define the concept of bottleneck link.

DEFINITION 3: (Bottleneck Link [14]), *With our system model above, we say that link l is a bottleneck for source s if and only if*

1. *link l is saturated:* $\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s = c_l$
2. *source s on link l has the maximum rate among all sources using link l.*

Intuitively, a bottleneck link for source s is a link which limits x_s.

THEOREM 2: *A max-min fair rate allocation exists if and only if every source has a bottleneck link.*

Proof: See [14] for proof.

Any discussion of the performance of a rate allocation scheme must address the issue of fairness, since there exist situations where a given scheme might maximize network throughput, for example, while denying access for some users or sources. Max-Min fairness is one the significant fairness criteria. Crudely speaking, a set of rates is max-min fair if no rate can be increased without simultaneously decreasing another rate which is already smaller. In a network with a single bottleneck link, max-min fairness simply means that flows passing through the bottleneck link would have equal rates.

The most famous and simplest algorithm to solve the max-min problem is the well-known Progressive Filling Algorithm [10]. We would like to modify the progressive filling algorithm as an iterative solution to the weighted max-min problem (8) for our system model. We finally would like to utilize it as BE flow control mechanism in NoC. The modified version of the progressive filling as a BE flow control mechanism is listed below as algorithm 1.

III. Weighted Max-Min-Fairness Algorithm

Theorem 2 is particularly useful in deriving a practical method for obtaining a max-min fair allocation, called "progressive filling". The idea is as follows: rates of all flows are increased at the same pace, until one or more links are saturated. The rates of flows passing through saturated links are then frozen, and the other flows continue to increase rates. All the sources that are frozen have a bottleneck link. This is because they use a saturated link, and all other sources using the saturated link are frozen at the same time, or were frozen before, thus have a smaller or equal rate. The process is repeated until all rates are frozen. Lastly, when the process terminates, all sources have been frozen at some time and thus have a bottleneck link. Using Theorem 2, the allocation is max-min fair.

Algorithm 1: Weighted Max-Min Fair (WMMF) Flow Control Algorithm for BE in NoC

Initialization:

1. Initialize $\hat{c}_l$ of all links.
2. Define:
   a. T as the set of sources not passing through any saturated link.
   b. B as the set of saturated links.
3. Set source rate vector to zero.
4. Initialize $T = S_{BE}$ and $B = \varnothing$.

Loop:

Do until ($T = \varnothing$)

1. $\Delta_s = \min_{l \in (L-B)} \left[\left(c_l - \sum_{s \in (S_{BE} - T)} R_{ls} x_s(t) \right) \Big/ \sum_{s \in T} w_s R_{ls} \right]$
2. $x_s(t+1) = x_s(t) + w_s \Delta_s \quad \forall s \in T$
3. Calculate new bottleneck links and update B.
4. $\forall s \in T$; if s passes through any saturated link then $T \Leftarrow T - \{s\}$

Output:

Communicate BE source rates to the corresponding nodes.

In the sequel, we modify the progressive filling algorithm as an iterative solution to the weighted max-min problem (8) for our system model and based on this algorithmic solution, we present a flow control scheme for BE traffic in NoC systems.

Thus, the aforementioned algorithm can be employed to control the flow of BE traffic in the NoC. The iterative algorithm can be addressed in distributed scenario. However, due to well-formed structure of the NoC, we focus on a centralized scheme; we use a controller like [7] to be mounted in the NoC to implement the above algorithm. The necessary requirement of such a controller is the ability to accommodate simple mathematical operations and the allocation of few wires to communicate flow control information to nodes with a light GS load.

IV. Simulation Results

In this section we examine the proposed flow control algorithms for a typical NoC architecture. In our scenario, we have used a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. In our scheme, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bit long.

In order to simulate our scheme, some nodes are considered to have a GS data, such as Multimedia, etc., to be sent to a destination while other nodes, which maybe in the set of nodes with GS traffic, have a BE traffic to be sent. As stated in section 2, GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE traffics.

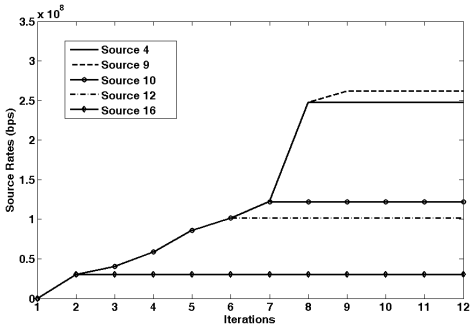


Fig. 1 Source Rates vs. Iteration Steps for Max-Min

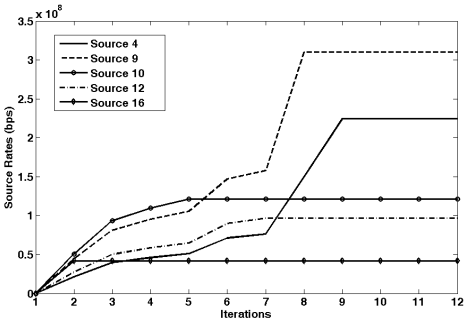


Fig. 2 Source Rates vs. Iterations for Weighted Max-Min with w_1

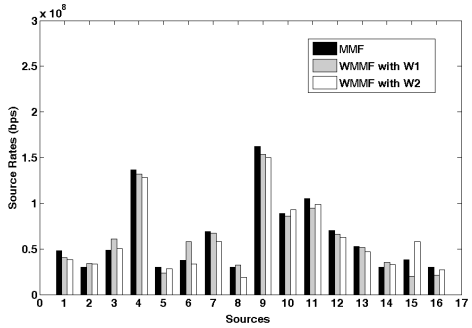


Fig. 3 Comparison of Max-Min, Weighted Max-Min with w_1 and Weighted Max-Min with w_2

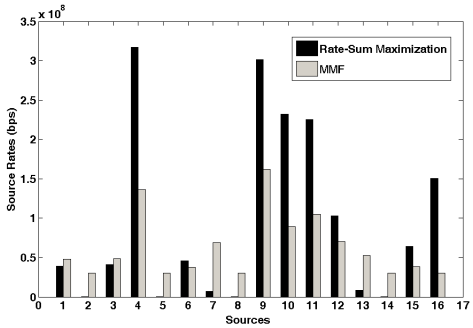


Fig. 4 Comparison between Rate-Sum and Max-Min

We are mainly interested in investigating the fairness properties among source rates.

A. WMMF with Various Weights

We obtained source rates using proposed algorithm in MATLAB. Rate variation versus iteration steps for both MMF and WMMF (with weight vector w_1) flow control schemes are shown in Fig. 1 and Fig. 2, respectively.

We solved problem (8) with different weight vectors such as w_2 and w_3 (due to space limit, values of weight vectors have been omitted) to control the priority of resource allocation. Such weighting factors can be appropriately derived so that to designate the network resources (link capacities) in favor of source priorities.

For the sake of convenience in comparing these schemes, steady state source rates for all sources are depicted in Fig. 3. It is clear from Fig. 3 that with different weight vectors, priorities of sources vary significantly and as a result, WMMF lead to great differences in rate allocations.

B. WMMF Fairness Metrics

In order to compare the results of the proposed Max-Min fair flow control with other fairness criteria, we have used rate allocation based on maximizing the sum of source rates, i.e. the so-called Rate-Sum Maximization. For comparing the two schemes, steady state source rates for all sources are depicted in Fig. 4. Comparing Rate-Sum and Max-Min in Fig. 4, it's evident that although Rate-Sum criterion aims to maximize the sum of source rates, there is no guarantee for the rates of *weak* sources, i.e. ones which achieve very small rate. Indeed, in many scenarios with Rate-Sum flow control, such sources will earn as small as zero. On the other hand, the weakest source in Max-Min scenario earns about 0.3 Gbps.

To compare the results of the above mentioned schemes in more detail, we have considered four parameters featuring the merit of the different schemes as following:

1. Least source rate
2. Variance of source rates with respect to mean value.
3. Jain's fairness Index (JFI) [15]
4. min-max ratio [15]

These parameters are presented in Fig. 5 and Fig 6. Jain's fairness Index and max-min ratio, are defined by (12) and (13), respectively.

$$\text{Jain's Fairness Index} = \frac{\left(\sum_{s=1}^{N} x_s\right)^2}{N\sum_{s=1}^{N} x_s^2} \tag{12}$$

$$\text{Min-Max Ratio} = \frac{\min\limits_{s\in S} x_s}{\max\limits_{s\in S} x_s} \tag{13}$$

Amongst the aforementioned parameters, Jain's Fairness Index, Min-Max Ratio and Variance of source rates for

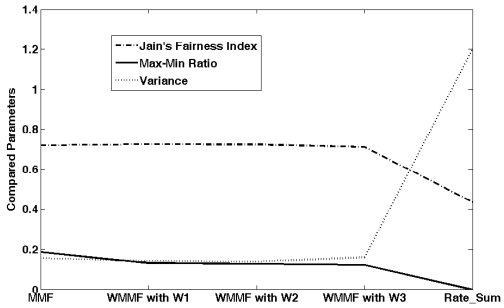


Fig. 5 Different parameters for Different scenarios

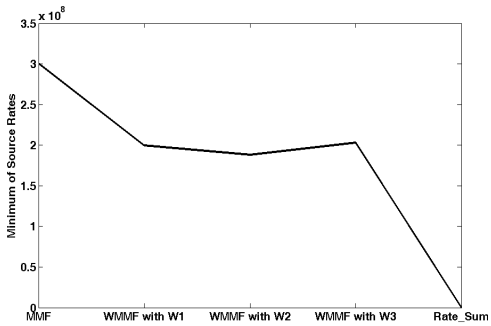


Fig. 6 Least source rate for Different scenarios

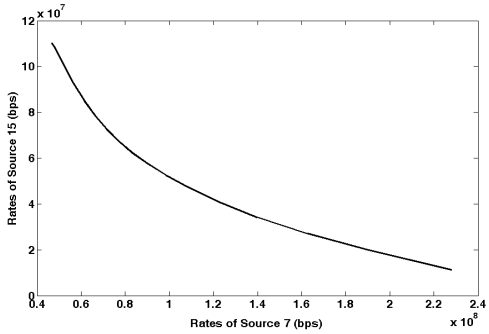


Fig. 7 Rate region for x_7 and x_{10}

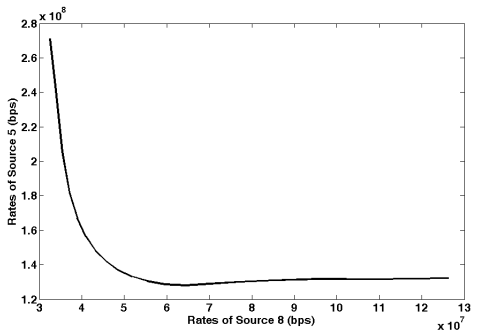


Fig. 8 Rate region for x_8 and x_5

MMF, WMMF (with three different weights) and Rate-Sum schemes are depicted in Fig. 5. It is apparent that using MMF and WMMF schemes, the variance of source rates are considerably less than Rate-Sum, which denotes the intrinsic fairness in these mechanisms with respect to Rate-Sum mechanism. Smaller variance results in the larger Min-Max Ratio and JFI; therefore MMF and WMMF schemes have greater Min-Max Ratio and JFI.

To have a better insight about the influence of weights on the MMF scheme, the rate of the weakest source for the aforementioned scenarios is shown in Fig. 6. It's apparent that with pure Max-Min scheme, the weakest user obtains the largest rate among other schemes. As the variance of weight elements increases, the weakest source's rate falls rapidly. Finally, in the Rate-Sum, the rate of the weakest source is approximately zero.

C. Rate-Region for WMMF

In order to analyze the effect of the weighting scheme in more detail, we introduce the concept of *Rate Region* for the flow control we considered in this paper. We think of a Rate Region as a region of all possible rate tuples $(x_1, x_2, ..., x_S)$ that satisfy link capacity constraints, i.e.

$$\left\{ (x_1, x_2, ..., x_S) \in \mathbb{R}^S \left| \sum_{s \in S(l)} x_s \le c_l; \forall l \in L \right. \right\}$$

Rate region for two Weighted Max-Min scenarios are depicted in Fig. 7 and Fig. 8. We briefly discuss about some heuristic insights that can be obtained from these figures.

For the sake of simplicity in representing such a region, we fix the weight of all nodes to 1 and assume the weights of two nodes, say nodes i and j, are set to w and 2-w, respectively. Then, by sweeping w over [0,2] interval, we study the effect of the rate allocation on the rate of node i and j, e.g. x_i and x_j. In this respect, rate region can be depicted efficiently using a two-dimensional curve whose axes are x_i and x_j. In Fig. 7, we have i=7 and j=15. It is apparent that by varying the weight from 0 to 2, x_{15} would vary from 0 to 0.12 Gbps, however, for the source 7 such a variation is limited to 0.04 to 0.22 Gbps. This means that in the worst case source 7 would obtain a considerably larger weight with regard to source 15. In fact, source 15 is more sensitive to weight selection than source 7. Setting i=8 and j=5 yields the rate region depicted in Fig. 8. A similar discussion also holds and we conclude that source 8 is more sensitive to weighting, because the range over which its rates varies is much larger.

Another advantage of such rate regions, which is worth discussing, might be the selection of efficient weighting factors which suits the demands and constraints of the underlying system. Crudely speaking, we may study such a S-dimensional rate region by evaluating a number of simpler two-dimensional rate-regions, as with above, and then determine source pairs which are highly-sensitive to weight selection. Regarding such regions, based on the rate demands of sources, we can obtain the appropriate point of the region, thereby the corresponding weights.

V. CONCLUSION

In this paper we addressed the flow control problem for BE traffic in NoC systems. We considered two objectives. First, choosing source rates (IP loads) of BE traffics so that to accomplish flow control in response to demands at a reasonable level. Second, maintaining Weighted Max-Min fairness for all sources. Flow control was modeled as the solution to a simple algorithmic solution to an optimization problem. The algorithm can be implemented by a controller which admits a light communication and communication overhead.

REFRENCES

[1] L. Benini, and G. DeMicheli, "Networks on Chips: A New SoC Paradigm." *Computer*, 2002, vol. 35, no. 1, pp. 70-78.

[2] W. J. Dally, and B. Towles, "Route Packets, Not Wires: On- Chip Interconnection Networks," in *Proceedings of Design Automation Conference*, 2001, pp. 684-689.

[3] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness, and stability." *Operational Research Society*, 1998, vol. 49, no. 3, pp. 237-252.

[4] Y. Gu, H. O. Wang, and L. G. Yiguang Hong Bushnell, "A predictive congestion control algorithm for high speed communication networks," in *Proceedings of American Control Conference*, 2001, vol. 5, pp. 3779-3780.

[5] C. Yang, and A. V. S. Reddy, "A taxonomy for congestion control algorithms in packet switching networks," *IEEE Network*, 1995, vol. 9, no. 4, pp. 34-45.

[6] U. Y. Ogras, and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," in *Proceedings of Design Automation Conference*, 2006.

[7] J. W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip," *Design, Automation and Test in Europe Conference*, 2007, pp. 948-953.

[8] J. Hu, and R. Marculescu, "DyAD - smart routing for networks-onchip," in *Proceedings of Design Automation Conference*, 2004, pp. 260-263.

[9] I. Cidon and I. Keidar, "Zooming in on Network on Chip Architectures." *Technion Department of Electrical Engineering*, 2005.

[10] D. Bertsekas, and R. Gallager, *Data Networks*, Prentice-Hall, 1992.

[11] J. Mo, J. Walrand, "Fair End-to-End Window-Based Congestion Control," *IEEE/ACM Transactions on Networking*, 2000, vol. 8, no. 5, pp. 556-567.

[12] S. H. Low, and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, 1999, vol. 7, no. 6, pp. 861-874.

[13] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[14] J. Y. Le Boudec, "Rate adaptation, Congestion Control and Fairness: A Tutorial." Ecole Polytechnique Federale de Lausanne (EPFL), 2001.

[15] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness And Discrimination For Resource Allocation In Shared Computer Systems", DEC Research Report TR-301, 1984.

Paper 7

Throughput-Fairness Tradeoff in Best Effort Flow Control for On-chip Architectures

F. Jafari
M. S. Talebi
M. H. Yaghmaee
A.Khonsari

In the Proceedings of the International Workshop on Performance Modeling, Evaluation, and Optimization of Ubiquitous Computing and Networked Systems (PMEO UCNS), in conjunction with the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rome, Italy, May 2009.

Throughput-Fairness Tradeoff in Best Effort Flow Control for On-Chip Architectures

Fahimeh Jafari*†, Mohammad S. Talebi†, Mohammad H. Yaghmaee*, Ahmad Khonsari‡† and Mohamed Ould-Khaoua§¶
* *Ferdowsi University of Mashhad, Mashhad, Iran*
† *School of Computer Science, IPM, Tehran, Iran*
‡ *ECE Department, University of Tehran, Tehran, Iran*
§ *Department of Electrical and Computer Engineering, Sultan Qaboos University, Oman*
¶ *Department of Computing Science, University of Glasgow*
Emails: jafari@ipm.ir, mstalebi@ipm.ir, hyaghmae@ferdowsi.um.ac.ir, ak@ipm.ir, mohamed@dcs.gla.ac.uk

Abstract

We consider two flow control schemes for Best Effort traffic in on-chip architectures, which can be deemed as the solutions to the boundary extremes of a class of utility maximization problem. At one extreme, we consider the so-called Rate-Sum flow control scheme, which aims at improving the performance of the underlying system by roughly maximizing throughput while satisfying capacity constraints. At the other extreme, we deem the Max-Min flow control, whose concern is to maintain Max-Min fairness in rate allocation by fairly sacrificing the throughput. We then elaborate our argument through a weighting mechanism in order to achieve a balance between the orthogonal goals of performance and fairness. Moreover, we investigate the implementation facets of the presented flow control schemes in on-chip architectures. Finally, we validate the proposed flow control schemes and the subsequent arguments through extensive simulation experiments.

1. Introduction

With the advance of the semiconductor technology, the enormous number of transistors available on a single chip allows designers to integrate dozens of IP (Intellectual Property) blocks together with large amounts of embedded memory. Such IPs may be CPU or DSP cores, video stream processors, high-bandwidth I/O, etc. Dedicated links may lead to an irregular architecture which is difficult to reuse. Also, shared-medium busses do not scale well, and do not fully utilize potentially available bandwidth. As the feature sizes shrink, and the overall chip size relatively increases, the interconnects start behaving as lossy transmission lines. Line delays have become very long as compared to gate delays causing synchronization problems between cores. This trend only worsens as the clock frequencies increase and the features sizes decrease.

One solution to these problems is to treat systems on a chip implemented using the Networks-on-Chip (NoC) paradigm. Multiple concurrent connections provide much higher bandwidth in Networks. Also, regularity enables design modularity, which in turn provides a standard interface for easier component reuse and better interoperability. Overall performance and scalability increase since the networking resources are shared [1]. Due to the rapid growth of the number of processing elements in NoCs [2], employing efficient policies for flow control has become an inevitable subject in the design of NoCs to provide the required Quality of Service (QoS). A NoC must have network level flow control in order to avoid congestion in the bottleneck links.

Recently, QoS provisioning in NoC's environment has attracted many researchers and currently is the focus of many literatures in NoC research community. NoCs are expected to serve as multimedia servers and are required to carry both Best Effort (BE) and Guaranteed Service (GS) traffics. It's trivial that such a networked architecture with data services should have some policies to avoid congestion. Congestion Control in data networks is known as a widely-studied issue over the past two decades. However, it is still a novel issue in NoC and to the best of our knowledge only a few works have been carried out in this field.

In this paper, we focus on the flow control for BE traffic as the solution to an optimization problem. In our previous work [3], we have modeled desired BE source rates as the solution to a utility-based optimization problem with a general form utility function and solved the problem using Newton method. In [4], we also considered this issue via Rate-Sum optimization problem and used a different approach to solve it. In this paper, we mainly focus on the two extreme cases of the utility optimization approach which lead to different performance and fairness properties. In the first view, in this paper we present two flow control mechanisms for BE traffic in NoC which are derived for the extreme cases of utility definition in [3]. These flow control schemes exhibit different fairness and performance

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

properties. In another view, investigating and balancing the tradeoff between such conflicting properties is the other contribution of ours in this paper.

The organization of the paper is as follows. In Section 2 we will briefly review the most significant works on this subject. In Section 3, we present the system model and problem formulation. In Section 4 we focus on the Rate-Sum problem and propose a flow control for BE as an iterative solution to it. In Section 5 we consider the Max-Min problem along with the concept of the Max-Min fairness and present another BE flow control mechanism. Section 6 is devoted to the discussion about the fairness and performance tradeoff for the presented flow control mechanisms and presents a remedy to balance between them. Section 7 presents the simulation results and discussion about them. Finally, the Section 8 concludes the paper.

2. Related Works

In this section, we briefly review the most significant works focused on this issue.

Flow control for data networks is a widely-studied issue [5]-[7]. A wide variety of flow control mechanisms in data network belongs to the class of End-to-End flow control schemes, like TCP/IP, which mainly act based on the window-based protocols. In this method, sent packets are subject to loss and the network must aim to provide an acknowledgment mechanism. On the other hand, On-chip networks pose different challenges. The reliability of on-chip wires and more effective link-level flow-control makes NoC a loss-less environment. Therefore, there is no need to utilize acknowledgment mechanisms and we face to a slightly different concept of flow control.

So far, several works have focused on this issue for NoC architectures. Dyad [8] controls the congestion by using adaptive routing when the NoC faces congestion. However this method can not guarantee that congestion is solved (i.e. the alternative paths might also be congested). In [9], a prediction-based flow-control strategy for on-chip networks is proposed in which each router predicts the buffer occupancy to sense congestion. In [10] link utilization is used as a congestion measure and a Model Prediction-based Controller (MPC), determines source rates. The authors in [10] state that their work is more complete than [9] because in [9] the router buffer filling information is used for toggling the sources while their approach allows both toggling and fluent control of loads offered by IPs.

To the best of our knowledge, none of the aforementioned works have dealt with the problem through utility optimization approach. As mentioned in [11], our approach has little control overhead than [10] because in [10] the link utilization measurements are periodically performed by hardware probes and are transported to a controller by GS connections while in our approach control packets are sent to the controller only when a flow enters to the network or exits from it. Besides, since we do not need any hardware probes, costs of hardware implementation is reduced.

3. System Model and Flow Control Problem

We consider a NoC architecture with wormhole switching. In wormhole switching networks, each packet is divided into a sequence of *flits* which are transmitted over physical links one by one in a pipeline fashion. A hop-to-hop credit mechanism guarantees that a flit is transmitted only when the receiving port has free space in its input buffer. We also assume that the NoC architecture is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing [2].

We model the flow control in NoC as the solution to a utility-based optimization problem. We turn the aforementioned NoC architecture into a mathematically modeled network, as in [12]. In this respect, we consider NoC as a network with a set of bidirectional links $\mathcal{L} = \{1, 2, \ldots, L\}$ and a set of sources $\mathcal{S} = \{1, 2, \ldots, S\}$. A source consists of Processing Elements (PEs), Routers and Input/Output ports. Also, each link $l \in \mathcal{L}$ has a fixed capacity of c_l bits/sec and is a set of wires and channels that are responsible for connecting different parts of the NoC. We denote the set of sources that share link l by $\mathcal{S}(l)$. Similarly, the set of links that source s passes through, is denoted by $\mathcal{L}(s)$. By definition, $s \in \mathcal{S}(l)$ if and only if $l \in \mathcal{L}(s)$.

As presented before, two classes of traffic are considered in a NoC: Guaranteed Service (GS) and Best Effort (BE). For notational convenience, we represent BE and GS traffic rates by x_s and y_s, respectively. The two classes flow over a link by sharing its capacity as following: GS traffic will obtain the required amount of link capacity and BE traffic can benefit from the remainder.

Our objective is to choose source rates with BE traffic so as to maximize the sum of utilities of BE sources. We assume that source s when transmitting BE packets at rate x_s bps, achieves a utility equal to $U_s(x_s)$. Thus, the optimization problem can be formulated as below:

$$\max_{x_s} \sum_{s \in \mathcal{S}} U_s(x_s) \tag{1}$$

subject to:

$$\sum_{s \in \mathcal{S}(l)} x_s + y_s \leq c_l; \qquad \forall l \in \mathcal{L} \tag{2}$$

$$x_s > 0; \qquad \forall s \in \mathcal{S} \tag{3}$$

where optimization variables are BE rates, which in vector form are denoted by $\mathbf{x} = (x_s, s \in \mathcal{S})$ and belong to $\mathfrak{R}_+^S$. ($\mathfrak{R}_+^S$ denotes nonnegative real).

The constraint (2) simply states that the sum of BE rates passing through link l cannot exceed its free capacity, i.e. the portion of c_l which hasn't been dedicated to GS

traffic. Constraint (2) is equivalent to the case in which the maximum capacity of links can be used. Such an assumption may not hold in general unless the buffering space at each router tends to infinity. Although, such an assumption may seem restrictive, it will be useful in the sense that it yields the upper bound of the achievable source rate. Moreover, we argue that one of the advantages of applying a flow control mechanism is to efficiently allocate source rates so as to better utilize the maximum capacity of links. Therefore, using flow control, we can avoid buffer shortage by efficiently limiting the injection rates of nodes, and therefore, we shift from a buffer-restricted regime towards a link capacity-limited regime.

Problem (1) is such a general form is a Utility Maximization Problem. Based on the convex optimization theory, for such a problem to have a unique optimal point, U_s should be positive, concave and strictly increasing [12]. Several candidates for U_s exist for which the aforementioned conditions hold. Amongst them, presumably α-Fair functions are the most significant ones as they have nice properties in terms of economically fair behavior.

In this paper, we consider problem (1) with the class of α-Fair utility functions, defined as below [13]:

$$U(x,\alpha) = \begin{cases} \frac{x^{1-\alpha}}{1-\alpha} & \alpha \neq 1 \\ \ln x & \alpha = 1 \end{cases} \tag{4}$$

where $\alpha > 0$ is a parameter. With the aforementioned choice of utility function, problem (1) is a convex optimization problem with linear constraints. Hence it admits a unique maximizer [14][15]. For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in \mathcal{S}(l)} y_s \tag{5}$$

Although $\hat{c}_l$ denotes the usable link capacity, with a slight abuse of definition, hereafter we will refer to $\hat{c}_l$ as the link capacity. Moreover, similar to BE rate vector, we represent link capacity vector as $\hat{\mathbf{c}} = (\hat{c}_l, l \in \mathcal{L})$. Finally, to avoid confusing with summations indices, we define Routing matrix as $\mathbf{R} = [R_{ls}]_{L \times S}$, where R_{ls} is defined as:

$$R_{ls} = \begin{cases} 1 & \text{if } l \in \mathcal{L}(s) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In this paper, we mainly focus on the two extreme cases; i.e. $\alpha = 0$ and $\alpha \to \infty$ which lead to performance tradeoffs in rate allocation. In the sequel, we first focus on the case of $\alpha = 0$ for which problem (1) reduces to the so-called Rate-Sum Maximization and then investigate the case of $\alpha \to \infty$ for which problem (1) converts to the so-called Max-Min problem.

4. Rate-Sum Flow Control

In this section, we consider the case of $\alpha = 0$ and solve the accordingly-derived Rate-Sum Maximization problem in an iterative manner. We finally present a flow control algorithm for BE source rates based on the iterative solution.

Algorithm 1: Maximum Rate-Sum Flow Control Algorithm for BE

Initialization:

1. Initialize $\hat{c}_l$ of all links.
2. Set source rate vector to zero.
3. Specify an appropriate value for ϵ.

Loop:

Do until $(\max_{s \in \mathcal{S}} \mid x_s(t+1) - x_s(t) \mid < \epsilon)$

1. $\forall s \in \mathcal{S}$: Compute new source rate: $\mathbf{x}(t+1) = \mathbf{x}(t) + \gamma(t)\mathbf{u}(\mathbf{x}(t))$

where $\gamma(t)$ can be selected as $\gamma(t) = \frac{a}{b+t}$ and

$$\mathbf{u}(\mathbf{x}(t)) = \begin{cases} \mathbf{1} & \sum_{s \in \mathcal{S}(l)} x_s(t) \leq \hat{c}_l, \quad \forall l \\ -\mathbf{R}^T \mathbf{e}_{l'} & \sum_{s \in \mathcal{S}(l)} x_s(t) > \hat{c}_l, \quad \exists l' \end{cases}$$

Output:

Communicate BE source rates to the corresponding nodes.

Regarding problem (1), it's apparent that by substituting $\alpha = 0$, problem (1) can be rewritten as:

$$\max_{x_s} \sum_{s \in \mathcal{S}} x_s \tag{7}$$

subject to:

$$\mathbf{R}\mathbf{x} \leq \hat{\mathbf{c}} \tag{8}$$

$$x_s > 0; \qquad s \in \mathcal{S} \tag{9}$$

We will solve this problem using Gradient Method for constrained problems [14]. The Projected Gradient Method for constrained problems is very similar to the original one which only applies to unconstrained problems [14][15]. Therefore, the update equation to solve problem (7) is given by:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \gamma(t)\mathbf{u}(\mathbf{x}(t)) \tag{10}$$

where $\gamma(t)$ is a diminishing step-size rule which satisfies specific conditions [14]. One typical example is $\gamma(t) = a/(b+t)$, where $a > 0$ and $b \geq 0$, which we have used in this paper. $\mathbf{u}(\mathbf{x}(t))$ is given by:

$$\mathbf{u}(\mathbf{x}(t)) = \begin{cases} \mathbf{1} & \sum_{s \in \mathcal{S}(l)} x_s(t) \leq \hat{c}_l, \quad \forall l \\ -\mathbf{R}^T \mathbf{e}_{l'} & \sum_{s \in \mathcal{S}(l)} x_s(t) > \hat{c}_l, \quad \exists l' \end{cases} \tag{11}$$

where $\mathbf{e}_{l'}$ is the l'-th unit vector of $\Re^L$ space which is zero in all entries except the l'-th at which it is 1.

Equations (10) and (11) together form an iterative algorithm to solve Rate-Sum Maximization problem (7). Algorithmic realization of the proposed flow control algorithm for BE traffic is listed as Algorithm 1.

5. Max-Min Flow Control Problem

In this section, our focus is on the case of $\alpha \to \infty$. First, we show that the corresponding problem can be construed as a Max-Min problem which can be solved using *Progressive Filling* algorithm [16]. Finally, Based on this algorithm, we present a flow control algorithm for BE source rates. Considering problem (1), it's apparent that when α tends to the infinity, we have

$$\max_{x_s} \lim_{\alpha \to \infty} \sum_{s \in \mathcal{S}} \frac{x_s^{1-\alpha}}{1-\alpha} \tag{12}$$

subject to:

$$\mathbf{Rx} \leq \hat{\mathbf{c}} \tag{13}$$

$$x_s > 0; \qquad s \in \mathcal{S} \tag{14}$$

Problem (12) in such an extreme case is disobedient. However, the following theorem states that it can be reduced to the well-known Max-Min optimization problem.

Theorem 1: The maximization problem (12) reduces to the Max-Min optimization problem, as below

$$\max_{x_s} \min_{s \in \mathcal{S}} x_s \tag{15}$$

subject to:

$$\sum_s R_{ls} x_s \leq \hat{c}_l; \qquad \forall l \in \mathcal{L} \tag{16}$$

$$x_s > 0; \qquad \forall s \in \mathcal{S} \tag{17}$$

Proof: Proof is omitted due to space limit.

Max-Min optimization problem is a widely-studied problem formulation in resource management scenarios such as rate allocation in data networks. This is mainly due to an important property, which is inherent in the Max-Min problem, and discriminates it from the others. The optimal solution to the max-min problem, if exists, admits a specific type of fairness characteristic known as Max-Min Fairness (MMF), which will formally be defined in the sequel.

Definition 1: (Max-Min Fair [17]) A feasible rate allocation $(x_s, s \in \mathcal{S})$ is said to be Max-Min Fair (MMF) *if and only if an increase of any rate within the domain of feasible allocations must be at the cost of a decrease of some already smaller rate. Formally, for any other feasible allocation* $\mathbf{y}$*, if $y_s > x_s$ then there must exist some s' such that $x_{s'} \leq x_s$ and $y'_s < x'_s$.*

Depending on the network topology, a Max-Min fair allocation may or may not exist. However, upon its existence, it is unique (see [17] for proof). In what follows the condition under which the Max-Min rate allocation exists will be stated. Before we proceed to this condition, we must define the concept of bottleneck link.

Definition 2: (Bottleneck Link [17]) A link l is said to be a bottleneck for source s if and only if:

1) *link l is saturated; i.e. $\sum_s R_{ls}(x_s + y_s) = c_l$*
2) *Source s on link l has the maximum rate among all sources passing through this link.*

Intuitively, a bottleneck link for source s is the link which limits x_s.

Theorem 2: A MMF rate allocation exists if and only if every source has a bottleneck link.

Proof: See [17] for proof.

Algorithm 2: Max-Min Fair Flow Control Algorithm for BE

Initialization:

1. Initialize $\hat{c}_l$ of all links.
2. Define:
   a. $\mathcal{T}$ as the set of sources not passing through any saturated link.
   b. $\mathcal{B}$ as the set of saturated links.
3. Set source rate vector to zero.
4. Initialize $\mathcal{T} = \mathcal{S}$ and $\mathcal{B} = \varnothing$.

Loop:
Do until ($\mathcal{T} = \varnothing$)

1. $\Delta_s = \min_{l \in (\mathcal{L}-\mathcal{B})} [(c_l - \sum_{s \in (\mathcal{S}-\mathcal{T})} R_{ls} x_s(t)) / \sum_{s \in \mathcal{T}} R_{ls}]$
2. $x_s(t+1) = x_s(t) + \Delta_s, \quad \forall s \in \mathcal{T}$
3. Calculate new bottleneck links and update $\mathcal{B}$.
4. $\forall s \in \mathcal{T}$; if s passes through any saturated link then $\mathcal{T} \Leftarrow \mathcal{T} - \{s\}$

Output:
Communicate BE source rates to the corresponding nodes.

The most famous and simplest algorithm to solve the Max-Min problem (15) is the well-known Progressive Filling Algorithm [16].

We would like to employ the progressive filling algorithm as an iterative solution to the max-min problem (15). Similar to (7), we finally would like to utilize it as a BE flow control mechanism in NoC. The modified version of the progressive filling as a BE flow control mechanism is listed below as algorithm 2.

Algorithm 1 and 2 can be used as centralized flow control mechanisms for BE sources in NoC. In this regard, we consider a simple controller that can be embodied by the NoC, whether as a separate hardware module or as a part of its operating system, which is responsible for running the algorithms. From computational complexity point of view, such a controller must have the ability of carrying out simple mathematical and logical operations, as in Algorithm 1 and 2. Another issue worth considering is the mechanism with which the controller communicates with sources. Since we would like source rate information being communicated without delay and loss, we send them by GS connections to assure that this communication is not subject to congestion.

6. Throughput-Fairness Tradeoff

So far, we have presented two different flow control schemes for BE traffic in NoC. The first one, i.e. Rate-Sum scheme, has been designed to maximize the aggregate source rates. With a slight abuse in the definition of throughput in lossless scenarios, as in NoC, we partly interpret the aggregate of source rate as the throughput of the system. In this respect, Rate-Sum flow control scheme might be construed as one whose aim is to maximize the throughput while simultaneously satisfying link capacity constraints. On the other hand, Max-Min scheme is responsible for maintaining Max-Min fairness among source rates while satisfying capacity constraints.

Any discussion of rate allocation must address the two conflicting issues:

1) *Throughput* as the measure of the efficiency of the network performance.
2) *Fairness* to guarantee that network resources have been allocated to transmitting sources in accordance to a specified fairness metric.

Conflicting with each other, the two mentioned issues are in the extremes of performance spectrum which cannot be simultaneously obtained. Indeed, with the exception of few trivial networking scenarios, there isn't any rate allocation mechanism that can simultaneously realize both optimal fairness and optimal throughput. In fact, any such mechanisms can be seen as providing a tradeoff between throughput and fairness metric. Roughly speaking, boosting one of them will be at the expense of alleviating the other.

One efficient and straight-forward way to establish a tradeoff between throughput and fairness is to introduce weight factors to the underlying optimization problems, i.e. by replacing x_s with $w_s x_s$ in problem (7) and (15). w_s is the weight assigned to source s which determines its priority of rate allocation with respect to other sources. Intuitively, each source upon increase of his weight will obtain more network resources.

As the focus of problem (7) (problem (15)) is on throughput maximization (maintaining max-min fairness) under capacity constraints, the perception of such tradeoffs could not be attained directly. In order to provide more insights, we employ several well-known measures which are defined based on the statistical properties of a rate allocation. In fact, they are defined to quantify performance and fairness factors for a rate allocation vector, regardless of its underlying allocation strategy. Such statistical measures can be further used to compare the inherent tradeoff in different flow control schemes.

Due to space limit, we only introduce the most significant one, i.e. *Jain Fairness Index* [18], which is a widely-addressed index for measuring the fairness maintained amongst the individuals of a rate allocation scenario. Jain Fairness Index, which hereafter will be abbreviated as JFI, is defined as [18]:

$$\text{JFI} = \frac{\left(\sum_{s=1}^{S} x_s\right)^2}{S \sum_{s=1}^{S} x_s^2} \tag{18}$$

It can be proven that JFI for positive rate vectors always falls within [0,1] interval. JFI can be interpreted as a positive fraction which reflects the efficiency of fairness maintained between rate elements. Unity and $1/S$ correspond to the most fair and the least fair cases, respectively.

JFI and throughput together can be used as two simple and efficient measures for quantifying the tradeoffs between performance (throughput) and fairness in any rate allocation scheme. In the next section, we utilize JFI as a fairness measure for different examined scenarios.

7. Simulation Results

In this section we examine the proposed flow control algorithms for a typical NoC architecture. Using MATLAB, we have simulated a NoC with 4×4 Mesh topology consisting of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. In our scheme, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bits long.

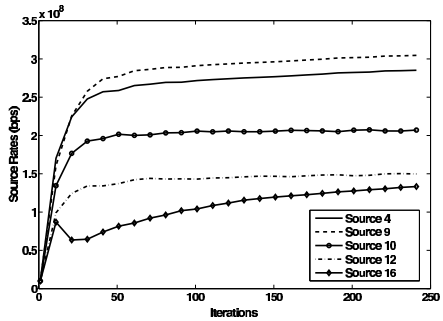


Figure 1. Source Rates vs. Iteration Steps for Rate-Sum

In order to simulate our scheme, some nodes are considered to have a GS-type traffic (such as Multimedia, etc.) to be sent to a destination while other nodes, while others have a BE traffic.

7.1. Comparison Between Rate-Sum and Max-Min

We obtained source rates using proposed algorithms in MATLAB. The evolution of source rates versus iteration steps for both Rate-Sum and Max-Min Fair flow control

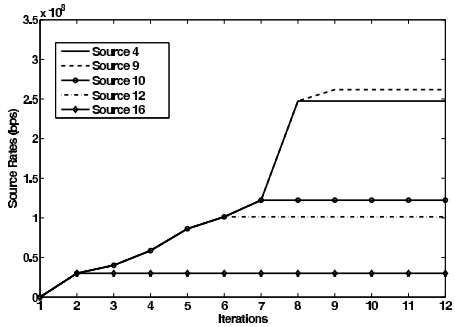


Figure 2. Source Rates vs. Iteration Steps for Max-Min

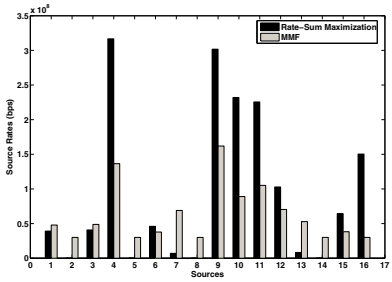


Figure 3. Comparison between Rate-Sum and Max-Min

schemes are shown in Fig. 1 and Fig. 2, respectively. These figures show that after passing enough iteration steps, the proposed algorithms converge to their steady state points. For the sake of convenience in comparing the two schemes, steady state source rates for all sources are depicted in Fig.

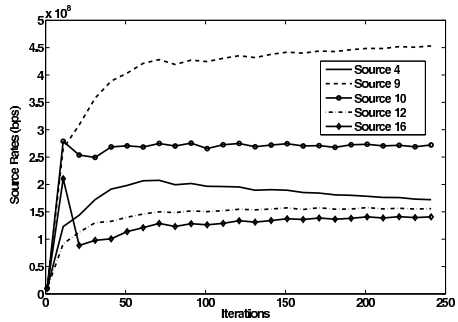


Figure 4. Source Rates vs. Iteration Steps for Weighted Rate-Sum with w_1

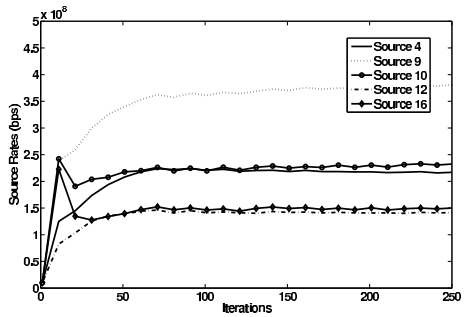


Figure 5. Source Rates vs. Iteration Steps for Weighted Rate-Sum with w_2

3. Comparing Rate-Sum and Max-Min in Fig. 3, it's evident that although Rate-Sum criterion aims at maximizing the sum of source rates, there is no guarantee for the rates of weak sources, i.e. those which achieve very small rates. Indeed, in many scenarios with Rate-Sum flow control, such sources will obtain as small as zero. On the other hand, the weakest source in Max-Min scenario obtains about 0.3 Gbps. Also, it is clear that the variance of Max-Min scheme is evidently less than that of Maximum Rate-Sum (MRS) scheme, which in turn implies the inherent fairness in the Max-Min rate allocation.

From Table 1 we realize that rate allocation with Maximum Rate-Sum criterion, yields greater aggregate rate than Max-Min Fair. However, as discussed above, Max-Min Algorithm guarantees that the rate allocation is Max-Min fair, and hence the minimum source rate wouldn't be greater with any other feasible rate allocation and therefore rate allocation is carried out in favor of such weak sources. On the contrary, Maximum Rate-Sum has no guarantee for such sources and as a result, the weakest source, has achieved as small as zero.

7.2. Influence of Weight Factors

In order to have much more flexibility to balance between throughput and fairness, we introduce two weight factors, w_1 and w_2 to determine the priority of resource allocation. Due to space limit, values of weight factors have been omitted. Such weight factors can be appropriately derived so as to designate network resources (link capacities) in favor

Table 1. Comparison between MRS and MMF

	Max-Min Fair	Rate-Sum
Latest Rate	0.310×10^8	0
Sum of Source Rate	10.079×10^8	15.349×10^8

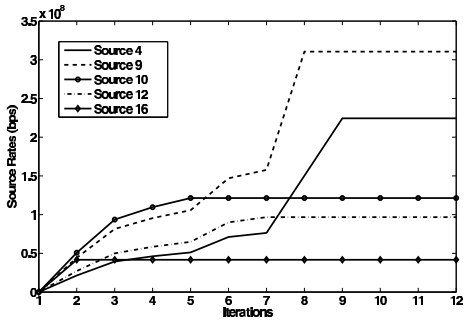


Figure 6. Source Rates vs. Iteration Steps for Weighted Max-Min with w_1

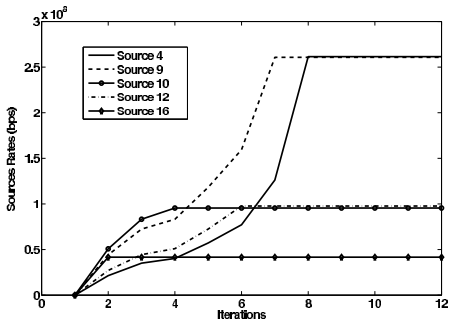


Figure 7. Source Rates vs. Iteration Steps for Weighted Max-Min with w_2

of throughput or fairness. In this respect, we have considered four additional scenarios featuring Weighted Rate-Sum (WMRS) and Weighted Max-Min (WMMF) schemes. The corresponding rate allocations are depicted in Fig. 4-7. From these figures, it is apparent that different weight factors have led to different rate allocations.

7.3. Fairness Metrics

To compare the results of the above mentioned schemes in more detail, we have considered four parameters featuring the merit of the different schemes as following:

1) *Variance of source rates with respect to mean value*
2) *Jain's fairness Index (JFI) [18]*
3) *Min-Max ratio [18]*

The Min-Max ratio is defined as (19).

$$\text{Min-Max Ratio} = \frac{\min_{s \in \mathcal{S}} x_s}{\max_{s \in \mathcal{S}} x_s} \tag{19}$$

The aforementioned parameters for MMF, WMMF (with two different weights), MRS and WMRS (with two different

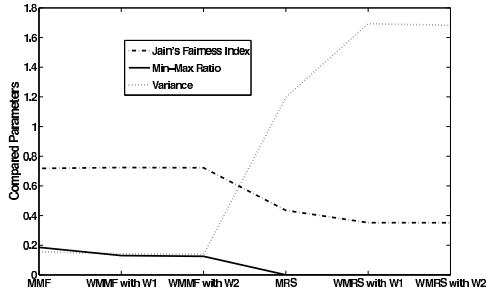


Figure 8. Different Parameters for Different Scenarios

weights) schemes are depicted in Fig. 8. It is apparent that using MMF and WMMF schemes, the variance of source rates are considerably less than MRS and WMRS, which denote the intrinsic fairness in these mechanisms with respect to MRS and WMRS mechanisms. Smaller variance results in the larger Min-Max Ratio and JFI; therefore MMF and WMMF schemes have greater Min-Max Ratio and JFI.

8. Conclusion

In this paper we addressed the problem of flow control for BE traffic in NoC architectures. We considered two extreme cases of the family of α-Fair utility maximization problems, whose solutions led to two iterative flow control algorithms for BE traffics. These extreme cases were 0-Fair and ∞-Fair, which are semantically connected to the throughput-optimal and fairness-optimal scenarios, respectively. These schemes aim at achieving two extreme goals: the first one, MRS aims at maximizing rate-sum (throughput) of the system, while the second, MMF aims at allocating resources in favor of weak sources. We focused on the concept of weight factors to remedy the problem of weak sources in MRS and the problem of throughput inefficiency in MMF. Simulation experiments validated that introducing appropriately-assigned weight factors, could efficiently compromise between throughput and fairness, making proposed flow control algorithms suitable for realistic scenarios.

References

[1] L. Benini, and G. DeMicheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, no. 1, pp. 70-78, 2002.

[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Design Automation Conference*, pp. 684-689, 2001.

[3] M. S. Talebi, F. Jafari, and A. Khonsari, "A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization," *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 381-386, 2007.

[4] M. S. Talebi, F. Jafari, A. Khonsari, and M. H. Yaghmaee, "A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization," *International Conference on Computational Science and its Applications*, pp. 398-409, 2007.

[5] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness, and stability," *Operational Research Society*, vol. 49, no. 3, pp. 237-252, 1998.

[6] Y. Gu, H. O. Wang, and Y. Hong, "A predictive congestion control algorithm for high speed communication networks," *American Control Conference*, vol. 5, pp. 3779-3780, 2001.

[7] C. Yang and A. V. S. Reddy, "A taxonomy for congestion control algorithms in packet switching networks," *IEEE Network*, vol. 9, no. 4, pp. 34-45, 1995.

[8] J. Hu and R. Marculescu, "DyAD - smart routing for networks-on-chip," *Design Automation Conference*, pp. 260-263, 2004.

[9] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," *Design Automation Conference*, pp. 839-844, 2006.

[10] J. W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip," *Design, Automation and Test in Europe Conference*, pp. 948-953, 2007.

[11] F. Jafari,M. S. Talebi, A. Khonsari, and M. H. Yaghmaee, "Best Effort Flow Control Mechanisms in on-Chip Architectures," Technical Report, TRCS2008-30, IPM, School of Computer Science, 2008.

[12] S. H. Low, and D. E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861-874, 1999.

[13] J. Mo, J. Walrand, "Fair End-to-End Window-Based Congestion Control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556-567, 2000.

[14] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[16] D. P. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1992.

[17] J. Y. Le Boudec, "Rate adaptation, Congestion Control and Fairness: A Tutorial," Ecole Polytechnique Federale de Lausanne (EPFL), 2001.

[18] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," DEC Research Report TR-301, 1984.

Paper 8

Optimal Regulation of Traffic Flows in Network-on-Chip

F. Jafari
Z. Lu
A. Jantsch
M. H. Yaghmaee

In the Proceedings of the Design Automation & Test in Europe (DATE), pp. 1621-1624, Dresden, Germany, March 2010.

Optimal Regulation of Traffic Flows in Networks-on-Chip

Fahimeh Jafari*†, Zhonghai Lu†, Axel Jantsch† and Mohammad H. Yaghmaee*
*Ferdowsi University of Mashhad, Iran
†Royal Institute of Technology (KTH), Sweden
Email: fjafari@wali.um.ac.ir, {zhonghai, axel}@kth.se, hyaghmae@ferdowsi.um.ac.ir

***Abstract*—We have proposed (σ, ρ)-based flow regulation to reduce delay and backlog bounds in SoC architectures, where σ bounds the traffic burstiness and ρ the traffic rate. The regulation is conducted per-flow for its peak rate and traffic burstiness. In this paper, we optimize these regulation parameters in networks on chips where many flows may have conflicting regulation requirements. We formulate an optimization problem for minimizing total buffers under performance constraints. We solve the problem with the interior point method. Our case study results exhibit 48% reduction of total buffers and 16% reduction of total latency for the proposed problem. The optimization solution has low run-time complexity, enabling quick exploration of large design space.**

I. Introduction

Integrating IPs into a SoC infrastructure presents challenges because (1) traffic flows from IPs are diverse and typically have stringent performance constraints; (2) the impact of interferences among traffic flows is hard to analyze; (3) due to the cost and power constraint, buffers in the SoC infrastructure must not be over-dimensioned while still satisfying performance requirements even under worst case conditions.

The admission of traffic flows from source IPs into the SoC infrastructure can be controlled by a regulator rather than injecting them as soon as possible [1]. In this way, we can control Quality-of-Service (QoS) and achieve cost-effective communication. To lay a solid foundation for our approach, flow regulation has been based on network calculus [2]. By importing and extending the analytical methods from network calculus, we can obtain worst-case delay and backlog bounds. In [3], we implemented the microarchitecture of the regulator and quantified its hardware speed and cost. The aim of this paper is to optimize the regulator parameters including peak rate and traffic burstiness of flows by formulating an optimization problem.

Silicon area and power consumption are two critical design challenges for NoC architectures. The network buffers take up a significant part of the NoC area and power consumption; consequently, the size of buffers in the system should be minimized. On the other hand, buffers should be large enough to obtain predictable performance. It means that, there is a trade-off between buffer size and performance metrics. Hence, we address an optimization problem of minimizing the total number of buffers subject to the performance constraints of the applications running on the SoC. Finally, we show the benefits of the proposed method and quantify performance improvement and buffer size reduction.

The remainder of this paper is organized as follows. Section II gives account of related works. In Section III, we introduce the flow regulation concept along with the basics of *Network Calculus*. Section IV discusses the underlying system model. Section V formulates the minimizing buffer optimization problem. Our simulation results are described in Section VI. Finally, Section VII gives the conclusions.

II. Related Work

NoC based SoC architectures are often designed for a specific application or a class of applications. Thus, designers customize it for a specific application to achieve best performance, and cost trade-offs. The authors in [4] show the advantage of the topological mapping of IPs on the NoC architectures. In [5], the network topology customization and its effects on the system are considered. In [6], the authors investigate the customized allocation of buffer resources to different channels of routers. Actually, these works strived to distribute a given budget of buffering space among channels. Also, they are based on the average-case analysis which is not appropriate for a system with hard real-time requirements.

The presented work in this paper follows a different direction by addressing an optimization problem to find the minimum total buffering requirements while satisfying acceptable communication performance. Also, our method is presented based on tight worst-case bounds derived by network calculus. Therefore, it is suitable for real-time system designs.

III. The Concepts of Flow Regulation

A. Network Calculus Basics

In network calculus [2], a flow $f_j(t)$ represents the accumulated number of bits transferred in the time interval $[0, t]$. To obtain the average and peak characteristics of a flow, Traffic SPECification (TSPEC) is used. With TSPEC, f_j is characterized by an *arrival curve* $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate ($p_j \geq \rho_j$), σ_j the burstiness ($\sigma_j \geq L_j$), and ρ_j the average (sustainable) rate that we denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. The burstiness also is a important case among these parameters because a flow with low average rate and unlimited burst size can incur an unlimited delay on its own packets.

The abstraction of service curve is used in Network calculus to model a network element processing traffic flows. A well-formulated service model is the latency-rate function $\beta_{R,T} = R(t-T)^+$, where R is the minimum service rate and T is the maximum processing latency of the node [2]. Notation $x^+ = x$ if $x > 0; x^+ = 0$, otherwise.

According to [2], the maximum delay and the buffer required for flow j are bounded by Eq. (1) and (2), respectively.

$$\bar{D}_j = \frac{L_j + \theta_j(p_j - R)^+}{R} + T \quad (1)$$

978-3-9810801-6-2/DATE10 © 2010 EDAA

$$\bar{B}_j = \sigma_j + \rho_j T + (\theta_j - T)^+[(p_j - R)^+ - p_j + \rho_j] \quad (2)$$

where $\theta_j = (\sigma_j - L_j)/(p_j - \rho_j)$. The output flow f_j^* is bounded by another affine arrival curve $\alpha_j^*(t) = (\sigma_j + \rho_j T) + \rho_j t,\ \theta_j \leq T$; $\alpha_j^*(t) = min((T+t)(min(p_j, R)) + L_j + \theta_j(p_j - R)^+, (\sigma_j + \rho_j T) + \rho_j t),\ \theta_j > T$.

B. Regulation Spectrum

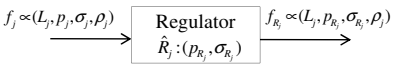


Fig. 1. Flow regulation

TSPEC can also be used to define a traffic regulator. Fig. 1 shows that an input flow f_j reshaped by a regulation component $\hat{R}_j(p_{R_j}, \sigma_{R_j})$ results in an output flow f_{R_j}. We assume the regulator has the same input and output data unit, *flit*, and the same input and output capacity *C flits/cycle*. We also assume that f_j's average bandwidth requirement must be preserved. The output flow f_{R_j} is characterized by the four parameters $(L_j, p_{R_j}, \sigma_{R_j}, \rho_j)$, where $p_{R_j} \in [\rho_j, p_j]$, $\sigma_{R_j} \in [L_j, \sigma_j]$. f_j can be losslessly reshaped by the regulator, meaning that f_{R_j} has the same L and average rate ρ as f_j. The two intervals $p_{R_j} \in [\rho_j, p_j]$ and $\sigma_{R_j} \in [L_j, \sigma_j]$ are called the *regulation spectrum*, where the former is for the regulation of peak rate and the latter for the regulation of traffic burstiness. We implemented microarchitecture of the regulator and quantified its hardware speed and cost in [3]. Selecting appropriate p_{R_j} and σ_{R_j} is very effective in performance and cost of communications.

IV. System Model

A. Assumptions and Notations

We consider an NoC architecture which can have different topologies. Every node contains an IP core and a router with $p+1$ input channels and $q+1$ output channels. NIs provide an interface between IPs and the network. Note that the presence of NIs is the consequence of using a network not regulators. Regulators are inserted between the source IP and NI and their number is the same as the number of flows originating from that node. We presume the number of Virtual Channels (VCs) for each Physical Channel (PC) is the same as the number of flows passing through that channel. Fig. 2 shows required buffers of flows f_1 and f_2 from different sources to the same destination. The following analysis on buffer requirements of flows is illustrated by this figure. Although in this paper we have focused on the output buffers of switches, our method can be easily adapted to input buffers, too. We also assume that the NoC architecture is lossless, and packets traverse the network in a best-effort fashion using a deterministic routing.

We consider NoC as a network with a set of bidirectional channels L, a set of sources S and a set of flows F. Each physical channel $i \in L$ has a fixed capacity of c_{l_i} *flits/cycle*. We denote the set of flows that share channel i by F_{l_i} and their number is denominated as n_{l_i}. Similarly, the set of channels that flow j passes through, is denoted by L_{f_j} and their number is denominated as n_{f_j}. By definition, $j \in F_{l_i}$ if and only if $i \in L_{f_j}$.

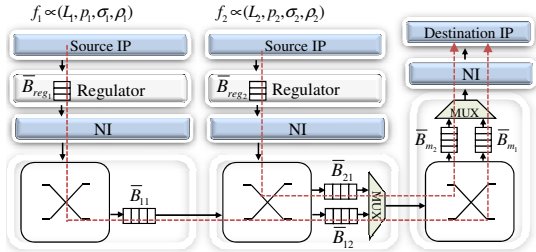


Fig. 2. An example of required buffers for two flows

B. The Analysis of Network Elements

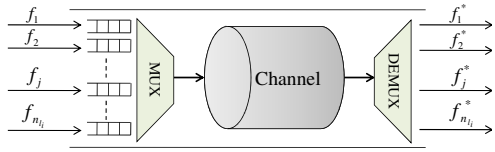


Fig. 3. Shared channel

Fig. 3 depicts a channel l_i allocated to n_{l_i} flows. Since the arbitration policy determines how much the flows influence each other, it has to be known. We assume that the channel access is arbitrated with a round robin policy. Assuming a fixed word length of L_w in all of flows, round robin arbitration means that each flow gets at least a c_{l_i}/n_{l_i} of the channel bandwidth. A flow may get more if the other flow uses less, but we now know a worst-case lower bound on the bandwidth. Round robin arbitration has good isolation properties because the minimum bandwidth for each flow does not depend on properties of the other flow. We can model a round robin arbiter of channel l_i as a latency-rate server [7] that its function is as $\beta_{R_{l_i}, T_{l_i}} = R_{l_i}(t - T_{l_i})^+$. R_{l_i} and T_{l_i} are defined as following:

$$R_{l_i} = \frac{c_{l_i}}{n_{l_i}} \quad (3)$$

$$T_{l_i} = \frac{(n_{l_i} - 1)L_w}{c_{l_i}} \quad (4)$$

Fig. 4 shows a traffic flow f_j after regulation which is called f_{R_j} and is passing through adjacent channels. Every channel $l_i \in L_{f_j}$ can be modeled as a latency-rate server with service curve $\beta_{R_{l_i}, T_{l_i}}$.

Assuming node k is destination of flow j, the ejection channel multiplexer of this node also can be modeled as a latency-rate server $\beta_{R_{m_k}, T_{m_k}}$. If processing capacity of the multiplexer is considered as c_{m_k} *flits/cycle*, it offers minimum service rate R_{m_k} *flits/cycle* and the maximum delay T_{m_k} *cycles* for each flow as following:

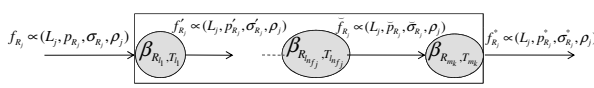


Fig. 4. Modeling each network element as a latency-rate server

$$R_{m_k} = \frac{c_{m_k}}{n_{d_k}} \tag{5}$$

$$T_{m_k} = \frac{(n_{d_k} - 1)L_w}{c_{m_k}} \tag{6}$$

where n_{d_k} is the number of flows with destination node k.

V. Buffer Size Optimization Problem

A. Tight Worst-Case Bounds for Each Flow

Let us assume that flow j passes through the regulator and several network elements offering each a latency-rate service curve. For determining the delay and backlog due to the regulation, the impact of it on the behavior of IPs should be considered. One is that IPs are *stalled* and therefore, there is no queuing buffer at the regulator. In the other case which is considered in this work, IPs are *not stalled* and the regulators use buffers to store transactions. This can reduce back-pressure at the expense of buffering cost. Let D_{reg_j} and B_{reg_j} be the delay and backlog for flow j, respectively. We have $B_{reg_j} = \Delta\sigma_j = \sigma_j - \sigma_{R_j}$, which is the difference between the input and output burstiness of the regulator, and $D_{reg_j} = \Delta\sigma_j/\rho_j$ [1].

For calculating tight worst-case bound on backlog along the network, the sum of the individual bounds on every element is computed. Thus, required buffer in network for flow j is bounded as following:

$$\bar{B}_j = \sum_{i \in L_{f_j}} \bar{B}_{ji} + \bar{B}_{m_j} \tag{7}$$

where $\bar{B}_{ji}$ is upper bound on the buffer of flow j for each $i \in L_{f_j}$ and $\bar{B}_{m_j}$ is maximum required buffer for the multiplexer of the destination node of flow j. $\bar{B}_{ji}$ and $\bar{B}_{m_j}$ can easily be obtained by Eq. (2). Finally, the buffer requirements for the flow j is bounded by $B_{reg_j} + \bar{B}_j$.

For obtaining tight worst-case delay bound along the network, we use the theorem of *Concatenation of network elements* [2]. Given are two nodes sequentially connected and each is offering a latency-rate service curve β_{R_i,T_i}, $i = 1$ *and* 2, can be represented as a single latency-rate server as follows:

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2} \tag{8}$$

We can model all network elements on a given flow as a single latency-rate server $\beta_{R_{e_j},T_{e_j}}$ with following characterictics:

$$R_{e_j} = min(min_{l_i \in Lf_j}(\frac{c_{l_i}}{n_{l_i}}), \frac{c_{m_k}}{n_{d_k}}) \tag{9}$$

$$T_{e_j} = \sum_{l_i \in Lf_j} (\frac{(n_{l_i} - 1)L_w}{c_{l_i}}) + \frac{(n_{d_k} - 1)L_w}{c_{m_k}} \tag{10}$$

Based on a corollary of this theorem which is known *as Pay Bursts Only Once* [2], the equivalent latency-rate server is used for obtaining worst-case delay bound. Therefore, according to (1), (9) and (10), the maximum delay for the flow j in network is bounded by Eq. (11).

$$\bar{D}_j = \frac{L_j + \theta_{R_j}(p_{R_j} - R_{e_j})^+}{R_{e_j}} + T_{e_j} + n_{f_j} d_p \tag{11}$$

where d_p is delay for propagation in a channel which is assumed identical for all channels. Therefore, $n_{f_j} d_p$ is propagation delay in whole network for flow j and $\theta_{R_j} = \frac{\sigma_{R_j} - L_j}{p_{R_j} - \rho_j}$. Hence, the maximum delay for the flow j is bounded as: $D_{reg_j} + \bar{D}_j$.

B. Problem Definition

As stated before, our objective is to choose output peak rate and traffic burstiness of regulators for each flow so as to minimize the buffer requirements while satisfying acceptable performance in the network. Thus, the minimization problem can be formulated as:

$$\min_{p_{R_j}, \sigma_{R_j}} \sum_{\forall f_j \in F} B_{reg_j} + \bar{B}_j \tag{12}$$

subject to:

$$D_{reg_j} + \bar{D}_j \leq d_j; \qquad \forall f_j \in F \tag{13}$$

$$\rho_j \leq p_{R_j} \leq p_j; \qquad \forall f_j \in F \tag{14}$$

$$L_j \leq \sigma_{R_j} \leq \sigma_j; \qquad \forall f_j \in F \tag{15}$$

$$\bar{B}_j > 0; \qquad \forall f_j \in F \tag{16}$$

p_{R_j} and σ_{R_j} are optimization variables and d_j is the maximum delay that flow j can suffer in the network. Since we measured the flow performance in terms of its latency, we can consider d_j as a criterion of minimum guaranteed performance for flow j. It is clear that by following the above mentioned equations, we can understand the effect of optimization variables on the objective function and all constrains of the defined problem.

In the literature, problem (12) is called a nonconvex Non-Linear Programming (NLP) problem [8]. There are different methods for solving this kind of optimization problems. In particular, we will use the Interior Point method [8] [9] to solve it.

VI. Experimental Results

A. Experimental Setup

To evaluate the capability of our method, we applied it to a real application provided by Ericsson Radio Systems which are mapped to a 4×4 2D mesh network. Although the experiments are performed on a mesh, our method is topology independent. In this work, the proposed analytical model is implemented in MATLAB and throughout the experiments, we consider an SoC with 500 MHZ frequency, 32 flits packets and 32 bits flits. We also assume that packets traverse the network on a shortest path using a deadlock free XY routing. As mapped onto a 4×4 mesh in Fig. 5, this application consists of 16 IPs. Specifically, n_2, n_3, n_6, n_9, n_{10}, and n_{11} are ASICs; n_1, n_7, n_{12}, n_{13}, n_{14}, and n_{15} are DSPs; n_5, n_8, and n_{16} are FPGAs; n_1 is a device processor which loads all nodes with program and parameters at startup, sets up, and controls resources in normal operation. Traffic to/from n_1 is for system initial configuration and no longer used afterward. There are 26 node-to-node traffic flows that are categorized into nine types of traffic flows $\{a, b, c, d, e, f, g, h, i\}$, as marked in the figure. The traffic flows are associated with a bandwidth requirement.

TABLE I
Comparison of the required buffer between different schemes

	Network Buffer	Regulator Buffer	Total Buffer
Without Reg.	421	0	421
Unoptimized Reg.	400	46	446
Optimized Reg.	196	21	217

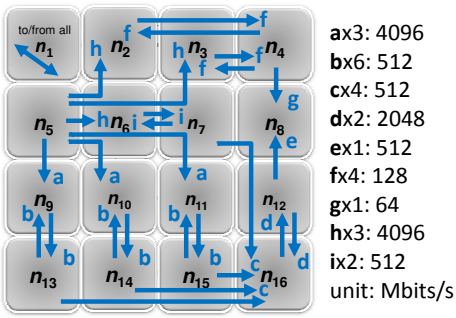


Fig. 5. Ericsson radio systems application

B. Buffer Size Optimization

Tables I and II, respectively, depict the maximum buffer requirements and delay for three schemes: In *without regulation*, there is no regulator; in *unoptimized regulation*, there is a regulator but it works on the worst-case with respect to buffer requirements; *optimized regulation* works based on the proposed minimizing buffer problem (12). From these tables, we can see that the *optimized regulation* scheme leads to a 48% reduction in total maximum required buffer and 16% in total maximum delay when compared with the *without regulation* scheme. Furthermore, these tables show that generally the regulator decreases the maximum buffer and delay in the network because of reducing the contention for shared resources. However, the *unoptimized regulation* scheme does not arrange these parameters appropriately; consequently, buffer area and packet latency in the regulator are increased to the extent that total buffer requirements and delay in this scheme become more than the *without regulation* scheme.

To go into more detail, we depict maximum required buffer and delay of each flow for these schemes in Fig. 6 and 7, respectively. Regarding Fig. 6, it is apparent that in the network with the proposed regulator, most flows require less buffer and also, as mentioned in Table I, total required buffer in this scheme is just a little more than half of it in the network wihout regulator. Also, Fig. 7 shows that regulated flows can experience longer or shorter delays than other schemes which depends on their requested QoS and also the buffer distribution in the whole network. However, due to Table II, total and network delay are decreased in the *optimized regulation* scheme because of buffer-aware allocation in the network and contention reduction for shared resources.

The run-time of the proposed method in MATLAB is typically in the order of a few seconds. It is about 0.22 *sec* for the proposed problem of this application. Another interesting point is that the proposed regulator have no negative effect on the network throughput and it is the same in with and without regulation schemes.

TABLE II
COMPARISON OF THE MAXIMUM DELAY BETWEEN DIFFERENT SCHEMES

	Network Delay	Regulator Delay	Total Delay
Without Reg.	1302.9	0	1302.9
Unoptimized Reg.	1219.3	677.6323	1897
Optimized Reg.	907.6691	183.8812	1091.6

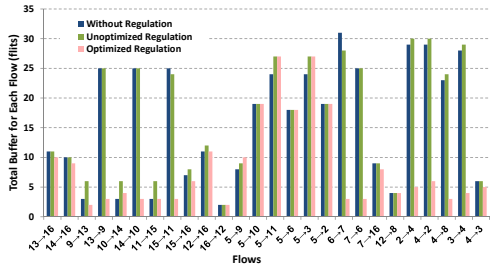


Fig. 6. Maximum buffer requirements for each flow

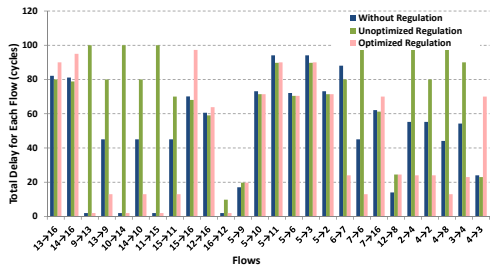


Fig. 7. Maximum delay for each flow

VII. CONCLUSION

In this paper, based on the concepts of regulation spectrum, we have presented an optimization problem for minimizing total buffers under QoS requirements. The regulation analysis is performed for best-effort packet switching networks. We have also demonstrated that the proposed model exerts significant impact on communication performance and buffer requirements. Since reusing similar or identical switches facilitates the design process of NoC-based systems, as future work we intend to model both objectives as a multi-objective problem.

REFERENCES

[1] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf and T. Henriksson, "Flow Regulation for On-Chip Communication", *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, Nice, France, April 2009.

[2] J. Y. L. Boudec and P. Thiran, *"Network Calculus: A Theory of Deterministic Queuing Systems for the Internet"*, Number 2050 in LNCS, 2004.

[3] Z. Lu, D. Brachos, and A. Jantsch, "A flow regulator for on-chip communication", *In Proceedings of the System on Chip Conference (SOCC)*, Belfast, Northern Ireland, 2009.

[4] A. E. Kiasari, S. Hessabi, H. Sarbazi-Azad, "PERMAP: A performance-aware mapping for application-specific SoCs", *Proceedings of ASAP*, pp. 73-78, 2008.

[5] A. Jalabert, S. Murali, L. Benini, and G. De Micheli, "xPipesCompiler: A tool for instantiating application-specific NoCs," *Proceedings of Design, Automation and Test in Europe Conference (DATE)*, pp. 884-889, 2004.

[6] L. P. Tedesco, N. Calazans, and F. Moraes, "Buffer Sizing for Multimedia Flows in Packet-Switching NoCs", *Journal Integrated Circuits and Systems*, Vol. 3, No. 1, pp. 46-56, 2008.

[7] F. Gebali and H. Elmiligi, editors, "Networks on Chip: Theory and Practice", Taylor and Francis Group LLC - CRC Press, 2009.

[8] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[9] H.Y. Benson, R.J. Vanderbei, D.F. Shanno. "Interior-Point Methods for Nonconvex Nonlinear Programming: Filter Methods and Merit Functions" *Computational Optimization and Applications*, Vol. 23, No. 2, pp. 257-272(16), 2002.

Paper 9

Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling

F. Jafari
A. Jantsch
Z. Lu

In the Proceedings of the International Conference on Computer Design (ICCD), pp. 445-446, Amherst, USA, October 2011.

Output Process of Variable Bit-Rate Flows in On-chip Networks Based on Aggregate Scheduling

Fahimeh Jafari, Axel Jantsch, and Zhonghai Lu
Royal Institute of Technology (KTH), Sweden
Email: {fjafari, axel, zhonghai}@kth.se

***Abstract*—This paper proposes an approach for more accurate analyzing of output flows in FIFO multiplexing on-chip networks with aggregate scheduling by considering peak behavior of flows. The key idea of our proposed method involves presenting and proving a technical proposition to derive output arrival curve for an individual flow under the mentioned system model.**

I. Introduction

Since the number of real-time communication services being deployed on NoCs is increasing [1], it is clear that architectures based on aggregate scheduling, which schedule multiple flows as an aggregate flow, will be an appropriate option for transmitting real-time traffic. For example, the composition of flows sharing the same buffer can be considered as an aggregate flow [2]. Furthermore, real-time applications require stringent QoS guarantees which usually employed by tight performance bounds. As analyzing output behavior of flows gives an exact vision about output metrics used for obtaining performance bounds, we aim for deriving the output characterization of Variable Bit-Rate (VBR) traffic transmitted in the FIFO order and scheduled as aggregate. In this paper, based on network calculus [3][4], we present and prove the required proposition for calculating output arrival curve under the mentioned system model.

The VBR is a class of traffic in which the rate can vary significantly from time to time, containing bursts. Real-time VBR flows can be characterized by a set of four parameters, (L, p, σ, ρ), where L is the maximum transfer size, p peak rate, σ burstiness, and ρ average sustainable rate [4]. Our assumption is that the application-specific nature of the network enables to characterize traffic with sufficient accuracy.

Authors in [5] present a theorem for calculating per-flow output arrival curve in tandem networks of rate-latency nodes traversed by leaky-bucket shaped flows. This theorem investigates computing output traffic characterization only for average behavior of flows while the proposed proposition in this paper considers both average and peak behavior, which results in a more accurate analysis.

II. Network Calculus Background

Network Calculus is a theory that provides deep analysis on flow problems encountered in networking. It uses the abstraction of service curve to model a network element processing traffic flows modeled with an arrival curve in terms of input and output flow relationships. Network elements such as routers, links, and regulators, can be modeled by corresponding service curves. A flow f is an infinite stream of unicast traffic (packets) sent from a source node to a destination node. To model the average and peak characteristics of a flow, Traffic SPECification (TSPEC) is used. As shown in Fig. 1, with TSPEC, f is characterized by an *arrival curve* $\alpha(t) = min(L + pt, \sigma + \rho t)$ in which $p \geq \rho$ and $\sigma \geq L$.

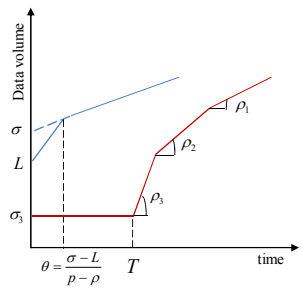


Fig. 1. Left Curve is the arrival curve of flow f with TSPEC (L, p, σ, ρ) and right one is the pseudoaffine service curve with three leaky-bucket stages

Theorem 1. *(Output Flow [4]). Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β, the output flow is constrained by the arrival curve $\alpha^* = \alpha \oslash \beta$, where $\oslash$ represents the min-plus deconvolution of two functions $f, g \in F$, $(f \oslash g)(t) = sup_{s \geq 0} \{f(t+s) - g(s)\}$.*

III. Analysis

We assume that flows are classified into a pre-specified number of aggregates at their source nodes. In addition, we assume that traffic of each aggregate is buffered and transmitted in the FIFO order and different aggregates are buffered separately. The network is lossless, and packets traverse the network using a deterministic routing.

we first consider a class of curves, namely pseudoaffine curves [5], which is a multiple affine curve shifted to the right and given by $\beta = \delta_T \otimes [\otimes_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$. In fact, a pseudoaffine curve represents the service received by single flows in tandems of FIFO multiplexing rate-latency nodes. Due to concave affine curves, it can be rewritten as $\beta = \delta_T \otimes [\wedge_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$, where the non-negative term T is denoted as *offset*, and the affine curves between square brackets as leaky-bucket stages. Fig. 1 shows a pseudoaffine service curve with three leaky-bucket stages.

We now propose the proposition for computing output arrival curve as follows.

Proposition 1. *(Output Arrival Curve with FIFO) Consider a VBR flow, with TSPEC (L, p, ρ, σ), served in a node that guarantees to the flow a pseudo affine service curve equal to $\beta = \delta_T \otimes \gamma_{\sigma_x,\rho_x}$. The output arrival curve α^* given by:*

$$\alpha^* = \begin{cases} \theta > T & \gamma_{(p\wedge R)T+\theta(p-R)^+ + L - \sigma_x, p\wedge R} \\ & \wedge \gamma_{\sigma-\sigma_x+\rho T,\rho} \\ \theta \leq T & \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{cases} \quad (1)$$

where $\wedge$ represents the minimum operation.

Proof. *From Theory 1, the output flow is constrained by the arrival curve $\alpha^* = \alpha \oslash \beta = sup_{u\geq 0}\{\alpha(t+u) - \beta(u)\}$. Thus, $\alpha^* = sup_{u\geq 0}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x - \rho_x(u-T)^+\}$*

We now consider two different situations including $\theta \leq T$ and $\theta > T$. If $\theta \leq T$, we have:

$$\begin{aligned} \alpha^* &= sup_{u\geq 0}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x \\ &\quad -\rho_x(u-T)^+\} \\ &= sup_{0\leq u\leq T}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x\} \\ &\quad \vee sup_{u>T}\{min(\sigma + \rho(t+u), L + p(t+u)) \\ &\quad -\sigma_x - \rho_x u + \rho_x T\} \\ &= \{min(\sigma + \rho(t+T), L + p(t+T)) - \sigma_x\} \vee \\ &\quad sup_{u>T}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x \\ &\quad -\rho_x u + \rho_x T\} \\ &= \{\sigma + \rho(t+T) - \sigma_x\} \vee sup_{u>T}\{\sigma + \rho(t+u) - \sigma_x \\ &\quad -\rho_x u + \rho_x T\} \\ &= \{\sigma + \rho(t+T) - \sigma_x\} \vee sup_{u>T}\{\sigma + \rho t + \rho_x T - \sigma_x \\ &\quad +u(\rho - \rho_x)\} \end{aligned}$$

Since $\rho \leq \rho_x$ and thus $\rho - \rho_x$ is negative, u in the second term should get its lowest possible value to achieve supremum. Thus, we have

$$\begin{aligned} &= \{\sigma + \rho(t+T) - \sigma_x\} \vee \{\sigma + \rho(t+T) - \sigma_x\} \\ &= \sigma + \rho(t+T) - \sigma_x = \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{aligned} \quad (2)$$

If $\theta > T$, we have:

$$\begin{aligned} \alpha^* &= sup_{u\geq 0}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x - \\ &\quad \rho_x(u-T)^+\} \\ &= sup_{0\leq u\leq T}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x\} \\ &\quad \vee sup_{u>T}\{min(\sigma + \rho(t+u), L + p(t+u)) - \sigma_x \\ &\quad -\rho_x u + \rho_x T\} \\ &= \{min(\sigma + \rho(t+T), L + p(t+T)) - \sigma_x\} \vee sup_{u>T}\{ \\ &\quad min(\sigma + \rho(t+u) - \sigma_x - \rho_x u + \rho_x T, L + p(t+u) \\ &\quad -\sigma_x - \rho_x u + \rho_x T)\} \end{aligned} \quad (3)$$

For completing the proof, we need to consider the second term in right side of Eq. (3) in details. Therefore, we call it $Term_2$ in the following:

$$\begin{aligned} Term_2 = &sup_{u>T}\{min(\sigma + \rho(t+u) - \sigma_x - \rho_x u + \rho_x T, \\ &L + p(t+u) - \sigma_x - \rho_x u + \rho_x T)\} \end{aligned}$$

For solving $Term_2$, we consider two different situations including $t + u \leq \theta$ and $t + u \geq \theta$. Thus, if $t + u \geq \theta$, we have $u > T$ and $t + u \geq \theta$.

$$\begin{aligned} \Rightarrow Term_2 &= sup_{u>T}(\sigma + \rho(t+u) - \sigma_x - \rho_x u + \rho_x T) \\ &= sup_{u>T}(\sigma + \rho t + \rho_x T - \sigma_x + (\rho - \rho_x)u) \\ &= \sigma + \rho t + \rho_x T - \sigma_x + (\rho - \rho_x)T \\ &= \sigma + \rho(t+T) - \sigma_x = \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{aligned} \quad (4)$$

If $t + u \leq \theta$, we have $u > T$ and $t + u \leq \theta \Rightarrow u \leq \theta - t$.

$$\begin{aligned} \Rightarrow Term_2 &= sup_{T<u\leq\theta-t}(L + p(t+u) - \sigma_x - \rho_x u + \rho_x T) \\ &= sup_{T<u\leq\theta-t}(L + pt + \rho_x T - \sigma_x + (p - \rho_x)u) \end{aligned}$$

Selecting an appropriate value for u depends on if $(p - \rho_x)$ is positive or negative. Therefore, we have two different situations including $p > \rho_x$ and $p \leq \rho_x$. If $p > \rho_x \Rightarrow (p - \rho_x)$ is positive and u should be the highest possible value to have supremum value. Thus, due to $u = \theta - t$, $Term_2 = L + \rho_x(t+T) - \sigma_x + \theta(p - \rho_x)$. If $p \leq \rho_x \Rightarrow (p - \rho_x)$ is negative. Therefore, u gets its lowest value and $Term_2$ is equal to $L + p(t+T) - \sigma_x$.

$$\Rightarrow Term_2 = L + (p \wedge \rho_x)(t+T) - \sigma_x + \theta(p - \rho_x)^+ \quad (5)$$

From Eq. 3, 4 and 5, if $\theta > T$, we have:

$$\begin{aligned} \alpha^* &= min\Big(L + (p \wedge \rho_x)(t+T) - \sigma_x + \theta(p - \rho_x)^+, \\ &\quad \sigma + \rho(t+T) - \sigma_x)) \\ &= \gamma_{(p\wedge R)T+\theta(p-R)^+ + L - \sigma_x, p\wedge R} \wedge \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{aligned} \quad (6)$$

From Eq. 2 and 6, we straightforwardly obtain the thesis.

IV. Conclusions

Real-time applications exert stringent requirements on networks. To this end, we have presented and proved the required proposition for computing the output arrival curve of VBR flows in a FIFO multiplexing network to detail output traffic characterization. The proposition can be applied for an architecture based on aggregate scheduling. In the future, we will present a formal approach to calculate performance bounds under the mentioned system model.

References

[1] Z. Shi, A. Burns, L. S. Indrusiak, "Schedulability Analysis for Real Time On-Chip Communication with Wormhole Switching", *International Journal of Embedded and Real-Time Communication Systems*, vol. 1, no. 2, pp. 1-22, 2010.

[2] Y. Qian, Z. Lu, W. Dou, "Analysis of Worst-case Delay Bounds for Best-effort Communication in Wormhole Networks on Chip", *In the Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*, pp. 44-53, San Diego, CA, 2009.

[3] C. Chang, *Performance Guarantees in Communication Networks*, Springer-Verlag, 2000.

[4] J. Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag, 2004.

[5] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks", *Performance Evaluation*, vol. 63, no. 9, pp. 956-987, 2006.

Paper 10

Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling

F. Jafari
A. Jantsch
Z. Lu

In the Proceedings of the Design Automation & Test in Europe (DATE), pp. 538-541, Dresden, Germany, March 2012.

Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling

Fahimeh Jafari, Axel Jantsch, and Zhonghai Lu
KTH Royal Institute of Technology, Sweden
Email: {fjafari, axel, zhonghai}@ kth.se

***Abstract*—Aggregate scheduling in routers merges several flows into one aggregate flow. We propose an approach for computing the end-to-end delay bound of individual flows in a FIFO multiplexer under aggregate scheduling. A synthetic case study exhibits that the end-to-end delay bound is up to 33.6% tighter than the case without considering the traffic peak behavior.**

I. Introduction

Real-time applications such as multimedia and gaming boxes etc., require stringent performance guarantees, usually enforced by a tight upper bound on the maximum end-to-end delay [1]. For the worst-case performance analysis, we derive the upper delay bound of a flow in a FIFO multiplexing and aggregate scheduling network. The behavior of a flow is determined by four parameters including the maximum transfer size (L), peak rate (p), burstiness (σ), and average sustainable rate (ρ). To calculate the tight delay bound per flow, the main problem is to obtain the *end-to-end Equivalent Service Curve (ESC)* which the tandem of routers provides to the flow. However, the required propositions for calculating performance metrics of *Variable Bit-Rate (VBR)* traffic characterized with (L, p, σ, ρ), transmitted in the FIFO order and scheduled as aggregate do not exist. Based on network calculus [2][3], we first present and prove the required propositions and then calculate the delay bound under the mentioned system model.

There are some works for deriving per-flow worst-case delay bound under different system models [4]-[6]. However, they investigate computing delay bounds only for average behavior of flows while we analyze both average and peak behavior. In [7], we presented a theorem for computing output traffic characterization. The aim of this paper is to represent and prove propositions for deriving end-to-end ESC and tighter upper bound on the end-to-end delay.

II. Network Calculus Background

In network calculus, traffic flows are modeled by arrival curves and network elements by service curves. Network calculus uses Traffic SPECification to model the average and peak characteristics of a flow. With TSPEC, f_j is characterized by an *arrival curve* $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate ($p_j \geq \rho_j$), σ_j the burstiness ($\sigma_j \geq L_j$), and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$.

Network calculus also derives delay bound for lossless systems with service guarantees as the following theorem proves.

Theorem 1. *(Delay Bound [3]). Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β, the virtual delay $d(t)$ for all t satisfies: $d(t) \leq h(\alpha, \beta)$.*

978-3-9810801-8-6/DATE12/©2012 EDAA

The theorem says that the delay is bounded by the maximum horizontal deviation between the arrival and service curves.

Now, we consider a node which guarantees a minimum service curve to an aggregate flow and also handles packets in order of arrival at the node.

Theorem 2. *(FIFO Minimum Service Curves [3]). Consider a lossless node serving two flows, 1 and 2, in FIFO order. Assume that packet arrivals are instantaneous. Assume that the node guarantees a minimum service curve β to the aggregate of the two flows. Assume that flow 2 has α_2 as an arrival curve. Define the family of functions $\beta^{eq}(t, \alpha_2, \tau) \equiv \beta_1^{eq}(t, \tau) = \beta_1^{eq}(t, \tau) = [\beta(t) - \alpha_2(t - \tau)]^+_{\{t > \tau\}}$. For any $\tau \geq 0$ such that $\beta_1^{eq}(t, \tau)$ is wide-sense increasing, then flow 1 is guaranteed the service curve $\beta_1^{eq}(t, \tau)$.*

III. System Model

We assume that flows are classified into a pre-specified number of aggregates. In addition, we assume that traffic of each aggregate is buffered and transmitted in the FIFO order, denoted as FIFO multiplexing. Different aggregates are buffered separately. The network is lossless, and packets traverse the network using a deterministic routing. We call the flow for which we shall derive its delay bound *tagged flow*, other flows that share resources with it *interfering flows*.

While building network calculus analysis models, we follow the notation conventions in the min-plus algebra [3]. $\otimes$ represents the min-plus convolution of two functions $f, g \in F$, the set of wide-sense increasing functions defined on $[0, t)$, $(f \otimes g)(t) = inf_{0 \leq s \leq t} \{f(t - s) + g(s)\}$; $\wedge$ represents the minimum operation, $f \wedge g = min(f, g)$. *Burst delay* function $\delta_T(t) = +\infty$, if $t > T$; $\delta_T(t) = 0$, otherwise. *Affine function* $\gamma_{b,r}(t) = b + rt$, if $t > 0$; $\gamma_{b,r}(t) = 0$, otherwise. Therefore, min-plus convolution of burst delay and affine function is given as $\delta_T \otimes \gamma_{b,r}(t) = b + r(t - T)$.

IV. Analysis

In this section, we propose and prove the propositions needed for analyzing performance of VBR flows in a FIFO multiplexing network. We consider a class of service curves, namely pseudoaffine curves [5], which is a multiple affine curve shifted to the right and given by $\beta = \delta_T \otimes [\otimes_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}] = \delta_T \otimes [\wedge_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$, where the non-negative term T is denoted as *offset*, and the affine curves between square brackets as leaky-bucket stages. In fact, a pseudoaffine curve represents the service received by single flows in tandems of FIFO multiplexing rate-latency nodes. It is clear that a rate-latency service curve is in fact pseudoaffine, since it can be expressed as $\beta = \delta_T \otimes \gamma_{0,R}$.

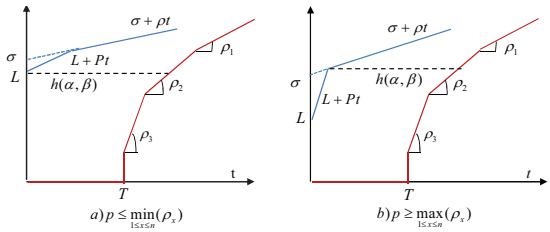


Fig. 1. Computation of delay bound for one VBR flow served by a pseudo affine curve

We now propose a proposition for computing delay bound as follows.

Proposition 1. *(Delay Bound) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage γ_{σ_x,ρ_x}, $1 \le x \le n$, this means we have $\beta = \delta_T \otimes [\otimes_{1\le x\le n}\gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\le x\le n}\gamma_{\sigma_x,\rho_x}]$ and let $\alpha = min(L+pt, \sigma+\rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \ge \rho$ ($\rho^*_\beta = min_{1\le x\le n}\rho_x$), then the maximum delay for the flow is bounded by*

$$h(\alpha,\beta) = T + \left[\vee_{1\le x\le n} \frac{L - \sigma_x + \theta\,(p-\rho_x)^+}{\rho_x}\right]^+ \quad (1)$$

where $\theta_j = (\sigma_j - L_j)/(p_j - \rho_j)$.

Proof. *As stated before in Theorem 1, the delay is bounded by the maximum horizontal deviation between the arrival and service curves. Thus, due to Fig. 1, if $p \le min_{1\le x\le n}(\rho_x)$, we have:*

$$\begin{cases} L = \sigma_1 + \rho_1\,(t_1 - T) \Rightarrow t_1 = T + \dfrac{L-\sigma_1}{\rho_1} \\ L = \sigma_2 + \rho_1\,(t_2 - T) \Rightarrow t_2 = T + \dfrac{L-\sigma_2}{\rho_2} \\ \vdots \quad \vdots \quad \vdots \\ L = \sigma_n + \rho_n\,(t_n - T) \Rightarrow t_n = T + \dfrac{L-\sigma_n}{\rho_n} \end{cases} \quad (2)$$

$$\Rightarrow h(\alpha,\beta) = max_{1\le x\le n} t_x = T + \left[\vee_{1\le x\le n}\frac{L-\sigma_x}{\rho_x}\right]^+ \quad (3)$$

If $p \ge max_{1\le x\le n}(\rho_x)$, due to Fig. 1, we have:

$$\begin{cases} L + p\theta = \sigma_1 + \rho_1\,(t_1 + \theta - T) \\ \qquad \Rightarrow t_1 = T + \dfrac{L + p\theta - \sigma_1}{\rho_1} - \theta \\ L + p\theta = \sigma_2 + \rho_2\,(t_2 + \theta - T) \\ \qquad \Rightarrow t_2 = T + \dfrac{L + p\theta - \sigma_2}{\rho_2} - \theta \\ \qquad \vdots \quad \vdots \quad \vdots \\ L + p\theta = \sigma_n + \rho_n\,(t_n + \theta - T) \\ \qquad \Rightarrow t_n = T + \dfrac{L + p\theta - \sigma_n}{\rho_n} - \theta \end{cases}$$

$$\Rightarrow h(\alpha,\beta) = max_{1\le x\le n} t_x = T + \left[\vee_{1\le x\le n}\frac{L+p\theta-\sigma_x}{\rho_x} - \theta\right]^+$$

$$= T + \left[\vee_{1\le x\le n}\frac{L-\sigma_x+\theta\,(p-\rho_x)}{\rho_x}\right]^+ \quad (4)$$

From Eq. 2 and 4, we can say:

$$h(\alpha,\beta) = T + \left[\vee_{1\le x\le n}\frac{L-\sigma_x+\theta\,(p-\rho_x)^+}{\rho_x}\right]^+ \quad (5)$$

In Propositions 2 and 3, we obtain ESC with FIFO multiplexing under different assumptions.

Proposition 2. *(Equivalent Service Curve) Let β be a pseudo affine curve as $\beta = \delta_T \otimes [\otimes_{1\le x\le n}\gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\le x\le n}\gamma_{\sigma_x,\rho_x}]$ and let $\alpha = min(L+pt, \sigma+\rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \ge \rho$ ($\rho^*_\beta = min_{1\le x\le n}\rho_x$) and $p \ge \rho^\circ_\beta$ ($\rho^\circ_\beta = max_{1\le x\le n}\rho_x$), then the equivalent service curve is obtained by subtracting arrival curve α, $\{\beta^{eq}(\alpha,\tau), \quad \tau = h(\alpha,\beta)\} \equiv \beta^{eq}(\alpha)$, with*

$$\beta^{eq}(\alpha) = \delta_{T+\vee_{1\le i\le n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ + \theta} \otimes [\otimes_{1\le x\le n}[\gamma_{\rho_x\left\{\vee_{1\le i\le n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ - \frac{\sigma-\sigma_x-(\rho_x-\rho)\theta}{\rho_x}\right\}, \rho_x-\rho}]] \quad (6)$$

Proof. *Let us apply Theorem 2 to service curve β as follows.*

$$\beta^{eq}(\alpha,\tau) = [\delta_T \otimes [\otimes_{1\le x\le n}\gamma_{\sigma_x,\rho_x}] - min\,(L + p\,(t-\tau), \sigma + \rho\,(t-\tau))] \quad (7)$$

Eq. (7) is wide-sense increasing for any $\tau \ge 0$. Since we assumed $\tau = h(\alpha,\beta)$, due to Proposition 1, we have:

$$\tau = T + \left[\vee_{1\le x\le n}\frac{L-\sigma_x+\theta\,(p-\rho_x)^+}{\rho_x}\right]^+ \quad (8)$$

Without losing generality, we follow proof for $n = 1$. Therefore, by Eq. (8) we have:

$$\tau - T = \left[\frac{L-\sigma_x+\theta\,(p-\rho_x)^+}{\rho_x}\right]^+ \quad (9)$$

We then apply Theorem 2 to service curve $\acute{\beta}$ ($\acute{\beta}$ is β when $n = 1$) as follows.

$$\beta^{\acute{e}q}(\alpha,\tau) = \delta_T \otimes \gamma_{\sigma_x,\rho_x} - min\,(L + p\,(t-\tau), \sigma + \rho\,(t-\tau)) = \sigma_x + \rho_x\,(t-T) - min\,(L + p\,(t-\tau), \sigma + \rho\,(t-\tau)) \quad (10)$$

We now consider two situations including $0 \le t-\tau \le \theta$ and $t-\tau > \theta$.

If $0 \le t-\tau \le \theta \Rightarrow min\,(L + p\,(t-\tau), \sigma + \rho\,(t-\tau)) = L + p\,(t-\tau)$. Let us assume $\acute{t} = t-\tau \Rightarrow t - T = \acute{t} + (\tau - T)$. From Eq. 9, we can say $t - T = \acute{t} + \left[\frac{L-\sigma_x+\theta(p-\rho_x)^+}{\rho_x}\right]^+$.

$$\beta^{\acute{e}q}(\alpha,\tau) = \sigma_x + \rho_x\left(\acute{t} + \left[\frac{L-\sigma_x+\theta\,(p-\rho_x)^+}{\rho_x}\right]^+\right) - \left(L + p\acute{t}\right) = \sigma_x + \rho_x\acute{t} + \left[L - \sigma_x + \theta\,(p-\rho_x)^+\right]^+ - L - p\acute{t} = -\,(p-\rho_x)\,\acute{t} + \theta\,(p-\rho_x)^+$$

Since $p \ge \rho^\circ_\beta$ and $\acute{t} \le \theta$, we have:

$$\beta^{\acute{e}q}(\alpha,\tau) = -\,(p-\rho_x)\,\acute{t} + \theta\,(p-\rho_x)^+ \le -\,(p-\rho_x)\,\theta + \theta\,(p-\rho_x) \le 0$$

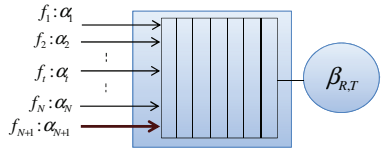


Fig. 2. Computation of ESC for flow $N+1$ in a rate-latency node

Therefore, $\beta^{'eq}(\alpha,\tau)=0$ where $0\le t-\tau\le\theta$. By definition of the service curve, we can say that if $0\le t\le\theta+\tau$ then $\beta^{'eq}(\alpha,\tau)=0$, and this means that the offset of $\beta^{'eq}(\alpha,\tau)$ is equal to $\tau+\theta$.

If $t-\tau>\theta\Rightarrow min\left(L+p\left(t-\tau\right),\sigma+\rho\left(t-\tau\right)\right)=\sigma+\rho\left(t-\tau\right)$. Therefore, $\beta^{'eq}(\alpha,\tau)=\sigma_x+\rho_x\left(t-T\right)-\left(\sigma+\rho\left(t-\tau\right)\right)$. If $\rho_x\tau$ is added to and subtracted from $\beta^{'eq}(\alpha,\tau)$, we have

$$\begin{aligned}\beta^{'eq}(\alpha,\tau)&=\sigma_x+\rho_x\left(t-T\right)-\left(\sigma+\rho\left(t-\tau\right)\right)+\rho_x\tau-\rho_x\tau\\&=\sigma_x-\sigma+\rho_x\left(\tau-T\right)+\left(\rho_x-\rho\right)\left(t-\tau\right)\\&=\delta_\tau\otimes\gamma_{\sigma_x-\sigma+\rho_x(\tau-T),\rho_x-\rho}\end{aligned}\tag{11}$$

Since we concluded that the offset of $\beta^{'eq}(\alpha,\tau)$ is $\tau+\theta$, we add $\left(\rho_x-\rho\right)\theta$ to Eq. 11 and then subtract it. We obtain:

$$\begin{aligned}\beta^{'eq}(\alpha,\tau)&=\sigma_x-\sigma+\rho_x\left(\tau-T\right)+\left(\rho_x-\rho\right)\left(t-\tau\right)\\&\quad+\left(\rho_x-\rho\right)\theta-\left(\rho_x-\rho\right)\theta\\&=\sigma_x-\sigma-\rho\theta+\rho_x\left(\tau+\theta-T\right)+\left(\rho_x-\rho\right)\left(t-\tau-\theta\right)\\&=\delta_{\tau+\theta}\otimes\gamma_{\sigma_x-\sigma-\rho\theta+\rho_x(\tau+\theta-T),\rho_x-\rho}\end{aligned}\tag{12}$$

Thus, the offset of $\beta^{eq}(\alpha,\tau)$ is equal to $\tau+\theta$. Furthermore, each leaky bucket-stage in $\beta^{eq}(\alpha,\tau)$ can be computed as $\gamma_{\acute{\sigma_j},\acute{\rho_j}}$, with $\acute{\sigma_j}=\sigma_x-\sigma-\rho\theta+\rho_x\left(\tau+\theta-T\right)$ and $\acute{\rho_j}=\rho_j-\rho$. Therefore, we have $\beta^{eq}=\delta_{\tau+\theta}\otimes\left[\otimes_{1\le x\le n}\gamma_{\acute{\sigma_x},\acute{\rho_x}}\right]$ and by substituting (8) into β^{eq}, we prove the proposition.

We can specifically capitalize on Proposition 2 to obtain a parametric expression. We assume that the number of flows passing through a rate-latency node is $N+1$. Therefore, for computing ESC for the tagged flow, we should subtract the arrival curves of other N flows. It can be calculated by iteratively applying Proposition 2 for N times. Without any loss of generality, we presume that the tagged flow is flow $N+1$. We now present following proposition:

Proposition 3. *(Equivalent Service Curve for Rate-Latency Service Curve With $N+1$ Flows) Consider one node with a rate-latency service curve $\beta_{R,T}=\delta_T\otimes\gamma_{0,R}$. Let $\alpha_i=min(L_i+p_it,\sigma_i+\rho_it)=\gamma_{L_i,p_i}\wedge\gamma_{\sigma_i,\rho_i}$ be arrival curve of flow i and $p_i\ge R-\sum_{j=1}^{i-1}\rho_j$, where $1\le i\le N+1$ and $N+1$ is the number of flows passing through that node as shown in Fig. 2. The equivalent service curve for flow $N+1$ in the node, obtained by subtracting N arrival curves, is:*

$$\beta_{N+1}^{eq}=\delta_{T+\sum_{i=1}^{N}\left(\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+\!+\theta_i\right)}\otimes\gamma_{0,R-\sum_{j=1}^{N}\rho_j}\tag{13}$$

Proof. *We use the simplest form of mathematical inductive proof method. It proves that a statement involving a number N holds for all values of N. The proof consists of two steps:*

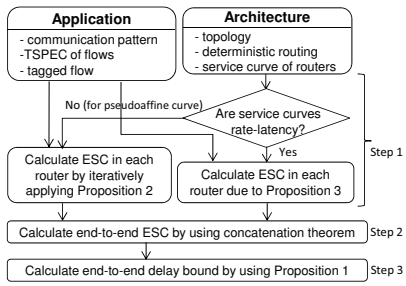


Fig. 3. End-to-end delay bound analysis flow

Base Step: *In this step, we show that the statement holds when $N=1$. In order to verify this, we compute the ESC obtained by subtracting one arrival curve ($N=1$), offered by Proposition 3:*

$$\beta_2^{eq}=\delta_{T+\left[\frac{L_1+\theta_1(p_1-R)^+}{R}\right]^+\!+\theta_1}\otimes\gamma_{0,R-\rho_1}\tag{14}$$

If we apply Proposition 2 for a rate-latency service curve $\beta_{R,T}$ where $n=1$, $\sigma_x=0$ and $\rho_x=R$, Eq. 14 is easily obtained. Therefore, the statement holds when $N=1$.

Inductive Step: *In this step, we show if the statement holds for some N, then the statement also holds when $N+1$ is substituted for N. Assume that β_{N+1}^{eq} is an ESC for flow $N+1$, obtained by subtracting N arrival curves as represented in Eq. 13. We shall compute ESC β_{N+2}^{eq} for flow $N+2$. Therefore, in this case we should subtract $N+1$ arrival curves. After subtracting N arrival curves, the ESC for aggregated flow $\{N+1,N+2\}$ will be equal to β_{N+1}^{eq}. Therefore, for computing β_{N+2}^{eq}, it is enough to subtract flow $N+1$ from β_{N+1}^{eq} by applying Proposition 2.*

From β_{N+1}^{eq}, we can say n, ρ_x, σ_x and T_x in Proposition 2 are as $n=1$, $\rho_x=R-\sum_{j=1}^{N}\rho_j$, $\sigma_x=0$, and $T_x=T+\sum_{i=1}^{N}\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+\!+\sum_{j=1}^{N}\theta_j$. Also, α in Proposition 2 is equal to $\alpha_{N+1}=min(L_{N+1}+p_{N+1}t,\sigma_{N+1}+\rho_{N+1}t)$. After applying Proposition 2 and computing some straightforward algebraic manipulation, β_{N+2}^{eq} is given by:

$$\beta_{N+2}^{eq}=\delta_{T+\sum_{i=1}^{N+1}\left(\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+\!+\theta_t\right)}\otimes\gamma_{0,R-\sum_{j=1}^{N+1}\rho_j}\tag{15}$$

which proves the inductive step.

Fig. 3 shows the overall analysis flow for computing end-to-end delay bound of a tagged flow under the mentioned system model. We illustrate the steps with an example in section V.

V. Numerical Example

To show how the proposed propositions are used, we applied them to a simple example depicted in Fig. 4. The figure depicts a network with 4 flows and 3 routers which serve flows in the FIFO order. f_3 is the tagged flow and f_1, f_2 and f_4 are interfering flows. Flows follow TSPEC, $f_1\propto(1,1,2,0.128)$, $f_2\propto(1,1,2,0.032)$, $f_3\propto(1,1,4,0.256)$, and $f_4\propto(1,1,2,0.008)$. Each router guarantees the service curve of $\beta_{R,T}(t)=\delta_T\otimes\gamma_{0,R}=1(t-1)^+$, where the serving rate $R=1$ *flit/cycle* and the processing latency $T=1$ *cycle*.

A. Computation of the end-to-end equivalent service curve

Step 1: We first calculate the ESC for the tagged flow in each node. Then, we can model a flow passing through a series of routers as a series of concatenated pseudoaffine servers. Before that, θ_j is computed for each flow f_j as $\theta_1 = (\sigma_1 - L_1)/(p_1 - \rho_1) = (2-1)/(1-0.128) = 1.146$, $\theta_2 = 1.033$, $\theta_3 = 4.032$, and $\theta_4 = 1.008$.

We use sub-index "(j, r_i)" for notations to indicate that they are related to flow j in router i. For example, $\beta^{eq}_{(j,r_i)}$ denotes the ESC of flow j in router i.

From Proposition 3, we obtain the ESC for f_3 in node 1 by subtracting arrival curves of f_1 and f_2. The serving rate and latency for aggregate flow $f_{(1,2,3)}$ in node 1 is equal to $R_1 = 1$ and $T_1 = 1$, respectively. Therefore, we have $T^{eq}_{(3,r_1)} = T_1 + \left(\left[\frac{L_1+\theta_1(p_1-R_1)^+}{R_1}\right]^+ + \theta_1\right) + \left[\frac{L_2+\theta_2(p_2-R_1+\rho_1)^+}{R_1-\rho_1}\right]^+ + \theta_2 = 5.477$, $\rho^{eq}_{(3,r_1)} = R_1 - \rho_1 - \rho_2 = 0.84$, and $\sigma^{eq}_{(3,r_1)} = 0$.

$$\Rightarrow \beta^{eq}_{(3,r_1)} = \delta_{5.477} \otimes \gamma_{0,0.84} \tag{16}$$

This Proposition also allows computing the ESC for f_3 in node 2 by subtracting arrival curve of flow f_4, as well. $T^{eq}_{(3,r_2)} = T_2 + \left(\left[\frac{L_4+\theta_4(p_4-R_2)^+}{R_2}\right]^+ + \theta_4\right) = 3.008$, $\rho^{eq}_{(3,r_2)} = R_2 - \rho_4 = 0.992$, and $\sigma^{eq}_{(3,r_2)} = 0$.

$$\Rightarrow \beta^{eq}_{(3,r_2)} = \delta_{3.008} \otimes \gamma_{0,0.992} \tag{17}$$

Since there is no interfering flow in node 3, the ESC of flow 3 in this node is equal to

$$\beta^{eq}_{(3,r_3)} = \sigma_1 \otimes \gamma_{0,1} \tag{18}$$

Step 2: We use the theorem of concatenation of nodes [3] for obtaining the equivalent end-to-end service curve. Given is a flow traversing two nodes sequentially connected and each node is offering a service curve β_i, $i = 1, 2$ to the flow. Then the concatenation of the two nodes offers a service curve of $\beta_1 \otimes \beta_2$ to the flow. Thus, β^{eq}_3 is given by

$$\beta^{eq}_3 = \beta^{eq}_{(3,r_1)} \otimes \beta^{eq}_{(3,r_2)} \otimes \beta^{eq}_{(3,r_3)} \tag{19}$$
$$= \delta_{5.477+3.008+1} \otimes [\gamma_{0,0.84} \wedge \gamma_{0,0.992} \wedge \gamma_{0,1}] = \delta_{9.485} \otimes \gamma_{0,0.84}$$

B. Computation of the end-to-end delay bound

Step 3: According to Proposition 1 and Eq. 19, the maximum delay for flow 3 is bounded by

$$h(\alpha_3, \beta^{eq}_3) = 9.485 \vee \left[\left(\frac{1-0+4.032(1-0.84)^+}{0.84}\right)^+, \left(\frac{1-0+4.032(1-0.992)^+}{0.992}\right)^+, \left(\frac{1-0+4.032(1-1)^+}{1}\right)^+\right]$$
$$= 9.485 + max(1.958, 1.04, 1) = 11.443 \tag{20}$$

Here if we only use (σ, ρ) instead of TSPEC, each flow j would be constrained by arrival curve $\alpha_j = \sigma_j + \rho_j t = \gamma_{\sigma_j,\rho_j}$. Therefore, flows in the example are represented as $f_1 \propto (2, 0.128)$, $f_2 \propto (2, 0.032)$, $f_3 \propto (4, 0.256)$, and $f_4 \propto (2, 0.008)$. We then follow the stages of computing individual delay bound for a tagged flow as stated before. For this purpose, we can easily revise our proposed propositions for (σ, ρ) flows by substituting σ and ρ into L and p, respectively, in all formulas. We can also apply the method presented in [5]. With both approaches, the same value for $h(\alpha_3, \beta^{eq}_3)$ is achieved and equals to 17.241. Thus, we have about 33.6% improvement on the tightness of the delay bound.

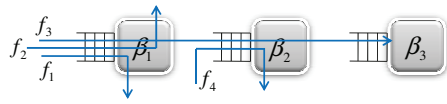


Fig. 4. An example

To analyze delay sensitivity, Table I depicts the end-to-end delay bound for tagged flow f_3 in a network with CBR (Constant Bit-Rate) flows ($Delay_{CBR}$) and also VBR flows ($Delay_{VBR}$) versus the different values of service rate R, along with values for the end-to-end equivalent service rate R^{eq}_3. From this table, it is clear that the end-to-end equivalent service rate, R^{eq}_3, is decreasing by reducing R, while the end-to-end delay bounds are increasing as well. Also, it is worth mentioning that the *improvement percentage (ImP)* decreases because of reduction of R^{eq}_3.

TABLE I
END-TO-END DELAY COMPARISON FOR f_3 UNDER DIFFERENT SERVICE RATES

	$R_1 = 1$	$R_2 = 0.7$	$R_3 = 0.5$
R^{eq}_3	0.84	0.54	0.34
$Delay_{CBR}$	17.241	22.804	31.327
$Delay_{VBR}$	11.443	17.773	27.541
Improvement Percentage	33.6%	22%	12%

VI. CONCLUSIONS

We have presented and proved the required propositions for computing delay bound of VBR flows in a FIFO multiplexing network. The propositions can be applied for an architecture based on aggregate scheduling. To exemplify the potential of our technique, derivation of formulas for computing equivalent service curve and the delay bound is detailed. In the future, we will apply our formal approach for performance analysis of concatenated routers with multiple virtual channels per inport.

ACKNOWLEDGMENT

The research is funded in part by Intel Corporation through a research gift.

REFERENCES

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with h.264/avc: Tools, performance and complexity", *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7-28, 2004.

[2] C. Chang, *Performance Guarantees in Communication Networks*, Springer-Verlag, 2000.

[3] J. Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag, 2004.

[4] Y. Jiang, "Delay bounds for a network of guaranteed rate servers with FIFO aggregation", *Computer Networks*, vol. 40, no. 6, pp. 683-694, 2002.

[5] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks", *Performance Evaluation*, vol. 63, no. 9, pp. 956-987, 2006.

[6] Y. Qian, Z. Lu, and Q. Dou, "QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin", *In the Proceedings of the 28th International Conference on Computer Design (ICCD'10)*, pp. 52-59, Amerstedam, The Netherlands, 2010.

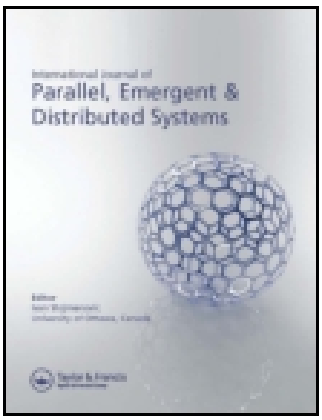
[7] F. Jafari, A. Jantsch, Z. Lu, "Output process of variable bit-rate flows in on-chip networks based on aggregate scheduling", *In the Proceedings of the International Conference on Computer Design (ICCD)*, pp. 445-446, Amherst, USA, 2011.

Paper 11

Proportionally Fair Flow Control Mechanism for Best Effort Traffic in Network-on-Chip Architectures

M. S. Talebi
F. Jafari
A.Khonsari
M. H. Yaghmaee

International Journal of Parallel, Emergent, and Distributed Systems (IJPEDS), Vol. 25, No. 4, pp 345-362 Jul. 2010.



International Journal of Parallel, Emergent and Distributed Systems
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gpaa20

Proportionally fair flow control mechanism for best effort traffic in network-on-chip architectures

Mohammad S. Talebi [a], Fahimeh Jafari [a b], Ahmad Khonsari [a c] & Mohammad Hossien Yaghmaee [b]

[a] School of Computer Science, IPM, Tehran, Iran

[b] Ferdowsi University of Mashhad, Mashhad, Iran

[c] Department of ECE, University of Tehran, Tehran, Iran

Published online: 09 Jul 2010.

To cite this article: Mohammad S. Talebi, Fahimeh Jafari, Ahmad Khonsari & Mohammad Hossien Yaghmaee (2010) Proportionally fair flow control mechanism for best effort traffic in network-on-chip architectures, International Journal of Parallel, Emergent and Distributed Systems, 25:4, 345-362, DOI: 10.1080/17445760902894647

To link to this article: http://dx.doi.org/10.1080/17445760902894647

International Journal of Parallel, Emergent and Distributed Systems
Vol. 25, No. 4, August 2010, 345–362

Proportionally fair flow control mechanism for best effort traffic in network-on-chip architectures

Mohammad S. Talebi[a]*, Fahimeh Jafari[a,b], Ahmad Khonsari[a,c] and Mohammad Hossien Yaghmaee[b]

[a]*School of Computer Science, IPM, Tehran, Iran;* [b]*Ferdowsi University of Mashhad, Mashhad, Iran;* [c]*Department of ECE, University of Tehran, Tehran, Iran*

(*Received 19 January 2009; final version received 11 February 2009*)

The research community has recently witnessed the emergence of multi-processor system on chip (MPSoC) platforms consisting of a large set of embedded processors. Particularly, Interconnect networks methodology based on network-on-chip (NoC) in MPSoC design is imminent to achieve high performance potential. More importantly, many well established schemes of networking and distributed systems inspire NoC design methodologies. Employing end-to-end congestion control is becoming more imminent in the design process of NoCs. This paper presents a centralised congestion control scheme in the presence of both elastic and streaming flow traffic mixture. We model the desired best effort source rates as the solution to an optimisation problem with weighted logarithmic objective which is known to admit proportional fairness criterion. The problem is constrained with link capacities while preserving guaranteed service traffics services requirements at the desired level. We propose an iterative algorithm as the solution to the optimisation problem which has the benefit of low complexity and fast convergence, and can be implemented by a controller unit with low computation and communication overhead.

Keywords: network-on-chip; flow control; best effort; optimisation; proportional fairness

1. Introduction

The high level of system integration characterising multi-processor system-on-chips (MPSoCs) is raising the scalability issue for communication architectures. Towards this direction, traditional system interconnects based on shared busses are evolving both from the protocol and the topology viewpoint. Advanced bus protocols acts in favour of better exploitation of available bandwidth, while more parallel topologies are instead being introduced in order to provide more bandwidth [1]. In the long run, many researchers and SoC designers agree on the fact that this trend approaches the network-on-chip (NoC) as a solution to the lack of SoCs' Scalability [5].

A NoC system fundamentally consists of three components: switches, network interfaces (NIs) and links. The switches can be arbitrarily connected to each other and to NIs, based on a specified topology. They are responsible for routing, switching and flow control logic, as well as error control handling. NIs are responsible for packetisation/de-packetisation and implement the service levels associated with each transaction.

* Corresponding author. Email: mstalebi@ipm.ir

ISSN 1744-5760 print/ISSN 1744-5779 online

DOI: 10.1080/17445760902894647
http://www.informaworld.com

Recently, quality-of-service (QoS) provisioning in NoC's environment has attracted many researchers and currently is the focus of many literature in NoC research community. NoCs are expected to serve as multimedia servers and are required not only to carry elastic flows, i.e. best effort (BE) traffic, but also inelastic flows, i.e. guaranteed service (GS) traffic which requires tight performance constraints such as necessary bandwidth and maximum delay boundaries.

The Internet Engineering Task Force, realising the limitations of the BE model, has undertaken serious steps to meet the QoS demand in the Internet infrastructure. Current achievements in integrating more processor cores on a single chip have made it possible to employ these MPSoCs as real time multimedia servers which require intensive computational power. Thus, it is imperative to provide in MPSoCs, capabilities such as QoS which has been well available in traditional Internet servers. This implies that the underlying on-chip communication will be required to provide deterministic bounds on delay and throughput for communication among communicating nodes on a chip. Congestion control as a critical means of providing QoS in traditional data networks is a well known issue and has been widely studied over the past two decades. However, it is still a novel problem in NoCs and to the best of our knowledge only few works has been carried out in this field [8,12,16].

Network congestion has a negative effect on its performance. The problem occurs in networks when resources, i.e. available bandwidths, get saturated. The resulting performance degradation is experienced by BE network users as an increase of latency and loss of bandwidth. Figure 1 shows a shared link transporting both constant bitrate BE traffic and variable bitrate (VBR) GS traffic with the reserved bandwidth depicted with a dashed line. BE traffic improves resource utilisation but at certain moments the shared resource is congested. Figure 2 shows a shared resource with congestion controlled BE and VBR GS traffic.

This paper is organised as follows. We discuss related work in Section 2. In Section 3 we present the system model and formulate the underlying optimisation problem for BE flow control. In Section 4 we solve the underlying optimisation problem using duality approach and propose a flow control algorithm. In Section 5 we analyse the performance of the proposed algorithm in terms of convergence behaviour and fairness. Section 6

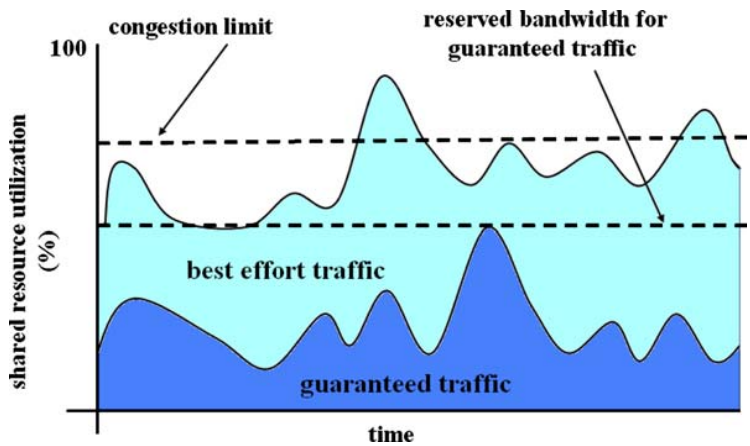


Figure 1. Shared resource without congestion controlled BE.

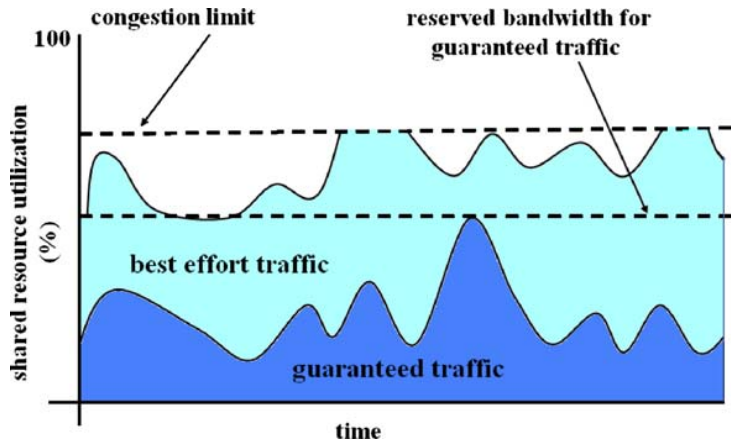


Figure 2. Shared resource with congestion controlled BE.

addresses the realisation aspects of the proposed flow control algorithm. Section 7 presents the simulation results. Finally, Section 8 concludes the paper and states some future work directions.

2. Related works

Flow control for data networks is a widely studied issue [7,9,11,17]. A wide variety of flow control mechanisms in data network belongs to the class of end-to-end control schemes, like TCP/IP, which is mainly based on the window-based protocols. In these protocols, intermediate routers avoid the network from becoming congested by means of packet dropping deterministically (as in DropTail) or randomly (as in RED). Therefore, transmitted packets are subject to loss and the network must aim to provide an acknowledgement mechanism. On the other hand, on-chip networks pose different challenges. The reliability of on-chip wires and more effective link-level flow-control allows NoCs to be lossless. Therefore, there is no need to utilise an acknowledgment mechanism and we face to a slightly different concept of flow control.

So far, several works have focused on this issue for NoC systems. In Ogras and Marculescu [12] a prediction-based flow-control strategy for on-chip architectures is proposed in which each router predicts the buffer occupancy to sense congestion. This scheme controls the packet injection rate and regulates the number of packets in the network. In van den Brand et al. [16] link utilisation is used as a congestion measure and a model prediction-based controller, determines the source rates. DyAD [8] controls the congestion by using adaptive routing when the NoC faces congestion.

In this paper, we focus on the flow control for BE traffic as the solution to a utility-based optimisation problem. To the best of our knowledge, none of the abovementioned works have dealt with the flow control problem using a utility optimisation approach. In our previous work [13], we have modelled desired BE source rates as the solution to a utility-based optimisation problem with a general form utility function and solved the proposed problem using Newton's method. In Talebi et al. [14], we focused this problem via sum-rate optimisation problem and used a different approach to solve the problem. In Talebi et al. [15] we have used the flow control problem outlined in Talebi et al. [13] and focused on a especial form utilityfunction and adopted a different approach to solve the

problem. As an extension to our previous work [15], in this paper we address the performance analysis of the flow control problem outlined in Talebi et al. [13] for a especial form of utility functions, known as *proportional utility functions*, which satisfies nice fairness features. We focus on the solution of the flow control problem and investigate its fairness and convergence behaviour.

3. System model and problem formulation

We consider a NoC architecture with wormhole routing. In wormhole-routed networks, each packet is divided into a sequence of *flits* which are transmitted over physical links one by one, in a pipeline fashion. A hop-to-hop credit mechanism guarantees that a flit is transmitted only when the receiving port has free space in its input buffer. We also assume that the NoC architecture is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing [5].

We model the flow control in NoC as the solution to a utility-based optimisation problem. For the sake of convenience, we turn the aforementioned NoC architecture into a mathematically modelled network, as in Low and Lapsley [10]. In this respect, we consider NoC as a network with a set of bidirectional links $\mathcal{L} = \{1, 2, \ldots, L\}$ and a set of sources $\mathcal{S} = \{1, 2, \ldots, S\}$. A source consists of processing elements, routers and input/output ports. Each link $l \in \mathcal{L}$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of c_l bits/s. We denote the set of sources that share link l by $\mathcal{S}(l)$. Similarly, the set of links that source s passes through, is denoted by $\mathcal{L}(s)$. By definition, $s \in \mathcal{S}(l)$ if and only if $l \in \mathcal{L}(s)$.

As discussed in Section 1, there are two types of traffic in a NoC: GS and BE. For notational convenience, we represent BE and GS traffic rates by x_s and y_s, respectively. Each link $l \in \mathcal{L}$ is shared between the two traffics. GS traffics will obtain the required amount of link capacity and BE traffics benefit from the remainder.

Our objective is to choose source rates with BE traffic so as to maximise the weighted sum of the logarithm of the BE source rates while satisfying capacity constraints. Thus, the optimisation problem can be formulated as [10]:

$$\max_{x_s} \sum_{s \in \mathcal{S}} a_s \log x_s \tag{1}$$

subject to:

$$\sum_{s \in \mathcal{S}(l)} x_s + y_s \leq c_l \quad \forall l \in \mathcal{L} \tag{2}$$

$$x_s > 0 \quad \forall s \in \mathcal{S} \tag{3}$$

where a_s is the positive weight of source s. Optimisation variables are BE rates, which in vector form are represented by $\mathbf{x} = (x_s, s \in \mathcal{S})$ and belong to $\Re_+^S$. ($\Re_+^S$ denotes nonnegative real).

The constraint (2) states that the sum of BE and GS traffic rates passing through link l cannot exceed its free capacity, i.e. c_l. The objective function of problem (1) is convex and its constraints are affine, and hence it is a convex optimisation problem with linear constraints and admits a unique maximiser [2,4]; i.e. there exists an optimal source rate vector, $\mathbf{x}^*$, which maximises (1) while satisfying capacity constraints.

In general, problem (1) belongs to the class of utility-based optimisation problems, for which the utility function is assumed to be weighted logarithmic, i.e. $U_s(x_s) = a_s \log x_s$. Such utility functions, are assumed to be concave and strictly increasing. There are many choices for such utility functions with specific features and behaviours. We discuss in Section 5, that logarithmic utility function has nice properties known as *proportional fairness* [9].

It is worth mentioning that despite the restriction of ourselves to a specific utility function, our work can be easily extended to arbitrary utility functions, as in our previous work [13].

For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in \mathcal{S}(l)} y_s. \tag{4}$$

Also, for the sake of simplicity in our derivations throughout this paper, we define the routing matrix as $\mathbf{R} = [R_{ls}]_{L \times S}$, where R_{ls} is defined as

$$R_{ls} = \begin{cases} 1 & \text{if } l \in \mathcal{L}(s) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Thereafter, we will follow this notation, unless the otherwise is stated. Regarding this, (1) for the aforementioned class of utility functions can be rewritten as

$$\max_{x_s} \sum_{s \in \mathcal{S}} a_s \log x_s \tag{6}$$

subject to:

$$\sum_s R_{ls} x_s \leq \hat{c}_l \quad \forall l \in \mathcal{L} \tag{7}$$

$$x_s > 0 \quad \forall s \in \mathcal{S} \tag{8}$$

4. Optimal flow control algorithm

In this section we solve (6) and derive the optimal flow control algorithm.

Although problem (6) is separable among sources, its constraints will remain coupled across the links over the network. The coupled nature of such constrained problems, necessitates usage of centralised methods like *interior point method* which poses great computational overhead on the system [2,4].

One way to reduce the computational complexity is to transform the constrained optimisation problem into an unconstrained one, which can be solved efficiently using several iterative methods. According to the *duality theory* [2,4], each convex optimisation problem has a *dual problem*. Regarding this terminology, the main problem is retroactively called *the primal problem*. Optimal solution of the dual for a maximisation (minimisation) problem leads to an upper bound (lower bound) to the optimal value of the primal. With certain conditions (such as strong convexity) such an upper bound (lower bound) is tight and hence solving the dual is equivalent to solving the primal [4]. However, as the dual problem can be defined in a way to be unconstrained, solving the dual is much simpler than the primal.

In the sequel, we will obtain the dual of problem (6) and solve it using efficient iterative algorithms.

4.1 Deriving the dual

We start by writing the Lagrangian of (6). Using the standard optimisation methods [4], the Lagrangian of problem (6) can be written as:

$$L(\mathbf{x}, \lambda) = \sum_s a_s \log x_s - \sum_l \lambda_l \left(\sum_{s \in \mathcal{S}} R_{ls} x_s - \hat{c}_l \right), \tag{9}$$

where λ_l is the positive Lagrange multiplier associated with the corresponding constraint of link l and $\lambda = (\lambda_l, l \in \mathcal{L})$ is the vector of Lagrange multipliers and belongs to $\Re_+^L$. In economics literature, λ_l is called *shadow price* [9] for the interpretation of its role in solving the primal problem via its dual. Later on, we will discuss about this issue.

Regarding the Lagrangian, the dual function is defined as [4]:

$$g(\lambda) = \max_{\mathbf{x}} L(\mathbf{x}, \lambda). \tag{10}$$

Duality theory states that when the duality gap[1] is zero, the optimal source rate vector, $\mathbf{x}^*$, corresponds to the optimal Lagrange multipliervector, λ^* [2,4]. In other words, if $\mathbf{x}$ is a feasible point of the primal problem and $\mathbf{x}$ is primal-optimal, the corresponding λ will be dual-optimal and vice versa. Therefore, at optimality using Karush–Kuhn–Tucker (KKT) condition, we have

$$\nabla_x L(\mathbf{x}, \lambda)|_{(\mathbf{x}^*, \lambda^*)} = 0, \tag{11}$$

where $\mathbf{0}$ is a vector with all zero. From (9), we have

$$\frac{\partial L}{\partial x_s}|_{(\mathbf{x}^*, \lambda^*)} = \frac{\mathrm{d}}{\mathrm{d}x_s}(a_s \log x_s)|_{x_s^*} - \sum_l R_{ls} \lambda_l^* = 0. \tag{12}$$

Hence, the optimal source rate is given by

$$x_s^* = \frac{a_s}{\sum_l R_{ls} \lambda_l^*}. \tag{13}$$

From (13) it is apparent that x_s^* is a decreasing function of λ_l; therefore λ_l can be construed as the price which must be paid for the source rate x_s. As the nature of such a price is hidden to the sources from the primal problem perspective, it is called *shadow price*. Substituting x_s^* into (9) yields

$$g(\lambda) = \sum_s \left(a_s(\log a_s - 1) - a_s \log \left(\sum_l R_{ls} \lambda_l \right) \right) \tag{14}$$

$$+ \sum_l \lambda_l \hat{c}_l. \tag{15}$$

The dual problem is defined as [4]:

$$\min_{\lambda_l} g(\lambda) \tag{16}$$

subject to:

$$\lambda_l \geq 0 \quad \forall l \in \mathcal{L}. \tag{17}$$

Therefore, the dual problem is given by:

$$\min_{\lambda_l \geq 0} \sum_l \lambda_l \hat{c}_l - \sum_s a_s \log\left(\sum_l R_{ls}\lambda_l\right). \tag{18}$$

The dual problem is always convex regardless of convexity or non-convexity of the primal problem. Moreover, the dual problem can be defined to be unconstrained. Thus, the primal has been transformed into an unconstrained convex optimisation problem.

Strict convexity of the primal problem (6) guarantees strong duality. Therefore the *duality gap* is zero; i.e. solving the dual leads to the optimal point of the primal [2,4]. Since dual problem is convex, it admits a unique minimiser, which can be obtained using iterative methods. As the dual problem is unconstrained; solving (18) using iterative methods is much simpler than the primal.

There exist several methods to search the optimal point of an unconstrained optimisation problem iteratively [2,4]. One famous and simple ones is *gradient projection method* [2] which admits tractable computational complexity. Another famous one is *Newton's method* that has better convergence behaviour at the expense of higher computational complexity [2,4].

We postpone solving the dual to the next subsection.

4.2 Solving the dual

In this subsection, we solve the dual problem using *gradient projection method* [2].

The gradient projection method adjusts shadow prices, i.e. Lagrange multipliers, in opposite direction to the gradient of the dual function, as follows:

$$\lambda(k+1) = [\lambda(k) - \gamma \nabla g(\lambda(k))]^+, \tag{19}$$

where $\gamma > 0$ is a sufficiently small constant stepsize, and $[z]^+ = \max\{z, 0\}$. Since the objective of problem (6) is strictly concave, $g(\lambda)$ is continuously differentiable [2], and thus $\nabla g(\lambda)$ exists. Using (14), the lth element of the gradient vector is given by:

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \frac{\partial}{\partial \lambda_l}\left[\left(\sum_l \lambda_l \hat{c}_l - \sum_s a_s \log \sum_l R_{ls}\lambda_l\right)\right]. \tag{20}$$

Therefore,

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \hat{c}_l - \sum_s \frac{R_{ls} a_s}{\sum_k R_{ks}\lambda_k}. \tag{21}$$

Regarding (13), (21) can be written as

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \hat{c}_l - \sum_s R_{ls} x_s \tag{22}$$

and the update equation is given by:

$$\lambda_l(k+1) = \left[\lambda_l(k) - \gamma\left(\hat{c}_l - \sum_s R_{ls} x_s(\lambda)\right)\right]^+. \tag{23}$$

where $x_s(\lambda(k))$ is obtained by (13), given $\lambda(k)$. In the next subsection, we propose a flow control algorithm based on the update Equation (23).

4.3 Optimal algorithm

In this subsection, we present a centralised flow control algorithm for BE traffic in NoC systems which controls the BE source rates in favour of problem (6). Regarding (23) and (13), it is clear that they form an iterative algorithm as the solution to problem (18) and thereby problem (6). In this respect, optimal source rates for BE sources can be found while satisfying capacity constraints and preserving GS traffic requirements. Thus, such an algorithm can be used to control the flow of BE sources in the NoC. This algorithm is listed below as Algorithm 1.

The above iterative algorithm has a decentralised nature and can also be addressed in distributed scenarios. However, due to well-formed structure of the NoC, we focus on a centralised scheme; a controller can be devised to implement such an algorithm. The necessary requirements of such a controller are the ability to accomplish simple mathematical operations as in (23) and (13) and the allocation of few dedicated links to communicate congestion control information to nodes with a light real-time load. Later, in Section 6 we will discuss about the implementation aspects of such a controller.

Algorithm 1. Fair BE flow control in NoC

Initialisation
Initialise the following items:
1. Sets of sources and links including the routing matrix.
2. $\hat{c}_l$ for $l \in \mathcal{L}$.

Main loop
Do until $\max_s |x_s(k+1) - x_s(k)| < \epsilon$
1.
$\forall l \in \mathcal{L}$ Compute new link prices:
$\lambda_l(k+1) = \left[\lambda_l(k) - \gamma\left(\hat{c}_l - \sum_l R_{ls} x_s(k)\right)\right]^+$
2.
$\forall s \in \mathcal{S}$ Compute new BE source rates as follows:
$x_s(k+1) = \frac{a_s}{\sum_l R_{ls}\lambda_l(k)}$

Output
Communicate BE source rates to the corresponding sources.

5. Performance analysis

In this section, we go into more detail of the performance of the proposed flow control algorithm. First, we analyse the convergence behaviour of it and then, we explore its fairness features.

5.1 Convergence analysis

In this subsection, we investigate the convergence behaviour of the proposed algorithm. Evolution of the proposed flow control algorithm is mainly relying on the update Equation (23), which in turn is dependent on the stepsize γ, as a parameter. Thus, we mainly focus on the effect of stepsize and state the conditions under which Algorithm 1 converges.

There are several choices for stepsize, each one belonging to a predefined category and having certain advantages and drawbacks (see [2] and references herein). In the family of gradient projection algorithms for distributed scenarios, stepsize is usually chosen to be a small enough constant so as to guarantee the convergence of the algorithm. Due to its simplicity and robustness, in this paper we focus on the case of constant stepsize.

Before proceeding to state the necessary conditions, we first present the fundamental lemma for the gradient optimisation algorithms.

LEMMA 5.1. Consider the unconstrained minimisation problem $\min_x f(x)$ with its minimum point denoted by x^*. If $\nabla f(x)$ has Lipschitz continuity property, i.e. there exist $M > 0$ such that

$$|\nabla f(x_1) - \nabla f(x_2)| \leq M\|x_1 - x_2\|_2 \tag{24}$$

then the sequence $x(k)$ defined as

$$x(k+1) = x(k) - \gamma \nabla f(x(k)) \tag{25}$$

converges to the neighbourhood of x^* provided that the following hold

$$\epsilon \leq \gamma \leq \frac{2-\epsilon}{M} \tag{26}$$

for some values of $\epsilon > 0$.

Proof: See [2] for proof.

The following theorem states the necessary condition on the stepsize, under which Algorithm 1 converges to the neighbourhood of the optimal point of problem (18) and thereby that of problem (6).

THEOREM 5.2. The iterative flow control algorithm proposed by (13) and (23) converges to the neighbourhood of the optimal point of the primal problem (6) provided that

$$0 < \gamma < \frac{2\underline{a}}{\bar{c}^2 \bar{L}\bar{S}}, \tag{27}$$

where $\bar{L}$ is the length of the longest path used by sources, $\bar{S}$ is the number of sources sharing the most congested link, $\underline{a}$ is the minimum weight of sources and $\bar{c}$ is the upper bound on link capacities.

Proof: See Appendix 1 for proof.

5.2 Proportional fairness

The choice of utility function directly influences the policy by which system resources, i.e. link bandwidth, are shared among competing sources. In this respect, in terms of economics terminology, utility function maintains a specific criterion of *fairness* among users or sources. Several fairness criteria have been defined in literature which can serve as the objective function in problem (6). Amongst them are *Max–Min Fairness* [3] and *Proportional Fairness* [9]. In networks with Max–Min fairness, resources are mainly shared in favour of weak users while in those with Proportional Fairness, resources are shared in proportion to the resource usage of each source. In the sequel the formal definition of Proportional Fairness is stated.

DEFINITION 1. (*Proportional Fairness* [9]) The optimal rate allocation $\mathbf{x}^* = (x_s^*, s \in \mathcal{S})$, is said to be *proportionally fair*, if for any other feasible rate allocation, say $\mathbf{x}' = (x_s', s \in \mathcal{S})$, the total proportional net benefit gained by the new source rates is decreased, i.e.

$$\sum_s \frac{x_s' - x_s^*}{x_s^*} \leq 0. \tag{28}$$

It is proven that systems with proportional fairness, i.e. those satisfying (28), must have logarithmic utility functions [9], i.e.

$$U_s(x_s) = \log x_s. \tag{29}$$

Thus, the proposed flow control algorithm, with equal weight factors will be proportionally fair. It is worth noting that the case of heterogeneous weight factors corresponds to another implementation of such a fairness criterion, the so-called *Weighted Proportionally Fair*, for which (28) turns to be

$$\sum_s a_s \frac{x_s' - x_s^*}{x_s^*} \leq 0 \tag{30}$$

and the corresponding utility function will be

$$U_s(x_s) = a_s \log x_s. \tag{31}$$

In the sequel, we briefly discuss about the effects of weight factors. As previously stated, a_s is the weight for source s in the optimisation problem which controls its priority in resource sharing. To gain more insights on the role of a_s in the flow control, we consider a simple case in which there is only a single bottleneck link, say link $k \in \mathcal{L}$. By a bottleneck link, say link k, we mean a link for which

$$\sum_s R_{ks} x_s^* = c_k \tag{32}$$

KKT conditions guarantee that in the optimality (i.e. equilibrium) the Lagrange multiplier associated to such a link is not zero. Furthermore, since all other links does not saturate, we have

$$\sum_s R_{ls} x_s^* < c_l \quad l \neq k. \tag{33}$$

Likewise, KKT conditions guarantee that in equilibrium, for such links we get

$$\lambda_l^* \left(\sum_s R_{ls} x_s^* - c_l \right) = 0 \quad l \neq k. \tag{34}$$

Using (33), we deduce that

$$\lambda_l^* = 0 \quad l \neq k. \tag{35}$$

We now proceed to see how bandwidth of the bottleneck link is shared among competing sources $s \in \mathcal{S}(k)$. Combining (13) and the above results, we have:

$$x_s = \frac{a_s}{\sum_{l \in \mathcal{L}(s)} \lambda_l}$$

$$= \frac{a_s}{\lambda_k} \tag{36}$$

$$\frac{x_i}{a_i} = \frac{x_j}{a_j} = \frac{1}{\lambda_k} \quad \forall i, j \in \mathcal{S}(k) \tag{37}$$

combining (36) and (37), leads to

$$x_i = \frac{a_i}{\sum_{s \in \mathcal{S}(k)} a_s} c_k. \tag{38}$$

Equation (38) offers a simple proportional rule for rate allocation in the abovementioned scenario. It says that in a network with single bottleneck link, the sources passing through the congested link, achieve their rates in proportion to their weights. For networks with multiple congested links, such an insight might not be easily seen, however, weight factors directly influence the capacity sharing at bottleneck links, similarly. In this respect, we draw a conclusion that more resources, i.e. link capacity, can be allocated to some specified sources by assigning larger weights to them.

6. Realisation aspects

6.1 Implementation

In this subsection we address the implementation aspects of the proposed BE flow control algorithm.

As stated earlier, Algorithm 1 can be used as a centralised flow control mechanism for BE sources in NoC. In this regard, we consider a simple controller that can be embodied by the NoC, whether as a separate hardware module or as a part of its operating system, which is responsible for running the algorithm. From computational complexity point of view, such a controller must have the ability of carrying out simple mathematical and logical operations, as in Algorithm 1. Another issue worth considering is the mechanism with which the controller communicates with sources. Since we would like source rate information being communicated without delay and loss, we designate to it several GS links in conjunction with all sources with light traffic load. This can be implemented as a control bus, to communicate the algorithm output to BE sources.

Such a controller, even if implemented as a separate hardware module, as in van den Brand et al. [16], would admit very low power consumption and as a result would pose negligible overhead to the system.

6.2 Comparison with congestion control in data networks

Motivated by the end-to-end nature of Algorithm 1, we briefly discuss about the inherent connection of it with those used for BE data transmission in the Internet.

The proposed flow control algorithm are very similar to end-to-end congestion control schemes in data networks, e.g. TCP variants which are widely used to control BE data flow in the Internet. Most of such end-to-end schemes use the well-known window-based method, in which each source maintains a window of packets that are transmitted, but not acknowledged. In data networks, packets may be lost due to dropping at the routers, and therefore destination should acknowledge the ordered receipt of them in the current window. Each source changes its window size in response to congestion signals, i.e. positive or negative acknowledges or duplicates ones, and thereby avoids the network facing congestion. Roughly speaking, the source rate in each round trip (i.e. the way from a source to its destination and then back to the source for acknowledgement), is the ratio of the window size to the Round Trip Time (RTT) (i.e. duration of the trip).

Although flow control in TCP is carried out through updating window size, one can derive the corresponding rate updates, too. The proposed flow control algorithm is very similar to rate update in TCP scheme. Such a close connection stems from the similarity in the underlying flow control problem in both schemes. However, it is worth mentioning that unlike TCP, in Algorithm 1 we have not devised any window-based transmission and acknowledgment mechanism. This is due to the fact that NoC architecture is lossless, as previously stated in Section 2, and hence all packets will be delivered successfully in the correct order and therefore no acknowledgement is needed.

7. Simulation results

In this section we examine the proposed flow control algorithm for a typical NoC architecture. In our scenario, we have used a NoC with 4×4 Mesh topology which consists of 16 nodes communicating using 24 shared bidirectional links; each one having a fixed capacity of 1 Gbps. In our scheme, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bits long.

In order to simulate our scheme, some nodes are considered to have a GS data (such as Multimedia, etc.) to be sent while other nodes have a BE traffic. As stated before, GS sources will obtain the required amount of the link capacities and the remainder should be allocated to BE traffics. Routing policy for BE sources is shown in Figure 3. We present our results in the following subsections as below.

7.1 Convergence behaviour

One of the most significant issues of interest is the convergence behaviour of the source rates. In this subsection, we have simulated our scheme using two different values for step-size, $\gamma = 1.05$ and 0.2. Weight factors for all sources are assumed to be unity. The convergence behaviour of source rates using the two abovementioned choices of step size are depicted in Figures 4 and 5. Regarding Figure 4, it's apparent that for $\gamma = 1.05$, after

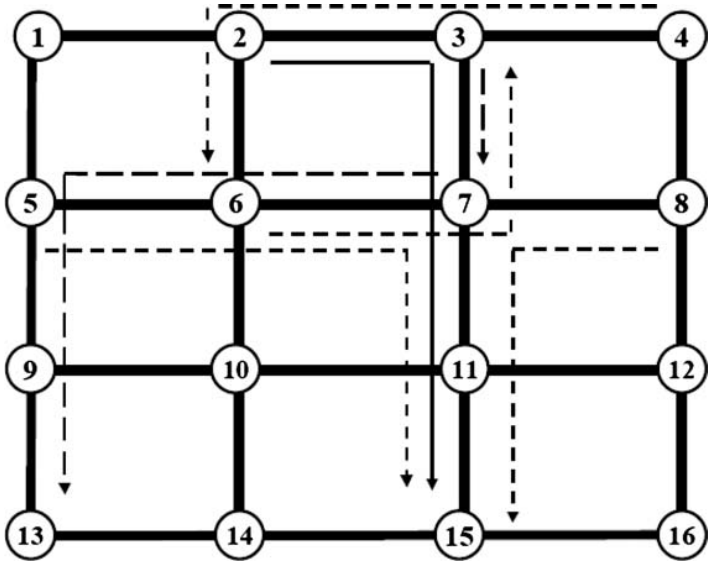


Figure 3. Network topology and routing policy.

20 iteration steps the source rates will have very little variations, however, from Figure 5, i.e. for $\gamma = 0.2$, such a threshold of iteration steps will be at least 85.

In order to have a better insight about the convergence behaviour of the algorithm, the relative error with respect to optimal source rates which is averaged over all active sources, is also depicted in Figure 6. Optimal source rates are obtained using CVX [6] which is a MATLAB-based software for solving disciplined convex optimisation problems. This figure reveals that using the first step size leads to less than 10% error in average just after running about 13 iteration steps, and after 20 steps the average error lies below 5%. However, with the second step size, the algorithm would reach the two aforementioned error margins at the expense of iterating for about 60 and 75 steps,

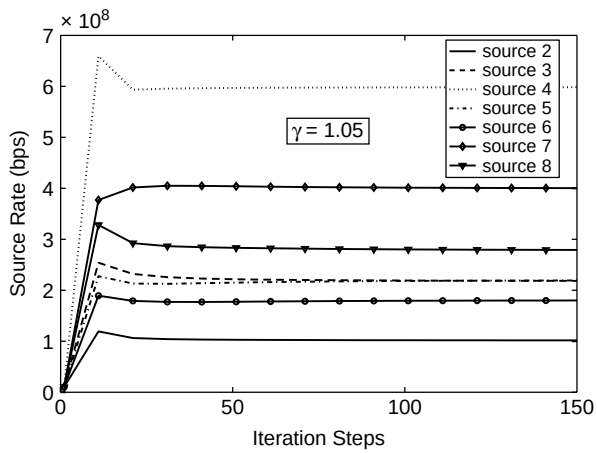


Figure 4. Source rates convergence for $\gamma = 1.05$.

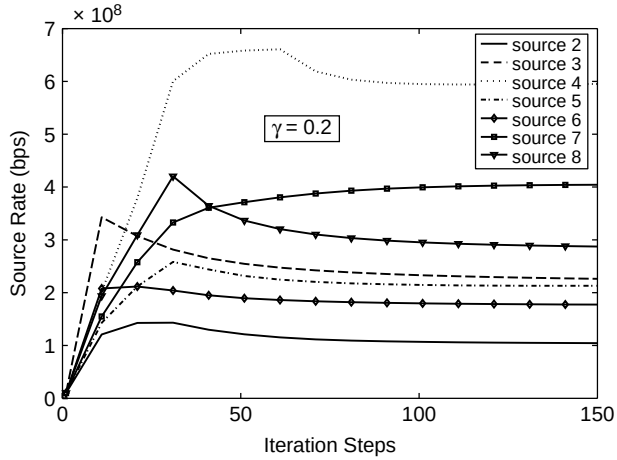


Figure 5. Source rates convergence for $\gamma = 0.2$.

respectively. For practical implementations and realistic applications, due to faster convergence speed, the first step size is more appropriate.

7.2 Convergence behaviour in dynamic scenarios

Although we have not considered the tracking feature of Algorithm 1, such a Gradient based algorithm has nice properties to track the dynamic conditions of the network. Such an ability of tracking the dynamic conditions emanates from the tracking capability of the gradient operator. In order to investigate the behaviour of the algorithm to track the conditions, we consider the following scenario: we assume that the network in the previous subsection with the routing policy as in Figure 3. At the iteration step 140, source 1 is activated and starts sending data. In such a case, the constraint (2) would change and therefore the optimal solution to the problem would be altered, as well. However, there is no need to restart the algorithm form its initial phase. The proposed flow control algorithm can track such a dynamic changes and has the capability to move towards the new optimal

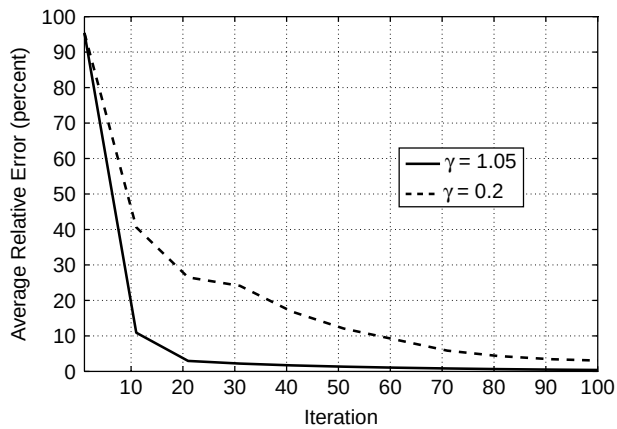


Figure 6. Average relative error.

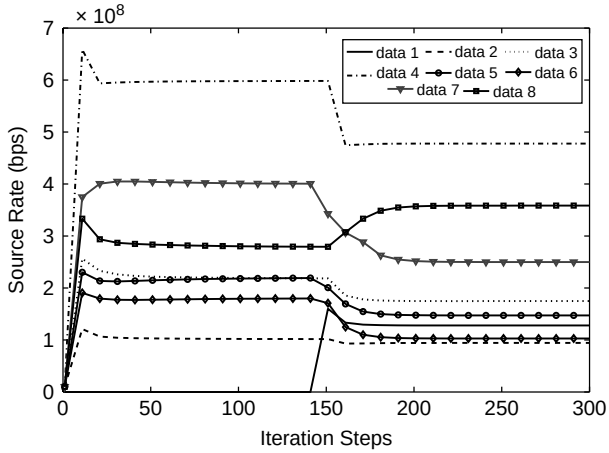


Figure 7. Source rate convergence in a time-varying scheme.

source rates without restarting from its initial points. Convergence behaviour of the source rates is depicted in Figure 7. It is apparent that after step 140, using the chosen step size, just as few as 20 iteration steps will suffice to move towards the new optimal source rates. Thus, the proposed algorithm would perform satisfactorily in a time-varying environment with and abrupt real-time changes.

7.3 Effect of the weight

Another case we consider is the role of weight factor on resource (link capacity) sharing. It is trivial that network shares its resources in favour of sources with larger weight factors. In the next simulation experiment, we set the weight factor of sources 2 and 7−20. Convergence behaviour and steady state source rates are shown in Figure 8. Comparing Figure 8 and Figure 5, we realise that using larger weight factors, sources 2 and 7 have

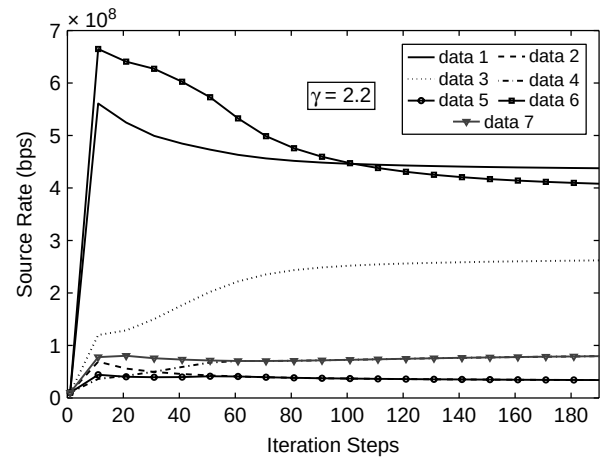


Figure 8. Source rate convergence for asymmetric weight factors.

achieved larger rates; however this is done at the expense of reducing the rate of some other nodes passing through bottleneck links that sources 2 and 7 were passing. It is also worth mentioning that such an asymmetric case, adversely influences the speed of convergence.

8. Conclusion

In this paper, we addressed the problem of flow control for BE traffic in NoC systems. Flow control was modelled as the solution to an optimisation problem whose objective was the sum of weighted logarithmic functions. We solved the problem indirectly through its dual using gradient projection method, which was led to a flow control algorithm that can be used to determine optimal BE source rates. Moreover, we evaluated the performance of the proposed algorithm from two aspects: first we investigated its convergence behaviour and proved that under certain condition, the algorithm would converge towards (very close vicinity of) optimal point, and second we lightened that this algorithm admits proportional fairness criterion. Finally, we argued that the proposed algorithm can be efficiently implemented by a controller unit which poses a light computation and communication overhead to the system.

Note

1. *Duality gap* is referred to as the difference between the optimal value of primal and dual problem.

References

[1] L. Benini and G. DeMicheli, *Networks on chips: A new SoC paradigm*, IEEE Comput. 35(1) (2002), pp. 70–78.
[2] D. Bertsekas, *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA, 1999.
[3] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
[4] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, 2004.
[5] W.J. Dally and B. Towles, *Route packets, not wires: On-chip interconnection networks*, Design Automation Conference (2001), pp. 684–689.
[6] M. Grant, S. Boyd, and Y. Ye, *CVX (Ver. 1.0RC3): MATLAB software for disciplined convex programming*, Download available at: http://www.stanford.edu/boyd/cvx.
[7] Y. Gu, H.O. Wang, and Y. Hong, *A predictive congestion control algorithm for high speed communication networks*, Am. Control Conf. 5 (2001), pp. 3779–3780.
[8] Hu. Jingcao and R. Marculescu, *DyAD-smart routing for networks-on-chip*, Design Automation Conference (2004), pp. 260–263.
[9] F.P. Kelly, A. Maulloo, and D.K.H. Tan, *Rate control for communication networks: Shadow prices, proportional fairness, and stability*, Oper. Res. Soc. 49(3) (1998), pp. 237–252.
[10] S.H. Low and D.E. Lapsley, *Optimization flow control I: Basic algorithm and convergence*, IEEE/ACM Trans. Netw. 7 (1999), pp. 861–875.
[11] S. Mascolo, *Classical control theory for congestion avoidance in high-speed internet*, Conf. Decis. Control 3 (1999), pp. 2709–2714.
[12] U.Y. Ogras and R. Marculescu, *Prediction-based flow control for network-on-chip traffic*, Design Automation Conference (2006), pp. 839–844.
[13] M.S. Talebi, F. Jafari, and A. Khonsari, *A novel flow control scheme for best effort traffic in NoC based on source rate utility maximization*, in *Proceeding of the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (2007), pp. 381–386.
[14] M.S. Talebi, F. Jafari, A. Khonsari, and M.H. Yaghmaee, *A novel congestion control scheme for elastic flows in network-on-chip based on sum-rate optimization*, International Conference on Computational Science and its Applications (2007), pp. 398–409.

[15] M.S. Talebi, F. Jafari, A. Khonsari, and M.H. Yaghmaee, *Proportionally-fair best effort flow control in network-on-chip architectures*, International Parallel and Distributed Processing Symposium (2008), pp. 1–8.
[16] J.W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, *Congestion-controlled best-effort communication for networks-on-chip*, Design, Automation and Test in Europe Conference (2007), pp. 948–953.
[17] C. Yang and A.V.S. Reddy, *A taxonomy for congestion control algorithms in packet switching networks*, IEEE Netw. 9(4) (1995), pp. 34–45.

Appendix 1. Proof of Theorem 1

We adopt the framework of the proof from Low and Lapsley [10] and briefly recollect the results from it. According to the duality theory, whenever the strong duality is certified, the duality gap is zero and the optimal point of the dual leads to the that of primal. Thus, it suffices to seek the conditions on the stepsize under which (23) converges to a neighbourhood of the dual-optimal point.

By Lemma 1, it is clear that to prove the convergence of Algorithm 1, we should find a constant M to satisfy Lipschitz condition. Also by Lemma 1, $\nabla g(\lambda)$ should admit the Lipschitz Continuity property and thereby it suffices to show that the Hessian of $g(\lambda)$ is upper bounded in l_2-norm. The Hessian of $g(\lambda)$ is a matrix $\mathbf{H} = [H_{ij}]_{L\times L}$, where H_{ij} is defined as

$$H_{ij} = \frac{\partial^2 g(\lambda)}{\partial \lambda_i \partial \lambda_j}. \tag{39}$$

Considering (13), we have

$$\frac{\partial x_s(\lambda)}{\partial \lambda_l} = -R_{ls}\frac{x_s^2}{a_s} \tag{40}$$

(40) can be rewritten in the matrix form as

$$\frac{\partial \mathbf{x}(\lambda)}{\partial \lambda} = -\mathbf{A}\mathbf{R}^{\mathrm{T}}, \tag{41}$$

where

$$\mathbf{A} = \mathrm{diag}\left(-\frac{x_s^2}{a_s}, s \in \mathcal{S}\right). \tag{42}$$

Recall that in matrix form, we can rewrite (22) as

$$\nabla g(\lambda) = \hat{\mathbf{c}} - \mathbf{R}\mathbf{x} \tag{43}$$

hence the Hessian of $g(\lambda)$ is given by

$$\begin{aligned}\mathbf{H} &= \nabla^2 g(\lambda)\\ &= -\mathbf{R}\left[\frac{\partial x(\lambda)}{\partial \lambda}\right]\\ &= \mathbf{R}\mathbf{A}\mathbf{R}^{\mathrm{T}}.\end{aligned} \tag{44}$$

To find the upper bound of the Hessian, we use the following inequality [4]:

$$\|\mathbf{H}\|_2 \le \|\mathbf{H}\|_1 \|\mathbf{H}\|_\infty, \tag{45}$$

where $\|\mathbf{H}\|_1$ is the maximum column-sum matrix norm of $\mathbf{H}$, and $\|\mathbf{H}\|_\infty$ is the maximum row-sum

matrix norm. for $\|\mathbf{RAR}^{\mathrm{T}}\|_\infty$ we have:

$$\begin{aligned}\|\mathbf{RAR}^{\mathrm{T}}\|_\infty &= \max_i \sum_j \left|\frac{\partial^2 g(\lambda)}{\partial\lambda_i \partial\lambda_j}\right| \\ &= \max_i \sum_j |[\mathbf{RAR}^{\mathrm{T}}]_{ij}| \\ &= \max_i \sum_j \sum_s \left| R_{is} R_{js} \frac{x_s^2}{a_s} \right| \\ &= \max_i \sum_s \left| R_{is} \frac{x_s^2 |\mathcal{L}(s)|}{a_s} \right|, \end{aligned} \tag{46}$$

where $|\mathcal{L}(s)|$ represents the number of links in the path of source s. Source rates are upper bounded as

$$\max x_s \le \max_l \hat{c}_l \le \max_l c_l. \tag{47}$$

Regarding the statement of the Theorem 1, we define:

$$\max_s |\mathcal{L}(s)| = \bar{L} \tag{48}$$

$$\max_l |\mathcal{S}(l)| = \bar{S} \tag{49}$$

$$\max_l c_l = \bar{c} \tag{50}$$

$$\min_s a_s = \underline{a}. \tag{51}$$

Hence, we have

$$\|\mathbf{H}\|_\infty \le \frac{\bar{c}^2 \bar{L}\bar{S}}{\underline{a}}. \tag{52}$$

Symmetry of $\|\mathbf{H}\|_\infty$ results in equality of $\|\mathbf{H}\|_\infty$ and $\|\mathbf{H}\|_1$. Therefore, Hessian is upper bounded at least as follows:

$$\|\mathbf{H}\|_2 \le \|\mathbf{H}\|_\infty \le \frac{\bar{c}^2 \bar{L}\bar{S}}{\underline{a}}. \tag{53}$$

Therefore, for sufficiently small ϵ, from Lemma 1 we conclude

$$0 < \gamma < \frac{2\underline{a}}{\bar{c}^2 \bar{L}\bar{S}}, \tag{54}$$

which completes the proof.

Paper 12

Buffer Optimization in Network-on-Chip through Flow Regulation

F. Jafari
Z. Lu
A. Jantsch

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 29, No. 12, pp 1973-1986, Dec. 2010.

Buffer Optimization in Network-on-Chip Through Flow Regulation

Fahimeh Jafari, Zhonghai Lu, *Member, IEEE,* Axel Jantsch, *Member, IEEE,* and Mohammad Hossein Yaghmaee, *Member, IEEE*

***Abstract*—For network-on-chip (NoC) designs, optimizing buffers is an essential task since buffers are a major source of cost and power consumption. This paper proposes flow regulation and has defined a regulation spectrum as a means for system-on-chip architects to control delay and backlog bounds. The regulation is performed per flow for its peak rate and burstiness. However, many flows may have conflicting regulation requirements due to interferences with each other. Based on the regulation spectrum, this paper optimizes the regulation parameters aiming for buffer optimization. Three timing-constrained buffer optimization problems are formulated, namely, buffer size minimization, buffer variance minimization, and multiobjective optimization, which has both buffer size and variance as minimization objectives. Minimizing buffer variance is also important because it affects the modularity of routers and network interfaces. A realistic case study exhibits 62.8% reduction of total buffers, 84.3% reduction of total latency, and 94.4% reduction on the sum of variances of buffers. Likewise, the experimental results demonstrate similar improvements in the case of synthetic traffic patterns. The optimization algorithm has low run-time complexity, enabling quick exploration of large design spaces. This paper concludes that optimal flow regulation can be a highly valuable instrument for buffer optimization in NoC designs.**

***Index Terms*—Buffer size, buffer variance, interior point method, network-on-chip (NoC), optimization problem.**

I. Introduction

THE advance of the technology is raising the level of integration of intellectual property (IP) and scalability issue for communication architectures in very large-scale integration systems. Since traditional buses do not scale well in the system-on-chip (SoC) platforms, this trend has driven bus-based architecture toward networks-on-chip (NoCs) [1]. Current achievements in integrating more processor cores on a single chip enable to employ these many-core systems as real time multimedia servers. Thus, it is imperative to provide quality of service (QoS) in these systems which have been well available in traditional Internet servers. IPs for a SoC are typically developed concurrently using a standard interface, e.g., advanced extensible interface or open core protocol. Despite the standard interfaces, integrating IPs into a SoC infrastructure presents challenges because: 1) traffic flows from IPs are diverse and typically have stringent performance constraints; 2) the impact of interferences among traffic flows is hard to analyze; and 3) of the cost and power constraint, buffers in the SoC infrastructure must not be over-dimensioned while still satisfying performance requirements even under worst-case conditions.

Fig. 1 illustrates the approach that we have proposed and investigated in [2] for addressing the IP integration problem. Master IPs send read and write requests to slave IPs which respond with read data and write acknowledgments. The admission of traffic flows from master IPs into the SoC infrastructure can be controlled by a regulator rather than injecting them as soon as possible. Thus, we can control QoS and achieve cost-effective communication. To lay a solid foundation of the approach, our flow regulation has been based on network calculus [3]–[6]. By importing and extending the analytical methods from network calculus, we can obtain worst-case delay and backlog bounds. In [7], we implemented the microarchitecture of the regulator and quantified its hardware speed and cost. The aim of this paper is to optimize the regulator parameters including peak rate and traffic burstiness of flows by formulating optimization problems.

Silicon area and power consumption are two critical design challenges for NoC architectures. The network buffers take up a significant part of the NoC area and power consumption [8]; consequently, the size of buffers in the system should be minimized. On the contrary, buffers should be large enough to improve communication performance. This means that there is a tradeoff between buffer size and performance metrics. Hence, we address an optimization problem of minimizing the total number of buffers subject to the performance constraints of the applications running on the SoC. Moreover, since reusing similar or identical switches facilitates the design process of NoC-based systems, we formulate another optimization problem to minimize the variances of buffer size in the respective output buffers of switches. As both of the mentioned objective functions are worthwhile for the design process, we formulate them as a multiobjective problem under QoS constraints. Finally, we show the benefits of the proposed method and quantify performance improvement and buffer size and variance reduction.

The remainder of this paper is organized as follows. Section II gives an account of related works. In Section III, we

Manuscript received December 1, 2009; revised May 12, 2010; accepted July 3, 2010. Date of current version November 19, 2010. This paper was recommended by Associate Editor V. Narayanan.

F. Jafari, Z. Lu, and A. Jantsch are with the Department of Electronic Systems, Royal Institute of Technology, SE-164 40 Kista, Stockholm, Sweden (e-mail: fjafari@kth.se; zhonghai@kth.se; axel@kth.se).

M. H. Yaghmaee is with the Computer Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad 91775-111, Iran (e-mail: hyaghmae@ferdowsi.um.ac.ir).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2063130

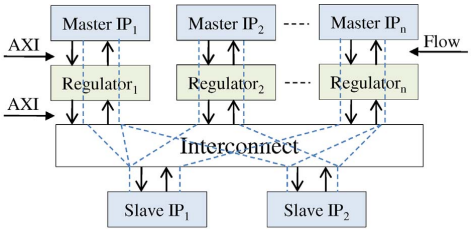


Fig. 1. IP integration in SoCs.

introduce the flow regulation concept along with the basics of *Network Calculus* [3]–[6]. Section IV discusses the underlying system model. Section V is devoted to the discussion about the buffer optimization problems. In Section VI, we present the solution method using an iterative approach. Our simulation results are described in Section VII. We discuss the scope and assumptions in Section VIII. Finally, Section IX gives the conclusion and future work.

II. Related Work

A. Network Calculus

Cruz [4] and Chang [6] have pioneered the network calculus [4], which is a mathematical framework to derive worst case bounds on maximum latency, backlog, and minimum throughput. In [5], a general latency-rate server model was proposed for analyzing traffic scheduling algorithms. Based on this model, they derived deterministic delay and backlog bounds. Le Boudec and Thiran [3] summarized the results of network calculus and their applications in Internet and ATM. Real-time calculus [9], close to network calculus, was developed for platform-based embedded systems. It generalizes standard event models via upper and lower arrival curves, and processing-element models via upper and lower service curves. Based on these curves, it derives delay and backlog bounds. The authors in [2] proposed a network calculus-based flow regulation and defined a regulation spectrum as a design instrument for SoC architects to control QoS. In this paper, we use the concept of regulation and regulation spectrum in [2] and address the issue of optimal regulation for buffer optimization. We optimize the regulator parameters including peak rate and traffic burstiness of flows by formulating optimization problems.

B. Application Specific Design

NoC-based SoC architectures are often designed for a specific application or a class of applications. Thus, designers customize it for a specific application to achieve best performance and cost trade-offs. The authors in [10] and [11] show the advantages of the topological mapping of IPs on the NoC architectures. In [12], the network topology customization and its effects on the system are considered. In [13] and [14], the authors investigate the customized allocation of buffer resources to different channels of routers. Actually, these works strived to distribute a given budget of buffering space among channels. Also, they are based on the average-case analysis which is not sufficient for a system with hard real-time requirements.

In [15], we followed a different direction by addressing an optimization problem to find the minimum total buffering requirements while satisfying acceptable communication performance in NoCs with round robin arbitration. In this paper, we have significantly extended the work in [15]. We address not only the buffer size minimization problem but also the buffer variance minimization problem. Moreover, since both objectives are desirable for NoC designs, we formulate a multiobjective optimization problem to minimize both buffer size and buffer variance. We give a systematic account of all the three problems, i.e., the buffer size minimization, the buffer variance minimization, and the multiobjective optimization. Furthermore, we construct the model for weighted round robin arbitration which outperforms round robin policy. It is worth mentioning that our method is presented based on tight worst-case bounds derived by network calculus. Therefore, it is suitable for real-time system designs.

C. Optimization Method

In this paper, we formulate optimization problems to optimize the regulator parameters with respect to buffer requirements.

In the literature, the proposed constrained problems are called nonconvex nonlinear programming (NLP) problems [16]. The general aim in constrained optimization is to transform the problem into an easier subproblem that can then be solved and used as the basis of an iterative process [16]. A characteristic of a large class of early methods is the translation of the constrained problem to a basic unconstrained problem by using a penalty function for constraints that are near or beyond the constraint boundary. In this way, the constrained problem is solved using a sequence of parameterized unconstrained optimizations, which in the limit converge to the constrained problem. These methods are now considered relatively inefficient and have been replaced by methods that have focused on the solution of the Karush-Kuhn-Tucker (KKT) equations [16], [17]. The KKT equations are necessary conditions for optimality for a constrained optimization problem.

The solution of the KKT equations forms the basis to many nonlinear programming algorithms. These algorithms attempt to compute the Lagrange multipliers directly. In particular, we will solve the proposed optimization problems using interior point method for constrained NLP problems [16], [17].

III. Concepts of Flow Regulation

A. Network Calculus Basics

A flow f is an infinite stream of unicast traffic (packets) sent from a source node and flow j is denoted as f_j. In network calculus [3], a flow $f_j(t)$ represents the accumulated number of bits transferred in the time interval $[0, t]$. To obtain the average and peak characteristics of a flow, traffic specification (TSPEC) is used. With TSPEC, f_j is characterized by an *arrival curve* $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate ($p_j \geq \rho_j$), σ_j the burstiness

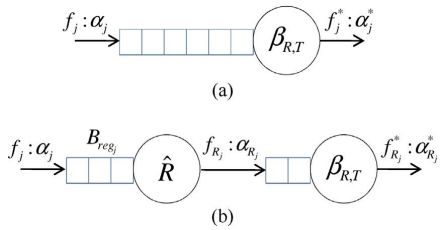


Fig. 2. Flow served by a latency-rate server. (a) Flow served without regulation. (b) Flow served after regulation.

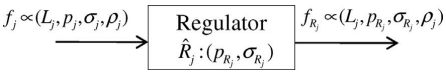


Fig. 3. Flow regulation.

$(\sigma_j \geq L_j)$, and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. The burstiness also is an important case among these parameters because a flow with low average rate and unlimited burst size can incur an unlimited delay on its own packets.

Network calculus uses the abstraction of service curve to model a network element (node) processing traffic flows. A service curve reflects the processing latency and service capability of the node. A well-formulated service model is the latency-rate function $\beta_{R,T} = R(t - T)^+$, where R is the minimum service rate and T is the maximum processing latency of the node [5]. Notation $x^+ = x$ if $x > 0; x^+ = 0$, otherwise.

As depicted in Fig. 2(a), a TSPEC flow $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$ (denoted as $f_j : \alpha_j$) is served by a node guaranteeing a latency-rate service $\beta_{R,T}$. According to [3], the maximum delay and the buffer required for flow j are bounded by (1) and (2), respectively

$$\bar{D}_j = \frac{L_j + \theta_j (p_j - R)^+}{R} + T \tag{1}$$

$$\bar{B}_j = \sigma_j + \rho_j T + (\theta_j - T)^+ [(p_j - R)^+ - p_j + \rho_j] \tag{2}$$

where $\theta_j = (\sigma_j - L_j)/(p_j - \rho_j)$. The output flow f_j^* is bounded by another affine arrival curve $\alpha_j^*(t) = (\sigma_j + \rho_j T) + \rho_j t,\ \theta_j \leq T$; $\alpha_j^*(t) = min((T + t)(min(p_j, R)) + L_j + \theta_j (p_j - R)^+, (\sigma_j + \rho_j T) + \rho_j t),\ \theta_j > T$.

B. Regulation Spectrum

TSPEC can be used to characterize flows. It can also be used to define a traffic regulator. Fig. 3 shows that an input flow f_j reshaped by a regulation component $\hat{R}_j(p_{R_j}, \sigma_{R_j})$ results in an output flow f_{R_j}. We assume the regulator has the same input and output data unit, *flit*, and the same input and output capacity *C flits/cycle*. We also assume that f_j's average bandwidth requirement must be preserved. The output flow f_{R_j} is characterized by the four parameters $(L_j, p_{R_j}, \sigma_{R_j}, \rho_j)$, where $p_{R_j} \in [\rho_j, p_j]$, $\sigma_{R_j} \in [L_j, \sigma_j]$. f_j can be losslessly reshaped by the regulator, meaning that f_{R_j} has the same L and average rate ρ as f_j. The two intervals $p_{R_j} \in [\rho_j, p_j]$ and

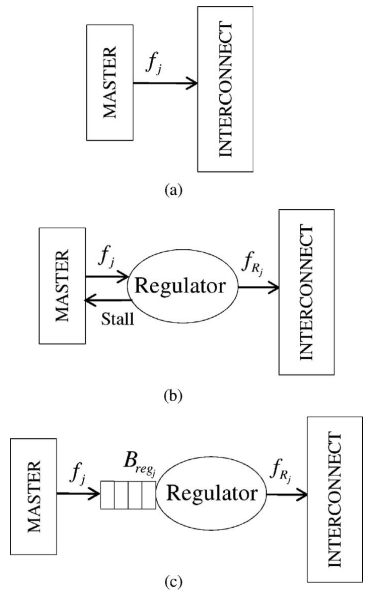


Fig. 4. Mechanisms of flow regulation. (a) Self-regulating master. (b) IPs are stalled: no queuing buffer. (c) IPs are not stalled: queuing buffer.

$\sigma_{R_j} \in [L_j, \sigma_j]$ are called the *regulation spectrum*, where the former is for the regulation of peak rate and the latter for the regulation of traffic burstiness.

The regulation spectrum defines the upper and lower limits of regulation. Fig. 2(b) shows how the flow is served after regulation. We implemented microarchitecture of the regulator and quantified its hardware speed and cost in [7]. Selecting appropriate p_{R_j} and σ_{R_j} is very effective in performance and cost of communications. In the next sections, we formulate three optimization problems that consider these regulation parameters as decision variables.

C. Mechanism and Cost of Flow Regulation

There are three different ways to realize the flow regulation, each of which incurs different costs.

1) *Regulation by design methodology:* as shown in Fig. 4(a), no regulator is implemented in the system. The IP or the application is designed such that it meets the regulation requirements. If that can be guaranteed, there is no additional cost in the network or the network interface. Also, there is no buffers and no delay due to regulation; consequently, there is no hardware cost for designing the regulator. However, the design structure of master should be changed to have a self-regulating master. This means that the workload is pushed to the master and application design. Thus, it applies to new IPs, but not applicable to legacy IPs.
2) *Regulation by a hardware regulator:* a hardware regulator is implemented which enforces traffic regulation at the network interfaces. There are two ways that the hardware regulator may affect the behavior of IPs as

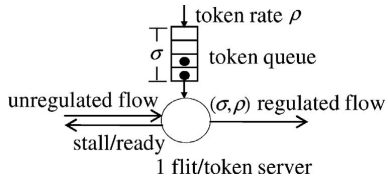


Fig. 5. (σ, ρ)-based regulation mechanism.

follows.

a) As shown in Fig. 4(b), the regulator does not buffer the packets, but stalls the traffic producers or IPs. In this case, no buffer due to regulation is required, but the behavior of masters should also be modified. This may be a good idea if the traffic producer is a multitasking CPU that can do something else while waiting. In this case the traffic generation is simply delayed and no buffering costs occur in the system.

b) The traffic producers or IPs are not stalled but the regulators use buffers to store transactions as depicted in Fig. 4(c). This can reduce back-pressure at the expense of buffering cost. Thus, this scheme allows any legacy IPs to be directly used in the system.

In principle, which option is best will depend on the context (application, IPs, architecture, and so on). The significant benefit of case 2b in comparison with others is simplicity of design process because no changes are required for the master structure. In this paper, we have implemented our proposed method based on case 2b concepts, but it can be easily extended for other cases.

To evaluate the overhead in silicon area due to the use of regulators, we designed and synthesized a multi-flow regulator with Synopsys tools using 180 nm technology [7]. When optimized for area, the multi-flow regulator using three regulators consumes $5K$ gates, running up to 730 MHz. Buffers and packet latency due to regulation depend on the values of the regulation parameters including peak rate and traffic burstiness which will be calculated in our case study in Section VII. The regulation mechanism in this paper is described as follows.

The regulator is implemented using the token-bucket mechanism [18] as shown in Fig. 5. The token queue has a size of σ. Initially the token queue is full. The 1-flit/token server admits one flit by de-asserting the "stall" signal as long as the token queue is not empty. The token queue is realized by a saturating credit counter that increments at rate ρ and saturates when it reaches a count of σ. A flit can be transmitted if and only if the credit counter is positive (at least one token available). Each time a flit is sent, the counter is decremented by 1.

IV. System Model and Delay/Backlog Bounds

We aim at optimizing buffer requirements while satisfying acceptable latency in on chip communications. We shall formulate optimization problems based on an analytical performance model. At first, we shall derive the per-flow worst-case delay and backlog bounds.

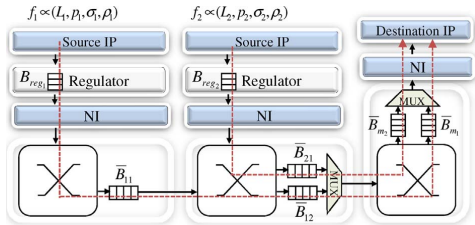


Fig. 6. Example of required buffers for two flows.

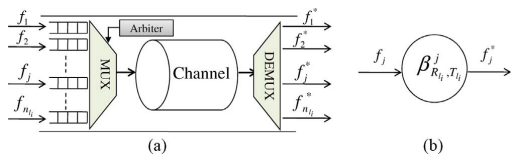


Fig. 7. (a) Channel sharing among set of flows. (b) Channel service model for flow j.

A. Assumptions and Notations

We consider a NoC architecture which can have different topologies. Every node contains an IP core and a router with $p+1$ input channels and $q+1$ output channels. Each IP core performs its own computational, storage or input/output processing functionality, and is equipped with a network interface (NI). NIs provide an interface between IPs and the network and they are responsible for packetization/depacketization of messages. Note that the presence of NIs is the consequence of using a network rather than using regulators. Regulators are inserted between the source IP and the NI. We presume the number of virtual channels for each physical channel is the same as the number of flows passing through that channel. Fig. 6 shows required buffers of flows f_1 and f_2 from different sources to the same destination. The following analysis on buffer requirements of flows is illustrated by this figure. We also assume that the NoC architecture is lossless, and packets traverse the network in a best-effort fashion using a deterministic routing. This means that the path of a flow is statically determined.

To facilitate our discussions, we turn the aforementioned NoC architecture into a mathematically modeled network. In this respect, we consider a NoC as a network with a set of bidirectional channels L, and a set of flows F. Each physical channel $i \in L$ has a fixed capacity of c_{l_i} *flits/cycle*. We denote the set of flows that share channel i by F_{l_i} and their number is denominated as n_{l_i}. Similarly, the set of channels that flow j passes through, is denoted by L_{f_j} and their number is denominated as n_{f_j}. By definition, $j \in F_{l_i}$ if and only if $i \in L_{f_j}$.

B. Channel Service Model

To compute the flow traversal delay and backlog bounds using the equations, we first need to build a channel service model. The network channel and the ejection channel at the destination node are treated in the same way since both types of channels are multiplexed by multiple flows with an arbitration policy.

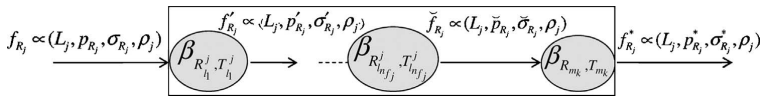


Fig. 8. Modeling each network element as a latency-rate server.

Fig. 7(a) depicts a channel l_i allocated to n_{l_i} flows. Since the arbitration policy determines how much the flows influence each other, it has to be known. We assume that, while serving multiple flows, the routers employ weighted round robin scheduling to share the link bandwidth. Assuming a fixed word length of L_w in all of flows, weighted round robin arbitration means that each flow j gets at least a $\frac{\rho_j}{\sum_{\forall f_k \in F_{l_i}} \rho_k} c_{l_i}$ of the channel bandwidth. A flow may get more if the other flow uses less, but we now know a worst-case lower bound on the bandwidth. Since network calculus uses the abstraction of service curve to model a network element processing traffic flows [3], we can also model a weighted round robin arbiter of channel l_i for flow j as a latency-rate server [19] that its function is as $\beta_{R^j_{l_i},T^j_{l_i}} = R^j_{l_i}(t - T^j_{l_i})^+$, where $R^j_{l_i}$ is the minimum service rate and $T^j_{l_i}$ is the maximum processing latency of the arbiter of channel l_i for flow j. $R^j_{l_i}$ and $T^j_{l_i}$ are defined as follows:

$$R^j_{l_i} = \frac{\rho_j}{\sum_{f_k \in F_{l_i}} \rho_k} c_{l_i} \tag{3}$$

$$T^j_{l_i} = \frac{(\sum_{f_k \in F_{l_i}} N^k_{l_i} - N^j_{l_i}) L_w}{c_{l_i}} \tag{4}$$

where $N^j_{l_i}$ is the minimum positive integer for flow j passing through channel i provided that

$$\frac{\rho_j}{\sum_{\forall f_k \in F_{l_i}} \rho_k} = \frac{N^j_{l_i}}{\sum_{\forall f_k \in F_{l_i}} N^k_{l_i}} \qquad \forall f_j \in F_{l_i}.$$

For (3), $R^j_{l_i}$ denotes the minimum weight-proportional bandwidth that flow j can take from channel i. For (4), $T^j_{l_i}$ denotes the maximum blocking time for flow j when passing through channel i. The channel service model for flow j is shown in Fig. 7(b).

With the channel service model, we can now model a flow passing through a series of channels including the ejection channel as a series of concatenated latency-rate servers. Fig. 8 shows a traffic flow f_j after regulation which is called f_{R_j} and is passing through adjacent channels. We construct an analytical model with the network elements depicted in this figure. Every channel $l_i \in L_{f_j}$ that flow j passing through can be modeled as a latency-rate server for flow j with service curve $\beta_{R^j_{l_i},T^j_{l_i}}$, and also the ejection channel in the destination node of flow j, node k, can be modeled as a latency-rate server with service curve $\beta_{R_{m_k},T_{m_k}}$.

C. Tight Worst-Case Bounds for Each Flow

Consider that flow j passes through the regulator and several network channels offering each a latency-rate service curve. For each flow, the delay and backlog bounds have two components: one incurred at the regulator and the other the network.

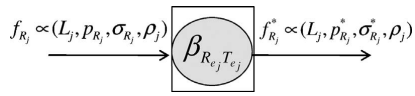


Fig. 9. Modeling all network elements as a latency-rate server.

1) *Delay and Backlog Bounds at Regulators:* To determine the delay and backlog due to the regulation, its impact on the behavior of IPs should be considered. As discussed in Section III-C, one is that IPs are *stalled* and therefore, there is no queuing buffer at the regulator. In the other case which is adopted in this paper, IPs are *not stalled* and the regulators use buffers to store transactions. This can decrease back-pressure at the expense of buffering cost. Let D_{reg_j} and B_{reg_j} be the delay and backlog for flow j due to regulation, respectively. We have $B_{reg_j} = \Delta\sigma_j = \sigma_j - \sigma_{R_j}$, which is the difference between the input and output burstiness of the regulator, and $D_{reg_j} = \Delta\sigma_j/\rho_j$ [2].

2) *Delay and Backlog Bounds in the Network:*

a) *Delay bound:* To compute the delay bound for a flow passing a series of nodes, one simple way is to calculate the summation of delay bounds at each node. However, this results in a loose total delay bound. To tighten the worst-case delay bound along the network, we use the theorem of *concatenation of network elements* [3]. Given are two nodes sequentially connected and each is offering a latency-rate service curve β_{R_i,T_i}, $i = 1$ and 2. These nodes can be represented as a single latency-rate server as follows:

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2}. \tag{5}$$

As depicted in Fig. 9, we can model all network elements on a given flow as a single latency-rate server $\beta_{R_{e_j},T_{e_j}}$ with the following characteristics:

$$R_{e_j} = min\left(min_{l_i \in L_{f_j}}\left(\frac{\rho_j}{\sum_{f_k \in F_{l_i}} \rho_k} c_{l_i}\right), \frac{\rho_j}{\sum_{f_r \in F_{d_k}} \rho_r} c_{m_k}\right) \tag{6}$$

$$T_{e_j} = \sum_{l_i \in L_{f_j}}\left(\frac{(\sum_{f_k \in F_{l_i}} N^k_{l_i} - N^j_{l_i}) L_w}{c_{l_i}}\right) + \frac{(\sum_{f_r \in F_{d_k}} N^r_{d_k} - N^j_{d_k}) L_w}{c_{m_k}} \tag{7}$$

where R_{e_j} denotes the minimum service rate among channels through which flow j passes and T_{e_j} the sum of maximum processing latency of the mentioned channels.

Based on a corollary of this theorem which is known as *Pay Bursts Only Once* [3], the equivalent latency-rate server is used for obtaining worst-case delay bound. Therefore, according to

(1), (6), and (7), the maximum delay for flow j in network is bounded by

$$\bar{D}_j = \frac{L_j + \theta_{R_j}(p_{R_j} - R_{e_j})^+}{R_{e_j}} + T_{e_j} + n_{f_j} d_p \tag{8}$$

where d_p is delay for propagation in a channel which is assumed identical for all channels. Therefore, $n_{f_j} d_p$ is propagation delay in whole network for flow j and $\theta_{R_j} = \frac{\sigma_{R_j} - L_j}{p_{R_j} - \rho_j}$. Hence, the total maximum delay for the flow j is bounded as $D_{reg_j} + \bar{D}_j$.

b) *Backlog bound:* For calculating tight worst-case bound on backlog along the network, the sum of the individual bounds on every element is computed [3]. Thus, the required buffer in network for flow j is bounded by

$$\bar{B}_j = \sum_{i \in L_{f_j}} \bar{B}_{ji} + \bar{B}_{m_j} \tag{9}$$

where $\bar{B}_{ji}$ is the upper bound on the buffer for flow j for each $i \in L_{f_j}$ and $\bar{B}_{m_j}$ is the maximum required buffer for the ejection channel multiplexer of the destination node of flow j. $\bar{B}_{ji}$ and $\bar{B}_{m_j}$ can be easily obtained by (2). For example, directly applying (2) for flow j in Fig. 8, $\bar{B}_{m_k}$ can be calculated by

$$\bar{B}_{m_k} = \breve{\sigma}_{R_j} + \rho_j T^j_{m_k} + (\breve{\theta}_j - T^j_{m_k})^+ [(\breve{p}_{R_j} - R^j_{m_k})^+ - \breve{p}_{R_j} + \rho_j]. \tag{10}$$

Finally, the total buffer requirements for flow j are bounded by $B_{reg_j} + \bar{B}_j$.

V. Buffer Optimization Problems

A. Buffer Size Optimization

As stated before, our objective is to choose output peak rate and traffic burstiness of regulators for each flow so as to minimize the buffer requirements while satisfying acceptable performance in the network. Thus, the buffer size minimization problem, *Minimize-Size*, can be formulated as follows.

Given a set of flows $F = \{f_j \propto (L_j, p_j, \sigma_j, \rho_j)\}$, routing matrix R, the maximum delay that each flow can suffer in the network $d = \{d_j\}$ for $\forall f_j \in F$, *find* the regulator parameters, peak rate p_{R_j} and traffic burstiness σ_{R_j} for $\forall f_j \in F$, such that

$$\min_{p_{R_j}, \sigma_{R_j}} \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) \tag{11}$$

subject to

$$D_{reg_j} + \bar{D}_j \le d_j \qquad \forall f_j \in F \tag{12}$$
$$\rho_j \le p_{R_j} \le p_j \qquad \forall f_j \in F \tag{13}$$
$$L_j \le \sigma_{R_j} \le \sigma_j \qquad \forall f_j \in F \tag{14}$$
$$\bar{B}_j > 0 \qquad \forall f_j \in F \tag{15}$$

where p_{R_j} and σ_{R_j} for $\forall f_j \in F$ are optimization variables.

Equation (11) is the objective function of this optimization problem which minimizes total buffer requirements. Constraint (12) says that the maximum delay of each flow j cannot exceed the maximum delay that it can suffer in the network d_j. Since we measured the flow performance in terms of its latency, we can consider d_j as a criterion of minimum guaranteed performance for flow j. Constraints (13) and (14) are related to two intervals $p_{R_j} \in [\rho_j, p_j]$ and $\sigma_{R_j} \in [L_j, \sigma_j]$ which called the regulation spectrum as described in Section III-B.

It is clear that by following the above mentioned equations, we can understand the effect of optimization variables on the objective function and all constraints of the defined problem.

In the literature, (11) is called a nonconvex NLP problem [16]. There are different methods for solving this kind of optimization problems. In particular, we will solve the optimization problem (11) using interior point method for constrained NLP problems [16], [17].

B. Buffer Variance Optimization

To reuse IP modules, designers would like to use similar switches as far as possible. However, flow requirements differ from each other in terms of buffer size; consequently, we would like to find appropriate peak rate and traffic burstiness of each flow so that variances of buffer size in the respective output buffers of switches are minimized. For example in a 2-D mesh network, we would like to minimize the variance of buffer size in northern output port of switches, as well as other output ports. Using general variance formula, we can easily calculate variances of the required buffer on each output port i which is denoted by var_i. Hence, we formulate another optimization problem to minimize the sum of required buffers variances while satisfying QoS requirements in the network. Thus, the buffer variance minimization problem, *Minimize-Variance*, can be formulated as follows.

Given a set of flows $F = \{f_j \propto (L_j, p_j, \sigma_j, \rho_j)\}$, routing matrix R, the maximum delay that each flow can suffer in the network $d = \{d_j\}$ for $\forall f_j \in F$, *find* the regulator parameters, peak rate p_{R_j} and traffic burstiness σ_{R_j} for $\forall f_j \in F$, such that

$$\min_{p_{R_j}, \sigma_{R_j}} \sum_i var_i \tag{16}$$

subject to

$$D_{reg_j} + \bar{D}_j \le d_j \qquad \forall f_j \in F \tag{17}$$
$$\rho_j \le p_{R_j} \le p_j \qquad \forall f_j \in F \tag{18}$$
$$L_j \le \sigma_{R_j} \le \sigma_j \qquad \forall f_j \in F \tag{19}$$
$$\bar{B}_j > 0 \qquad \forall f_j \in F. \tag{20}$$

Optimization variables are p_{R_j} and σ_{R_j}, $\forall f_j \in F$, that can be detected in the objective function and constraints by the following equations. Similar to (11), (16) also is a nonconvex NLP that can be solved via the interior point method.

C. Multiobjective Optimization Problem

As both of the aforementioned objective functions are worthwhile for designing the network, we formulate a multiobjective optimization problem which minimizes both total buffers and variances, *Multiobjective*, as follows.

Given a set of flows $F = \{f_j \propto (L_j, p_j, \sigma_j, \rho_j)\}$, routing matrix R, the maximum delay that each flow can suffer in the

network $d = \{d_j\}$ for $\forall f_j \in F$, *find* the regulator parameters, peak rate p_{R_j} and traffic burstiness σ_{R_j} for $\forall f_j \in F$, such that

$$\min_{p_{R_j},\sigma_{R_j}} f_1 = \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) \tag{21}$$

$$\min_{p_{R_j},\sigma_{R_j}} f_2 = \sum_i var_i \tag{22}$$

subject to

$$D_{reg_j} + \bar{D}_j \le d_j \qquad \forall f_j \in F \tag{23}$$

$$\rho_j \le p_{R_j} \le p_j \qquad \forall f_j \in F \tag{24}$$

$$L_j \le \sigma_{R_j} \le \sigma_j \qquad \forall f_j \in F \tag{25}$$

$$\bar{B}_j > 0 \qquad \forall f_j \in F. \tag{26}$$

In multiobjective optimizations, there is not even a universally accepted definition of *optimum* as in single-objective optimization, which makes it difficult to even compare results of one method to another, because normally the decision about what the *best* answer corresponds to the so-called *decision maker* [23]. Overall, there are different ways for solving multiobjective optimizations. One of them is combining objectives into a single function which normally denominated *Weighted Sum Approach*. Since objective functions in this paper are in the same direction and they are not in conflict with each other, we adopt this approach. The results in Section VII also confirm that the obtained solution of the proposed multiobjective problem is very close to optimal points of *Minimize-Size* and *Minimize-Variance* problems. This means that it is an appropriate method for solving this problem. The main advantage of this approach is the simplicity of its implementation and its computational efficiency. This method consists of adding all the objective functions together using weighting coefficients for each one of them. Specifically, our multiobjective problem is transformed into a scalar optimization problem of the form

$$min(w_1 f_1 + w_2 f_2) \tag{27}$$

where w_1 and w_2 are the weighting coefficients representing the relative importance of the objectives. In this paper, they are assumed the same. This approach has a low run-time complexity because of its simplicity and efficiency and therefore, can be applied for complex SoC designs. We solve the mentioned problem still using the interior point method.

VI. Optimization Method

A. Optimization Algorithm

As stated before, the proposed optimization problems are called nonconvex NLP problems [16] and solved by the interior point method. There are different packages for solving this kind of optimization problems and we particularly use the MATLAB optimization package in this paper.

To exemplify the optimization approach, we will solve the buffer size optimization problem (11), using the interior point method for constrained NLP problems [16], [17].

The interior point approach to constrained minimization is to solve a sequence of approximate minimization problems called *barrier* problem [17]. Due to (11), for each $\mu > 0$, the barrier problem is

$$\min_{p_{R_j},\sigma_{R_j},s_i} \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) - \mu \sum_{i=1}^{6|F|} ln(s_i) \tag{28}$$

subject to

$$D_{reg_j} + \bar{D}_j - d_j + s_i = 0 \quad \forall f_j \in F \quad i = 1, ..., |F| \tag{29}$$

$$\rho_j - p_{R_j} + s_i = 0 \quad \forall f_j \in F \quad i = |F| + 1, ..., 2|F| \tag{30}$$

$$p_{R_j} - p_j + s_i = 0 \quad \forall f_j \in F \quad i = 2|F| + 1, ..., 3|F| \tag{31}$$

$$L_j - \sigma_{R_j} + s_i = 0 \quad \forall f_j \in F \quad i = 3|F| + 1, ..., 4|F| \tag{32}$$

$$\sigma_{R_j} - \sigma_j + s_i = 0 \quad \forall f_j \in F \quad i = 4|F| + 1, ..., 5|F| \tag{33}$$

$$s_i - \bar{B}_j = 0 \quad \forall f_j \in F \quad i = 5|F| + 1, ..., 6|F| \tag{34}$$

where $|F|$ is the cardinality of set F.

There are as many slack variables s_i as inequality constraints (12)–(15). The s_i are restricted to be positive to keep $ln(s_i)$ bounded. As μ decreases to zero, the minimum of f_μ should approach the minimum of f. The approximate problem (28) is a sequence of equality constrained problems. These are easier to solve than the original inequality-constrained problem (11).

To facilitate our discussion, we define $p_R = (p_{R_1}, ..., p_{R_{|F|}})^T$, $\sigma_R = (\sigma_{R_1}, ..., \sigma_{R_{|F|}})^T$, $s = (s_1, ..., s_{6|F|})^T$ and assume $g(p_R, \sigma_R) = (g_1(p_R, \sigma_R), ..., g_{6|F|}(p_R, \sigma_R))^T$ so that $g(p_R, \sigma_R)+ s$ is a vector that its elements are constraints (29)–(34). Thus, the barrier problem (28) can be rewritten as

$$\min_{p_R,\sigma_R,s} f_\mu(p_R, \sigma_R, s) = \min_{p_R,\sigma_R,s} f(p_R, \sigma_R) - \mu \sum_{i=1}^{6|F|} ln(s_i) \tag{35}$$

subject to

$$g(p_R, \sigma_R) + s = 0. \tag{36}$$

In the following, we shall find an approximate solution to (35), for fixed μ. Then, the used method is applied repeatedly to (35), for decreasing values of μ, to approximate the solution of the original problem (11).

Using the optimization methods [16], the Lagrangian of the problem (35) can be written as

$$L(p_R, \sigma_R, s, \lambda) = f(p_R, \sigma_R) - \mu \sum_{i=1}^{6|F|} ln(s_i) + \lambda^T (g(p_R, \sigma_R) + s) \tag{37}$$

where $\lambda = (\lambda_1, ..., \lambda_{6|F|})^T$ is the vector of Lagrange multipliers. Regarding the first-order optimality conditions, at an optimal solution (p_R, σ_R, s) of the barrier problem, we have

$$\nabla_{p_R} L(p_R, \sigma_R, s, \lambda) = \nabla_{p_R} f(p_R, \sigma_R) + A(p_R, \sigma_R)\lambda = 0 \tag{38}$$

$$\nabla_{\sigma_R} L(p_R, \sigma_R, s, \lambda) = \nabla_{\sigma_R} f(p_R, \sigma_R) + \acute{A}(p_R, \sigma_R)\lambda = 0 \tag{39}$$

$$\nabla_s L(p_R, \sigma_R, s, \lambda) = -\mu S^{-1} e + \lambda = 0 \tag{40}$$

where $A(p_R, \sigma_R) = (\nabla_{p_R} g_1(p_R, \sigma_R), ..., \nabla_{p_R} g_{6|F|}(p_R, \sigma_R))$ and $\acute{A}(p_R, \sigma_R) = (\nabla_{\sigma_R} g_1(p_R, \sigma_R), ..., \nabla_{\sigma_R} g_{6|F|}(p_R, \sigma_R))$ are the

matrixes of constraint gradients with respect to p_R and σ_R, respectively, and where

$$e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, S = \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_{6|F|} \end{pmatrix}.$$

To solve the approximate problem, we should generate a step d for displacement at an iterate z, where

$$d = \begin{pmatrix} d_{p_R} \\ d_{\sigma_R} \\ d_s \end{pmatrix}.$$

One of the two main types of steps is used at each iteration.

1) A *direct* step in (p_R, σ_R, s). This step attempts to solve the KKT equations for the approximate problem via a linear approximation. This is also called a Newton step [20].
2) A *conjugate gradient (CG)* step, using a trust region [21].

The algorithm first attempts to take a direct step. If it cannot, it attempts a CG step. One case where it does not take a direct step is when the approximate problem is not locally convex near the current iteration.

Afterward, it is necessary to decide if the step obtained from the abovementioned methods is acceptable. For this purpose, a merit function is introduced. The merit function is given by

$$\phi = \sum_{\forall f_j \in F} (B_{reg_j} + \bar{B}_j) - \mu \sum_{i=1}^{6|F|} ln(s_i) + \nu \, \| g(p_R, \sigma_R) + s \| \quad (41)$$

where $\nu > 0$ is a *penalty parameter* and can increase with iteration number in order to force the solution toward feasibility.

The step is accepted if it gives sufficient reduction in the merit function; otherwise it is rejected. More details of the *direct* and *CG* steps are described in the following.

According to the above discussions, we present an iterative algorithm as the solution to (11). Algorithmic realization of the solution method is listed as Algorithm 1. In this respect, optimal peak rate and traffic burstiness for traffic flows can be found while minimizing total buffer requirements under performance constraints.

Algorithm 1: Buffer Size Minimization Algorithm

Initialization:

1. Choose a *penalty parameter* $\nu > 0$ and a *barrier parameter* $\mu > 0$.
2. Initialize trust region radius $R > 0$ and Lagrange multipliers λ.
3. Set an appropriate initial value for peak rate and burstiness of flows for problem (11) denoted as $p_R(0)$, $\sigma_R(0)$.
4. Specify an appropriate value for ϵ, $\acute{\epsilon}$ ($\acute{\epsilon}$ denote value of expectable reduction in merit function).

1. *Loop* 1: Do until $(\max \mid p_R(t+1) - p_R(t) \mid < \epsilon) \& (\max \mid \sigma_R(t+1) - \sigma_R(t) \mid < \epsilon)$

2. Set an appropriate initial value for peak rate and burstiness of flows and slack variables for barrier problem (35) denoted as $\acute{p}_R(0)$, $\acute{\sigma}_R(0)$, $\acute{s}(0)$.

3. *Loop* 2: Do until $(\max \mid \acute{p}_R(k+1) - \acute{p}_R(k) \mid < \epsilon) \& (\max \mid \acute{\sigma}_R(k+1) - \acute{\sigma}_R(k) \mid < \epsilon)$

4. if H is not definite positive go to 5
   4.1. Calculate d based on *Direct Step* as described in Section VI-B
   4.2. Go to 6.

5. Calculate d based on *CG Step* as described in Section VI-C

6. $p_{temp} = \acute{p}_R(k) + d_{p_R}$;
7. $\sigma_{temp} = \acute{\sigma}_R(k) + d_{\sigma_R}$;
8. $s_{temp} = \acute{s}(k) + d_s$

9. Calculate $\phi(k+1)$ by substituting p_{temp}, σ_{temp}, s_{temp} in merit problem (41).

10. if $(\phi(k+1) - \phi(k) \geq \acute{\epsilon})$
    10.1. Decrease R;
    10.2. Go to 4;

11. $p_R(k+1) = p_{temp}$; $\sigma_R(k+1) = \sigma_{temp}$; $s(k+1) = s_{temp}$

12. Compute new Lagrange multipliers λ.
13. End of loop 2.

14. Decrease *barrier parameter* μ.
15. End of loop 1.

Output:
Communicate optimal peak rates and traffic burstinesses to the corresponding regulators.

B. Direct Step

This step attempts to solve the KKT equations for the barrier problem via a linear approximation. Regarding the KKT conditions for the equality constrained barrier problem (35), we have

$$\begin{pmatrix} \nabla_{p_R} f(p_R, \sigma_R) + A(p_R, \sigma_R)\lambda \\ \nabla_{\sigma_R} f(p_R, \sigma_R) + \acute{A}(p_R, \sigma_R)\lambda \\ -\mu S^{-1} e + \lambda \\ g(p_R, \sigma_R) + s \end{pmatrix} = 0. \quad (42)$$

After applying Newton's method to this system, we have

$$\begin{pmatrix} \nabla^2_{p_R,p_R} L & \nabla^2_{p_R,\sigma_R} L & 0 & A(p_R,\sigma_R) \\ \nabla^2_{\sigma_R,p_R} L & \nabla^2_{\sigma_R,\sigma_R} L & 0 & \acute{A}(p_R,\sigma_R) \\ 0 & 0 & \mu S^{-2} & I \\ A(p_R,\sigma_R) & \acute{A}(p_R,\sigma_R) & I & 0 \end{pmatrix} \begin{pmatrix} d_{p_R} \\ d_{\sigma_R} \\ d_s \\ \lambda^+ \end{pmatrix}$$

$$= \begin{pmatrix} \nabla_{p_R} f(p_R, \sigma_R) \\ \nabla_{\sigma_R} f(p_R, \sigma_R) \\ \mu S^{-1} e \\ -g(p_R, \sigma_R) - s \end{pmatrix} \quad (43)$$

where $\lambda^+ = \lambda + d_\lambda$. Thus, steps d_{p_R}, d_{σ_R} and d_s can be calculated by solving (43). Letting H be the Hessian of the Lagrangian of the barrier problem, we have

$$H = \begin{pmatrix} \nabla^2_{p_R,p_R} L & \nabla^2_{p_R,\sigma_R} L & 0 \\ \nabla^2_{\sigma_R,p_R} L & \nabla^2_{\sigma_R,\sigma_R} L & 0 \\ 0 & 0 & \mu S^{-2} \end{pmatrix}. \quad (44)$$

If the barrier problem is locally convex near the current iteration, i.e., H is positive definite, the algorithm uses this step; otherwise, it uses a CG step, described in the next section.

C. Conjugate Gradient (CG)

The CG approach to solving the approximate problem (35) is similar to other CG calculations. In this case, the algorithm adjusts p_R, σ_R, and s, keeping the slacks s positive.

The approach is to minimize a quadratic approximation to the barrier problem in a trust region, subject to linearized constraints.

The algorithm obtains Lagrange multipliers by approximately solving the KKT equations, subject to λ being positive. Then it takes a step $d = (d_{p_R}, d_{\sigma_R}, d_s)^T$ to approximately solve

$$\min_d \nabla f_\mu^T d + \frac{1}{2} d^T H d \tag{45}$$

subject to

$$(A(p_R, \sigma_R)^T I) d_{p_R} + (\acute{A}(p_R, \sigma_R)^T I) d_{\sigma_R} + g(p_R, \sigma_R) + s = 0$$

where ∇f_μ is the gradient of the barrier problem and is given by

$$\nabla f_\mu = \begin{pmatrix} \nabla_{p_R} f(p_R, \sigma_R) \\ \nabla_{\sigma_R} f(p_R, \sigma_R) \\ -\mu S^{-1} e \end{pmatrix}. \tag{46}$$

To obtain convergence from remote starting points, we introduce a trust region constraint in (45) of the form

$$\left\| \begin{pmatrix} d_{p_R} \\ d_{\sigma_R} \\ S^{-1} d_s \end{pmatrix} \right\| \leq R \tag{47}$$

where $R > 0$ denotes the trust region radius and is updated at every iteration.

To solve (46), the algorithm tries to minimize a norm of the linearized constraints inside a region with radius scaled by R. Then (45) is solved with the constraints being to match the residual from solving (46), staying within the trust region of radius R, and keeping s strictly positive. Since it is not desirable to impede progress of the iteration by employing small trust regions, the slack variables are bounded away from zero by imposing the well-known fraction to the boundary rule [22]

$$s + d_s \geq (1 - \tau) s$$

where the parameter $\tau \in (0, 1)$ is chosen close to 1. Therefore, (45) can be rewritten as follows:

$$\min_d \nabla f_\mu^T d + \frac{1}{2} d^T H d \tag{48}$$

subject to

$$A(p_R, \sigma_R)^T I d_{p_R} + \acute{A}(p_R, \sigma_R)^T I d_{\sigma_R} + g(p_R, \sigma_R) + s = 0 \tag{49}$$

$$\left\| (d_{p_R}, d_{\sigma_R}, S^{-1} d_s) \right\| \leq R \tag{50}$$

$$d_s \geq -\tau s. \tag{51}$$

Although, (48) could be difficult and complex to solve exactly, but we intend to only compute approximate solutions which are sufficiently good solutions [17].

Further details about the optimization method can be found in [17], [20], and [21].

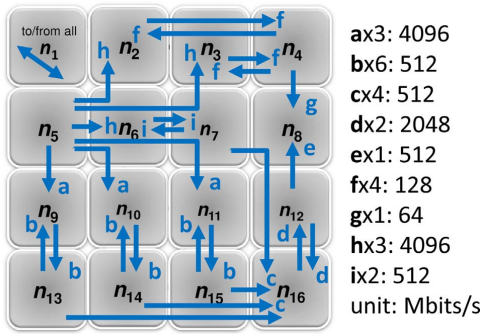


Fig. 10. Ericsson radio systems application.

VII. Experimental Results

A. Experimental Setup

To evaluate the capability of our method, we applied it to a realistic traffic pattern and two synthetic traffic patterns including hot-spot and bit-complement which are mapped to a 4 × 4 2-D mesh network. Although the experiments are performed on a mesh, our method is topology independent.

In this paper, the proposed analytical model is implemented in MATLAB and throughout the experiments, we consider an SoC with 500 MHz frequency, 32-flit packets, and 32-bit flits. We also assume that packets traverse the network on a shortest path using the dimension order XY routing, which is deadlock free.

B. Realistic Traffic Pattern

We used a real application provided by Ericsson Radio Systems [1] as shown in Fig. 10. This application consists of 16 IPs. Specifically, n_2, n_3, n_6, n_9, n_{10}, and n_{11} are ASICs; n_1, n_7, n_{12}, n_{13}, n_{14}, and n_{15} are DSPs; n_5, n_8, and n_{16} are FPGAs; n_1 is a device processor which loads all nodes with program and parameters at startup, sets up, and controls resources in normal operation. Traffic to/from n_1 is for system initial configuration and no longer used afterward. There are 26 node-to-node traffic flows that are categorized into nine types of traffic flows $\{a, b, c, d, e, f, g, h, i\}$, as marked in the figure. The traffic flows are associated with a bandwidth requirement.

As stated before, each flow j is characterized by $(L_j, p_j, \sigma_j, \rho_j)$ that are input parameters of the regulator. We assume L_j and p_j for all flows are the same and equal to 1 *flit* and 1 *flit/cycle*, respectively. ρ_j is determined in *flits/cycle* due to Fig. 10 and also, σ_j can be easily calculated for each flow which its value will be shown in Section VII-B3.

1) *Buffer Size Optimization:* As we mentioned before, a regulator limits a flow injection process with two parameters (peak rate and burstiness). Since there are 26 flows in the example, 52 parameters have to be assigned to regulators. To show that how these parameters heavily affect the required buffer and communication delay, we consider two different regulator sets.

1) Optimized regulators, which are optimized based on the proposed minimizing buffer problem (11).

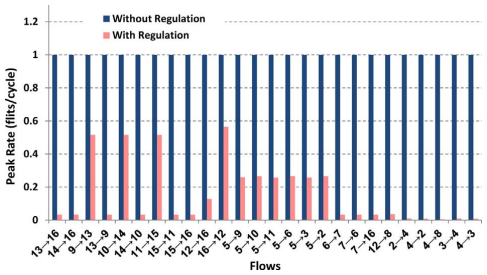


Fig. 11. Peak rate of flows.

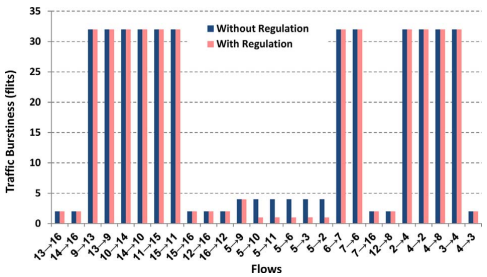


Fig. 12. Traffic burstiness of flows.

TABLE I

COMPARISON OF THE REQUIRED BUFFER BETWEEN DIFFERENT SCHEMES

	Network Buffer	Regulator Buffer	Total Buffer
Without reg.	404	0	404
Optimized reg.	118	28	146
Unoptimized reg.	384	37	421

TABLE II

COMPARISON OF THE MAXIMUM DELAY BETWEEN DIFFERENT SCHEMES

	Network Delay	Regulator Delay	Total Delay
Without reg.	3460	0	3460
Optimized reg.	502	61	563
Unoptimized reg.	3396	163	3559

2) Unoptimized regulators, which are not optimized. Obviously, there is a huge number of unoptimized configurations. We consider a configuration that needs maximum amount of buffers to regulate flows. In fact, we modify the buffer optimization problem (11) to maximize the total number of required buffers instead of minimization.

Then, the total maximum buffer and total maximum delay are calculated and depicted in Tables I and II, respectively, along with values for a system without regulators.

From these tables, we can see that the optimized regulation scheme leads to about 64% reduction in total maximum required buffer and about 84% in total maximum delay when compared with the without regulation scheme. Also these tables show that unoptimized regulators decrease the maximum required buffer and delay in the network because of reducing the contention for shared resources. However, buffer and delay in the regulators are increased to the extent that the total buffer requirements and delay become more than the without regulation scheme because the regulator parameters are not configured appropriately. As a result, we can minimize total buffer cost and improve communications performance by

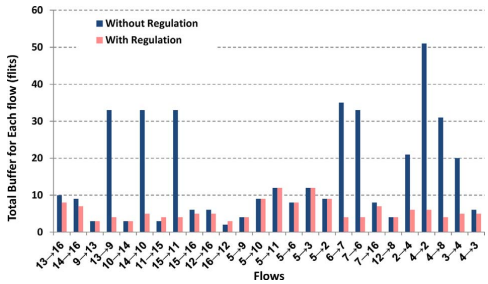


Fig. 13. Maximum required buffers for every flow.

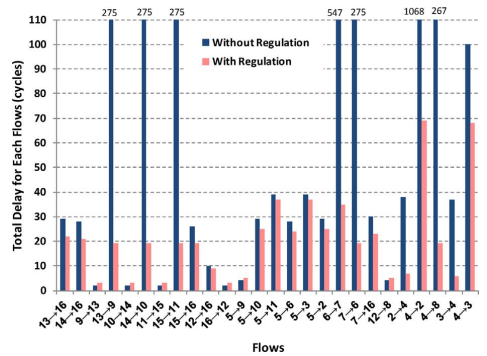


Fig. 14. Maximum worst-case delay for every flow.

TABLE III

COMPARISON BETWEEN DIFFERENT SCENARIOS

	Required Buffer *(flits)*	Variance	
Without regulation	404	436.36	
Minimize-size	146	33.82	
Minimize-variance	192	22.96	
Multiobjective	150	24.29	

consuming a few buffers in the regulator and assigning the peak and burstiness parameters of regulators in a wise manner.

2) *Buffer Variance Optimization:* Identical switches throughout the network may be a constraint in NoC-based systems. Therefore, we have formulated the *Minimize-Variance* optimization problem to design similar switches as far as possible. The results show that if there is no regulator in the network, the sum of variances over different channels of switches is about 436.36, while by controlling flows based on obtained output peak rate and traffic burstiness of solving the *Minimize-Variance* problem, it is equal to 22.96. So, we have about 94% reduction on the sum of variances of buffers.

In this respect, the structures of latter switches are more similar than the former one. It is worth mentioning that if the peak and burstiness parameters of regulators are not appropriately assigned with respect to buffer variance minimization, we may have similar or even more buffer variance in comparison to *without regulation* scheme. For instance, in one of the unoptimized schemes, the sum of variances over different channels of switches is about 436.

3) *Multiobjective Optimization:* As both minimizing total required buffer and buffer variance are important for designers,

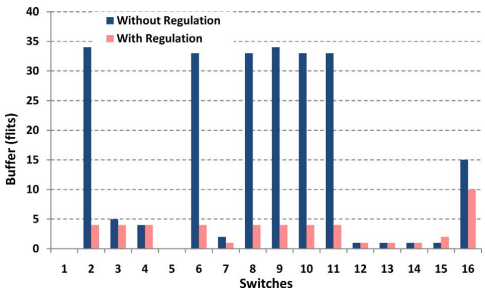


Fig. 15. Maximum required buffers for the ejection channels in switches.

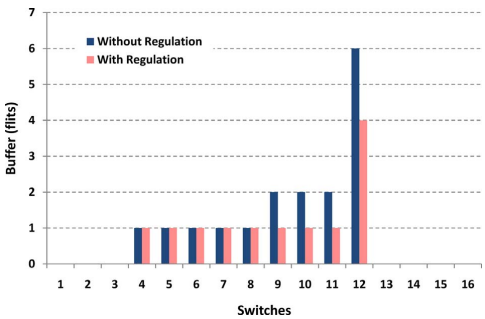


Fig. 16. Maximum required buffers for the southern channels in switches.

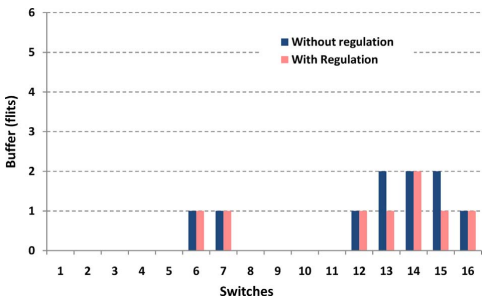


Fig. 17. Maximum required buffers for the northern channels in switches.

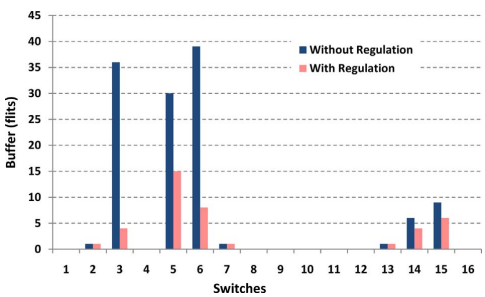


Fig. 18. Maximum required buffers for the eastern channels in switches.

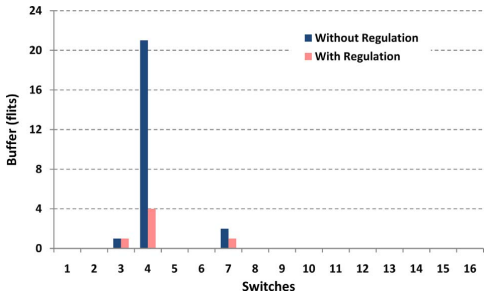


Fig. 19. Maximum required buffers for the western channels in switches.

TABLE IV
COMPARISON OF THE MAXIMUM DELAY BETWEEN DIFFERENT SCENARIOS

	Network Worst-Case Delay	Regulator Worst-Case Delay	Total Worst-Case Delay	Average Worst-Case Delay
Without regulation	3460	0	3460	49.99
With regulation	463	81	544	21.70

we have modeled them as a multiobjective optimization problem. For more detail, we have calculated two parameters *Total Required Buffer* and *Variance* which are listed in Table III.

As can be observed from Table III, *Minimize-Size* problem guarantees that output peak and traffic burstiness selection is carried out in favor of minimizing total required buffer while there is no such guarantee for the sum of variances over various channels. On the contrary, although *Minimize-Variance* yields greater required buffer than *Minimize-Size*, it gives almost the same structure of switches. The results in Table III show that the presented *Multiobjective* problem might be seen as providing a tradeoff between such parameters. Since the *Total Required Buffer* and *Variance* parameters in this problem are very close to their optimal values in *Minimize-Size* and *Minimize-Variance* problems, respectively, they are definitely acceptable for the decision maker. So, in the rest of paper, *with regulation* scheme refers to the regulator which has been optimized for both buffer size and variance.

As can be vividly seen in Figs. 11 and 12, regulators reduce peak rate and traffic burstiness of flows, respectively.

To go into more detail, we depict maximum required buffer and delay of each flow for these schemes in Figs. 13 and 14, respectively. Regarding Fig. 13, it is apparent that in the network with the proposed regulator, most flows require less buffer and also, as mentioned in Table III, total required buffer in this scheme is less than half of it in the network without regulator. Also, Fig. 14 shows that regulated flows can experience longer or shorter delays than other schemes which depends on their requested QoS and also the buffer distribution in the whole network. However, from Table IV, we can see that the total network and average worst-case delay are decreased in the *with regulation* scheme because of buffer-aware allocation in the network and contention reduction for shared resources. We have about 84.3% reduction in total worst-case delay when compared with the *without regulation* scheme.

To better understand the effects of the regulator, maximum required buffers for ejection, southern, northern, eastern, and western channels are revealed in Figs. 15–19, respectively. It is obvious that when regulators control traffic parameters of flows based on the proposed multiobjective problem, the total number of required buffers and their variances are decreased. The *with regulation* scheme leads to about 62.8% reduction in total required buffer and 94.4% reduction on the sum of variances of buffers in comparison to the *without regulation* scheme. So, we have smaller, more similar and more efficient switches. Furthermore, there is desirable QoS in communications through defined constraints in the mentioned multiobjective problem.

C. Synthetic Traffic Patterns

In the case of synthetic traffic patterns, we experimented with hotspot and bit-complement traffic, which represent two extremes of traffic distribution, i.e., unbalanced and balanced workloads.

1) *Hotspot:* in our case, we set a corner node of the 4×4 mesh, node 1, as the hotspot node, and all other nodes send packets to this node.
2) *Bit-complement:* in bit-complement traffic, a node with binary coordinates $b_{n-1}b_{n-2}\ldots b_1b_0$ sends packets only to a node with binary coordinates $\bar{b}_{n-1}\bar{b}_{n-2}\ldots\bar{b}_1\bar{b}_0$. With this workload, all packets must cross the horizontal and vertical network bisections, and the traffic is evenly distributed in the 4×4 network.

For all traffic flows, we set the same values for their maximum packet length L_j and peak rate p_j, which are equal to 1 *flit* and 1 *flit*/*cycle*, respectively. For different flows, rate ρ_j varies between 0.008 and 1 *flits*/*cycle*, and burstiness σ_j between 2 and 32 *flits*. We apply the multiobjective optimization here, which is referred to as *with regulation* scheme. Compared with the optimization of single objectives, it is likely more desirable for designers as it can optimize both buffer size and variance,

Table V compares total maximum required buffer, variance, and total maximum delay under the hotspot traffic pattern. This table reveals that by using optimized regulators, the total maximum required buffer, the variance, and the total maximum delay are reduced by 45.4%, 84.3%, and 58.4%, respectively, in comparison with the *without regulation* scheme.

We also compare these results under the bit-complement traffic pattern in Tables VI. As can be seen from this table, the optimized regulation results in about 49.6% reduction in the total maximum required buffer, 95.1% reduction in the variance, and 64.9% reduction in the total maximum delay.

To present more details, we show the maximum required buffer and delay of each flow under the hotspot traffic in Figs. 20 and 21, respectively. Also, these results under the bit-complement are plotted in Figs. 22 and 23.

The run-time of the proposed method in MATLAB is typically in the order of a few seconds. It is about 2.7 s, 5.76 s, and 0.22 s for the multiobjective optimization of the realistic, hotspot, and bit-complement traffic patterns, respectively. Another interesting point is that the proposed regulator has no negative effect on the network throughput and it is the same with and without the regulation schemes. This is because the flow rates are maintained.

TABLE V
COMPARISON BETWEEN DIFFERENT SCENARIOS UNDER HOTSPOT TRAFFIC

	Network Buffer	Regulator Buffer	Total Buffer	Variance
Without regulation	361	0	361	830.4023
With regulation	144	53	197	129.7305
	Network Worst-Case Delay	Regulator Worst-Case Delay	Total Worst-Case Delay	Average Worst-Case Delay
Without regulation	3328	0	3328	89.10
With regulation	789	597	1386	53.68

TABLE VI
COMPARISON BETWEEN DIFFERENT SCENARIOS UNDER BIT-COMPLEMENT TRAFFIC

	Network Buffer	Regulator Buffer	Total Buffer	Variance
Without regulation	254	0	254	178.73
With regulation	112	16	128	8.72
	Network Worst-Case Delay	Regulator Worst-Case Delay	Total Worst-Case Delay	Average Worst-Case Delay
Without regulation	410	0	410	28.10
With regulation	128	16	144	9.23

VIII. SCOPE AND ASSUMPTION

We discuss possible extensions to address the main assumptions of our approach. We have made two main assumptions.

1) The network routing is deterministic. As such, the path of each flow is determined and thus flow contention becomes predictable. Therefore we can use and have used deterministic network calculus to derive deterministic delay and backlog bounds.
   Deterministic routing has advantages in easier analysis, simplicity, and low implementation overhead. However, it may lead to inferior performance due to being unable to adapt workload to the network congestion status. Due to this limitation, adaptive routing may be favored, though complicating implementation. Adaptive routing means that a flow may use multiple possible paths when delivering packets. For each alternative path, one may find a probability for its use. In such a case, stochastic network calculus [24] can be used to calculate delay and backlog bounds. Still, stochastic network calculus keeps the same fundamentals as the deterministic network calculus. However, the derived delay and backlog bounds will accordingly become stochastic.
2) We assume a static set of flows, which are mapped statically on the network nodes.
   The reason to use static flows with static mapping is that the deterministic analysis relies on known traffic characteristics and known source and destination for each flow. Flows' characteristics may be obtained through traffic profiling. Static mapping can usually facilitate the search of mapping design space in order to find an optimal or near-optimal mapping under performance and energy constraints [10]. As a consequence, the static flows and mapping allow us to apply static regulations on the flows.

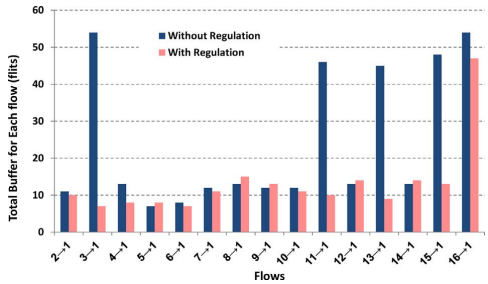


Fig. 20. Maximum required buffers for every flow under hotspot traffic.

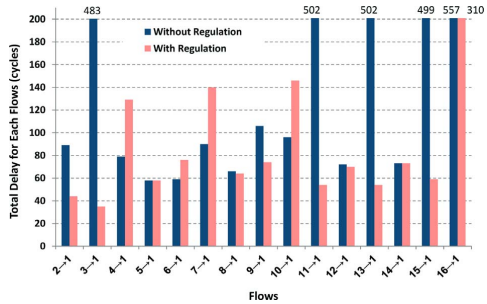


Fig. 21. Maximum worst-case delay for every flow under hotspot traffic.

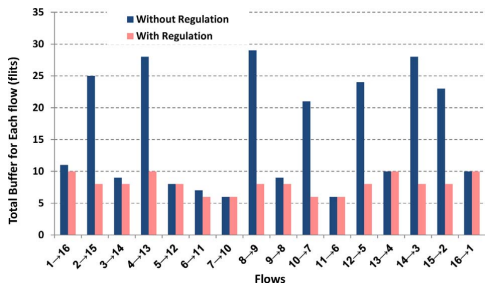


Fig. 22. Maximum required buffers for every flow under Bit-complement.

To alleviate this assumption, there are a few possibilities to enable semi-dynamic and dynamic regulations as we explained as follows.

1) *Semi-dynamic regulation:*
   a) Dynamically changing traffic specifications for each input flow. If a flow's traffic specification may change, we may prepare a set of variants for its parameters. Depending on different traffic specifications, different regulations for the same flow may apply at run-time.
   b) Different use cases and mappings. An application usually contains multiple use cases [25]. For each use case, a set of flows with possible mappings can be pre-compiled. All the use cases must fit into the maximum buffer sizes. These use cases can then be invoked and switched at run-time by reconfiguring the regulators and the network.

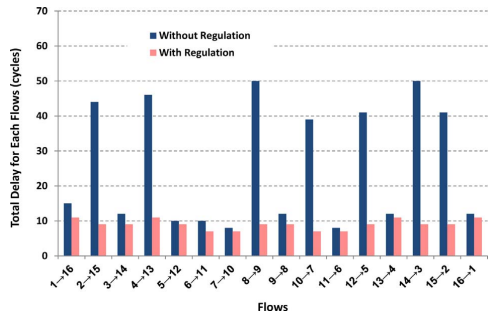


Fig. 23. Maximum worst-case delay for every flow under Bit-complement.

Semi-dynamic configurations can be realized by checking user-defined values of a configurable register in the network interface. Our current regulator implementation in hardware supports re-configuration of regulation parameters at run-time [7].

2) *Dynamic regulation:* we can embed a closed-loop control mechanism in which the network feedback is used as an input to help make regulation decisions. For example, network congestion status could be gathered from the network and then the regulation parameters are adjusted accordingly. This mechanism complicates the regulation mechanisms but has promises in improving performance. In addition, best effort traffic, i.e., traffic without the requirement of delay guarantees, can be better accommodated by allowing them to use the slack bandwidth. The closed-loop control mechanism is currently under our investigation.

IX. Conclusion

IP integration requires the provision of performance guarantees for traffic flows and efficient buffer dimensioning techniques. The regulation changes the burstiness and timing of traffic flows, and thus can be used to control delay and reduce buffer requirements in the SoC. Since a larger fraction of the NoC cost is due to the network buffers, minimizing buffer requirements is an important problem to achieve an efficient NoC implementation. Also, designing similar switches, as far as possible, facilitates the design process of NoC-based systems. In this paper, based on the concepts of formal regulation, we have presented three relevant optimization problems for weighted round robin arbitration, first one for minimizing total required buffers, second one for minimizing the variance of buffers, and last one which is a multiobjective optimization problem for minimizing both of them under QoS requirements. The regulation analysis is performed for best-effort packet switching networks. We have also demonstrated that the proposed model exerts significant impact on communication performance and buffer requirements. The algorithm for solving the proposed minimization problems runs very fast. For the case studies, the optimized solution is found within seconds. Although in this paper we have focused on the output

buffers of switches, our method can be easily adapted to input buffers, too.

REFERENCES

[1] Z. Lu and A. Jantsch, "TDM virtual-circuit configuration for network-on-chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 8, pp. 1021–1034, Aug. 2008.

[2] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf, and T. Henriksson, "Flow regulation for on-chip communication," in *Proc. DATE*, Apr. 2009, pp. 578–581.

[3] J. Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet* (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag, 2004.

[4] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation; part II: Network analysis," *IEEE Trans. Inform. Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

[5] D. Stiliadis and A. Varma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 611–624, Oct. 1998.

[6] C. Chang, *Performance Guarantees in Communication Networks*. London, U.K.: Springer-Verlag, 2000, p. 410.

[7] Z. Lu, D. Brachos, and A. Jantsch, "A flow regulator for on-chip communication," in *Proc. SOCC*, 2009, pp. 151–154.

[8] H. Wang, X. Zhu, L. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *Proc. MICRO*, 2002, pp. 294–305.

[9] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, "System architecture evaluation using modular performance analysis: A case study," *Int. J. STTT*, vol. 8, no. 6, pp. 649–667, 2006.

[10] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. DATE*, 2004, pp. 896–901.

[11] A. E. Kiasari, S. Hessabi, and H. Sarbazi-Azad, "PERMAP: A performance-aware mapping for application-specific SoCs," in *Proc. ASAP*, 2008, pp. 73–78.

[12] A. Jalabert, S. Murali, L. Benini, and G. De Micheli, "xPipesCompiler: A tool for instantiating application-specific NoCs," in *Proc. DATE*, 2004, pp. 884–889.

[13] L. P. Tedesco, N. Calazans, and F. Moraes, "Buffer sizing for multimedia flows in packet-switching NoCs," *J. Integr. Circuits Syst.*, vol. 3, no. 1, pp. 46–56, 2008.

[14] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2919–2933, Dec. 2006.

[15] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Optimal regulation of traffic flows in network-on-chip," in *Proc. DATE*, Mar. 2010, pp. 1621–1624.

[16] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[17] H. Y. Benson, R. J. Vanderbei, and D. F. Shanno, "Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions," *Computat. Optimiz. Applicat.*, vol. 23, no. 2, pp. 257–272, 2002.

[18] P. P. Tang and T. Y. C. Tai, "Network traffic characterization using token bucket model," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 51–62.

[19] F. Gebali and H. Elmiligi, Eds., *Networks on Chip: Theory and Practice*. Boca Raton, FL: CRC Press, 2009.

[20] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, Society for Industrial and Applied Mathematics (SIAM), 2000.

[21] L. M. Adams and J. L. Nazareth, *Linear and Nonlinear Conjugate Gradient-Related Methods*, Society for Industrial and Applied Mathematics (SIAM), 1996.

[22] M. H. Wright, "Interior methods for constrained optimization," *Acta Numerica*. vol. 1, pp. 341–407, Jan. 1992.

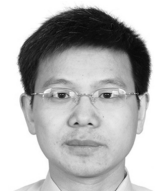
[23] C. A. Coello Coello, "A comprehensive survey of evolutionary based multiobjective optimization techniques," *Knowl. Inform. Syst.: An Int. J.*, vol. 1, no. 3, pp. 269–308, 1999.

[24] Y. Jiang, "A basic stochastic network calculus," in *Proc. Conf. Applicat., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2006, pp. 123–134.

[25] A. Hansson and K. Goossens, "Tradeoffs in the configuration of a network on chip for multiple use-cases," in *Proc. 1st Int. Symp. NoCs*, 2007, pp. 233–242.

Fahimeh Jafari received the B.S. and M.S. degrees in computer engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2002 and 2005, respectively. She is currently pursuing the Ph.D. degree from the Department of Electronic Systems, Royal Institute of Technology, Kista, Stockholm, Sweden.

Her current research interests include design methodologies, interconnection networks, optimization theory, and performance evaluation.



Zhonghai Lu (M'05) received the B.S. degree in radio and electronics from Beijing Normal University, Beijing, China, in 1989, and the M.S. degree in system-on-chip design and the Ph.D. degree in electronic and computer systems design, both from the Royal Institute of Technology (KTH), Kista, Stockholm, Sweden, in 2002 and 2007, respectively.

From 1989 to 2000, he worked extensively on the areas of electronic and embedded systems. He took research visits to Samsung Electronics, Seoul, Korea, the National Institute of Informatics, Tokyo, Japan, and the Swiss Federal Institute of Technology, Zürich, Switzerland. He is currently a Senior Researcher with the Department of Electronic Systems, School of Information and Communication Technology, KTH. His current research interests include network-on-chip/system-on-chip, multicore computing architectures, cyber-physical systems, performance analysis, and design automation. He has published about 70 papers in these areas.

Axel Jantsch (M'97) received the Dipl.Ing. and Dr. Tech. degrees from the Technical University of Vienna, Vienna, Austria, in 1988 and 1992, respectively.

He was with Siemens Austria, Vienna, Austria, as a System Validation Engineer from 1995 to 1997. Since 1997, he has been an Associate Professor with the Royal Institute of Technology (KTH), Kista, Stockholm, Sweden. Since 2000, has been a Docent, and since December 2002, a Full Professor of Electronic System Design with the Department of Electronic Systems. He has published over 200 papers in international conferences and journals, and one book in the areas of very large scale integration design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip.

Dr. Jantsch received the Alfred Schrödinger Scholarship from the Austrian Science Foundation while a Guest Researcher with KTH between 1993 and 1995. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES+ISSS, SOC, NOCS, and others. He has been the TPC Chair of SSDL/FDL 2000, the TPC Co-Chair of CODES+ISSS 2004, the General Chair of CODES+ISSS 2005, and the TPC Co-Chair of NOCS 2009. From 2002 to 2007, he was a Subject Area Editor for the *Journal of System Architecture*. At KTH, he is heading a number of research projects involving a total number of ten Ph.D. Students, in two main areas: system modeling and networks-on-chip.

Mohammad Hossein Yaghmaee (M'09) was born in Mashhad, Iran, in July 1971. He received the B.S. degree in communication engineering from the Sharif University of Technology, Tehran, Iran, in 1993, and the M.S. and Ph.D. degrees in communication engineering from the Tehran Polytechnic (Amirkabir) University of Technology, Tehran, in 1995 and 2000, respectively.

Since 1992, he has been a Computer Network Engineer with several networking projects at the Iran Telecommunication Research Center, Tehran, Iran. From November 1998 to July 1999, he was a Visiting Research Scholar with the Network Technology Group, C&C Media Research Laboratories, NEC Corporation, Tokyo, Japan. From September 2007 to August 2008, he was a Visiting Associate Professor with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown. He is currently an Associate Professor with the Computer Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad. He is the author of four books, all in Farsi. He has published more than 90 international conference and journal papers. His current research interests include wireless sensor networks, traffic and congestion control, high-speed networks including ATM and MPLS, quality of services, and fuzzy logic control.

Paper 13

Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels

F. Jafari
Z. Lu
A. Jantsch

Submitted to ACM Transactions on Design Automation of Electronic Systems (TODAES).

A

Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels

FAHIMEH JAFARI, KTH Royal Institute of Technology, Sweden
ZHONGHAI LU, KTH Royal Institute of Technology, Sweden
AXEL JANTSCH, KTH Royal Institute of Technology, Sweden

Real-time applications such as multimedia and gaming boxes require stringent performance guarantees, usually enforced by a tight upper bound on the maximum end-to-end delay. We consider worst-case delay bounds for Variable Bit-Rate (VBR) flows in a FIFO multiplexing on-chip network with aggregate scheduling, which schedules multiple flows as an aggregate flow. In this paper, flows are characterized by a maximum transfer size (L), peak rate (p), burstiness (σ), and average sustainable rate (ρ). Based on network calculus, we present and prove technical theorems to analyze performance. We use the theorems to derive per-flow end-to-end equivalent service curve employed for computing Least Upper Delay Bounds (LUDBs) of individual flows. We then implement algorithms employed in our methodology. A realistic case study exhibits that the end-to-end delay bound is up to 46.9% more accurate than the case without considering the traffic peak behavior. Simulation results look into the accuracy of the proposed analysis method. Likewise, the experimental results demonstrate similar improvements in the case of synthetic traffic patterns.

Additional Key Words and Phrases: Network-on-chip (NoC), performance evaluation, worst-case delay bound, FIFO multiplexing

1. INTRODUCTION

In networks-on-chip, resources like wires, buffers, and switches are shared among multiple communication flows to provide cost efficiency. At the same time many applications have real-time requirements and, consequently, delay and throughput constraints on the communication. To guarantee maximum delay and minimum throughput for one given communication flow, the interference in the shared resources from other flows has to be analyzed and bounded. We assume that all traffic can be well characterized as flows and scheduled as aggregate which means multiple flows are scheduled as an aggregate flow. For a given flow, we study the maximum interference of all other flows based on the network calculus theory [Le Boudec et al. 2004].

In network calculus, flows are characterized as *arrival curves* and the service offered to flows by a network element such as a link or a switch is abstracted as *service curve*. Since the network contention for shared resources includes not only direct contention but also indirect contention, predicting the worst-case performance is extremely hard.

To calculate the accurate delay bound per flow, the main problem is to obtain the *end-to-end Equivalent Service Curve (ESC)* and internal output arrival curves of individual flows in an arbitrary network of servers in terms of the latencies of the individual schedulers in the network. Since the required theorems for calculating performance metrics of VBR traffic transmitted in the FIFO order and scheduled as aggregate have not been represented so far, we have defined and proved them based on network calculus [Chang 2000; Le Boudec et al. 2004] in [Jafari et al. 2011] and [Jafari et al. 2012]. In [Jafari et al. 2011], we proposed and proved the required theorem for deriving the output characterization of VBR traffic under the defined system model to have exact vision about output metrics used for obtaining performance bounds. In [Jafari et al. 2012], the required theorems for computing end-to-end ESC and end-to-end delay bound are defined and proved. Moreover, we presented a simple example to show how the proposed theorems can be used in the network. The method presented in [Jafari et al. 2012] only considers direct contentions of a tagged flow. In this paper, we use the proposed theorems in [Jafari et al. 2011; Jafari et al. 2012] to present a formal approach for performance analysis modeling both direct and indirect contentions.

VBR is a class of traffic in which the rate can vary significantly from time to time, containing bursts. Real-time compressed voice and video and time-sensitive bursty data traffic are examples of VBR traffic. Real-time VBR flows can be characterized by a set of four parameters, (L, p, σ, ρ), where L is the maximum transfer size, p peak rate, σ burstiness, and ρ average sustainable rate [Le Boudec et al. 2004]. For instance, in a NoC with a link data width of 32 *bits*, frequency of 500 *MHz*. This means a link bandwidth of 16 $Gbits/s$ (32 $bits \times 500$ *MHz*). An HDTV video stream can be characterized with $L = 32$ *bits*, $p = 16$ $Gbits/s$, $\sigma = 960$ *Kbits*, $\rho = 76$ $Mbits/s$. Our assumption is that the application-specific nature of the network enables to characterize traffic with sufficient accuracy.

For an individual flow, called a *tagged flow*, we first consider resource sharing scenarios (channel sharing, buffer sharing, and channel&buffer sharing) in the routers and then build analysis models for different resource sharing components. We assume that the routers employ round robin scheduling to share the link bandwidth. Based on these models, we can derive the intra-router ESC for an individual flow. To consider the contention which a flow may experience along its routing path, we present a recursive algorithm to classify and analyze flow interference patterns. The algorithm uses the proposed theorems to analyze the effect of contention flows on the tagged flow. Based on this algorithm, we derive the end-to-end ESC and then Least Upper Delay Bound (LUDB) for a tagged flow under the mentioned system model. To show the potential of our method, we experiment three case studies to derive delay bounds and compare them with simulation results. It is worth mentioning that the paper does not deal with the back-pressure, but calculates the buffer size thresholds to make sure the back-pressure does not occur in the network.

The remainder of this paper is organized as follows. Section 2 gives an account of related works. In Section 3, we introduce the basics of network calculus. Section 4 discusses the underlying system model and notations in our analysis. Section 5 is devoted to the theorems required for computation of performance metrics. We present our formal method for the performance analysis and computation of LUDB in Section 6. Numerical results are reported in Section 7. Finally, Section 8 gives the conclusions and highlight directions for future work.

2. RELATED WORK

Recently, NoC designers have a great deal of interest in the development of analytical performance models [Bakhouya et al. 2011]. Authors in [Ogras et al. 2005] give a unified representation of NoC architectures and applications and consider some major research problems in the design area. As represented in this work, most of research problems need to analytically analyze and evaluate performance metrics in the network. In [Kiasari et al. 2013], we have surveyed four popular mathematical formalisms -*dataflow analysis*, *schedulability analysis*, *queueing theory*, and *network calculus*- along with their applications in NoCs. Also, we have reviewed strengths and weaknesses of each technique and its suitability for a specific purpose.

Dataflow analysis is a deterministic approach based on graph theory. As an example, authors in [Hansson et al. 2008] present a model using a cyclo-static dataflow graph for buffer dimensioning for NoC applications. In dataflow analysis, it is assumed that the pattern of communication among cores and switches are deterministic and predefined. Dataflow analysis must be used with restricted models such as SDF and CSDF to capture dynamic behavior. In other words, the expressiveness is typically traded off against analyzability and implementation efficiency in this formalism.

Schedulability analysis is an analytical approach for investigating the timing properties in real-time systems. It gets a set of tasks, their worst-case execution time, and a scheduling policy as inputs and determines whether these tasks can be scheduled such

that deadline misses never occur. One example of this approach in NoCs is presented in [Shi et al. 2008]. Schedulability analysis uses simpler event models compared to the other mathematical formalisms and consequently the performance model is easily extracted with less accuracy.

The proposed models in [Lee 2003], [Rahmati et al. 2009], and [Rahmati et al. 2013] are inspired by schedulability analysis. In [Lee 2003], the author presents a worst-case analysis model for real-time communication and also proposes a feasibility test algorithm for a simplex virtual circuit in wormhole networks. This work is extended by [Rahmati et al. 2009] towards NoCs, computing real-time bounds for high bandwidth traffic. In [Rahmati et al. 2013], the authors extend the model to provide more detailed switch models and consider virtual channels and variable buffer lengths. The key advantage of these methods is that they compute the worst-case bounds with low time complexity without any special hardware support, but the main limitation is that they do not leverage the input arrival patterns, which it leads to over approximations of the performance analysis.

Most of the current works use *queuing theory*-based approaches. For example, authors in [Moadeli et al. 2007] analyze the traffic behavior in a NoC with the spidergon topology and wormhole routing and then present a queuing-theory-based analytical model for evaluating the average message latency in the network. In [Ben-Itzhak et al. 2011], authors propose an analytical model for deriving average end-to-end delay in a heterogeneous wormhole based NoC with heterogeneous traffic patterns, non-uniform link capacities and a variable number of virtual channels per link. Queuing approaches often use probability distributions like Poisson to model traffic in the network while Poisson distribution used in queuing model is not appropriate for characterizing traffic patterns in NoC applications because it is not able to model all significant features in this network. Queuing theory generally evaluate average quantities of metrics in an equilibrium state and characterizing the transient behavior is a very difficult problem. An approach for addressing this problem is suggested in [Bogdan et al. 2007]. Authors in this work proposed a statistical physics-inspired framework to model the information flow and buffers behavior in NoCs. They analyze the traffic dynamics in NoCs and effectively capture the nonstationary effects of the system workload. In following up to this work, authors in [Bogdan et al. 2010] proposed QuaLe model based on statistical physics that can account for nonstationary observed in packet arrival processes. They also investigated the impact of packet injection rate and the data packet sizes on the multifractal spectrum of NoC traffic.

Network calculus is a mathematical framework for deriving worst-case bounds on maximum latency, backlog, and minimum throughput in network-based systems. It is able to model all traffic patterns with bounds defined by arrival curves. In this respect, designers can capture some dynamic features of the network based on shapes of the traffic flows [Bakhouya et al. 2011]. Network calculus can also abstract many scheduling algorithms and arrival classes at single queue with multiplexed arrival flows, by service curves. The service curves through a network can be convolved as a single service curve. Hence a multi-node network analysis can be simplified to a single-node analysis. Regarding these two features, network calculus can analyze many scheduling algorithms and arrival classes over a multi-node network in a uniform framework while classical queuing theory separately models different combination of them [Ciucu et al. 2012]. The probabilistic version of (deterministic) network calculus is stochastic network calculus. In some networks, such as wireless networks, the service offered by a communication channel may vary randomly over time due to channel contention and impairment. Such networks can only provide stochastic services and guarantees. For example, authors in [Rizk et al. 2012] use stochastic network calculus to derive per-flow end-to-end performance bounds in a network of tandem queues under open-loop

fBm cross traffic which is a model for self-similar and long-range dependent aggregate Internet traffic. Since we employ deterministic network calculus, in the rest of our paper, network calculus refers to the deterministic type. In [Bakhouya et al. 2011], authors present a network calculus-based methodology for analysis and evaluation of on-chip interconnects in terms of performance and cost metrics, such as latency, energy consumption, and area requirements. Authors in this paper compare 2D mesh, spidergon, and WK-recursive topologies using a given traffic pattern and show that WK-recursive outperforms mesh and spidergon in all considered metrics. The proposed model in this paper is simple without considering virtual channel effects and modeling all interferences between flows sharing a resource in the network. Moreover, the model does not investigate the peak behavior of flows which leads to less accurate bounds while we consider performance analysis for VBR traffic in on-chip networks employing aggregate resource management.

The performance evaluation of real-time services in networks employing aggregate scheduling is particularly challenging because of its complexity. Aggregate scheduling arises in many cases. In addition to NoC, for example, it can also be applied for obtaining scalability in large-size networks. The Differentiated Services (DiffServ) [Blake et al. 1998] is an example of an architecture based on aggregate scheduling in the Internet. Despite the research efforts, few results have appeared on this subject. A survey on the subject can be found in [Bennett et al. 2002]. The authors in [Charny et al. 2000] consider a closed-form delay bound for a generic network configuration under the fluid model assumption. It is also extended in [Jiang 2002] to consider packetization effects. However, these works can derive bounds only for small utilization factors in a generic network configuration.

Authors in [Martin et al. 2006; Martin et al. 2003; Bauer et al. 2010] employ Trajectory Approach (TA) to compute end-to-end delay bounds in FIFO systems. The Trajectory Approach computes all the possible trajectories of a system under constraints and then takes maximum end-to-end delays on them. [Bauer et al. 2010] compares Network Calculus and the Trajectory approaches on a real avionics AFDX configuration and shows that The Trajectory approach computes upper bounds which are tighter than the upper bounds computed by the network calculus one. However, authors derive delay bounds by summing per-node bounds, expectedly not arriving at tight bounds but reported as being at least close under practical conditions.

The computation of delay bounds through network calculus in feed-forward networks under arbitrary multiplexing has already been addressed in different lectures such as [Schmitt et al. 2008; Kiefer et al. 2010; Bouillard et al. 2010]. In [Bouillard et al. 2010], authors describe the first algorithm which can compute the worst-case end-to-end delay for a given flow for any feed-forward network under blind multiplexing, with concave arrival curves and convex service curves. Since the problem is intrinsically difficult (NP-hard), they show that in some cases, like tandem networks with cross-traffic interfering along intervals of servers, the complexity becomes polynomial. Then, authors in [Bouillard et al. 2011] refine the approach of [Bouillard et al. 2010] in order to take into account fixed priorities. They study networks with a fixed priority service policy which means each flow is assigned a fixed priority and try to take into account the pay multiplexing only once (PMOO) phenomenon. This stream of works deal with networks of arbitrary multiplexing also known as general or blind multiplexing, which means no assumption is made about the service policy while by assuming an explicit multiplexing scheme like FIFO, tighter bounds can be obtained.

Authors in [Lenzini et al. 2006; Lenzini et al. 2008; Bisti et al. 2010] propose a methodology which calculates delay bounds in tandem networks of rate-latency nodes traversed by leaky bucket shaped flows. They also introduce a software tool, called DEBORAH, which implements algorithms employed in their methodology to compute

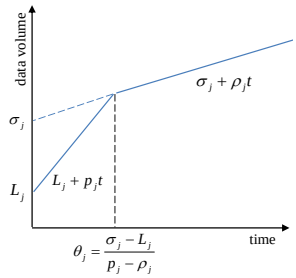


Fig. 1. Arrival curve of flow f_j with TSPEC $(L_j, p_j, \sigma_j, \rho_j)$.

delay bounds. These works consider servers in tandem or sink trees, while our proposed method computes end-to-end delay in a generic topology of NoC. Moreover, these works investigate computing delay bounds only for average behavior of flows and they do not consider peak behavior, which results in less accurate bounds.

In [Boyer 2010], the authors try to model shaping for an end-to-end delay where each server is shared by two flows. An applicative token bucket $\gamma_{r,b}$ is shaped by the bit-rate of the link λ_R, leading to a two-slopes affine arrival curve which this arrival curve is similar to one we consider for double leaky buckets. The paper investigates a simple topology, a sequence of rate-latency servers, each one shared by two flows with a FIFO policy, and a simple case of nested contentions. Moreover, authors state that their modeling is incomplete: when computing the worst-case traversal time of a flow, they model only the shaping on the considering flow, not on the interfering ones (leading to the title 'half-modeling of shaping') In this paper, we investigate both nested and crossed contentions in general to model all flows (even interfering ones) with complex interferences in on-chip networks.

All aforementioned works in the subject of aggregate resource management compute delay bounds in various network infrastructures but not on-chip networks. As regards to NoC architecture, analytical models are very close to the reality of the system. For instance, a router in on-chip networks can be modeled in pure hardware which means the micro-architecture is feasible for analysis. Therefore, network calculus can provide the analysis more accurate in on-chip networks.

Authors in [Qian et al. 2010] present analytical models for traffic flows under strict priority queueing and weighted round robin scheduling in on-chip networks. They then derive per-flow end-to-end delay bounds using these models. Like most of mentioned works, [Qian et al. 2010] does not deal with peak behavior of flows, which results in less accurate bounds. The proposed method in this paper considers performance analysis for VBR traffic characterized by (L, p, σ, ρ) in on-chip networks employing aggregate resource management. As such, our method achieves more accurate delay bounds.

3. NETWORK CALCULUS BACKGROUND

Network calculus is a mathematical framework to derive worst case bounds and analyze performance guarantees in networks.

This paper uses Traffic SPECification (TSPEC) [Wroclawski 1997] to model the average and peak characteristics of flow f_j as *arrival curve* $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate $(p_j \geq \rho_j)$, σ_j the burstiness $(\sigma_j \geq L_j)$, and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. As

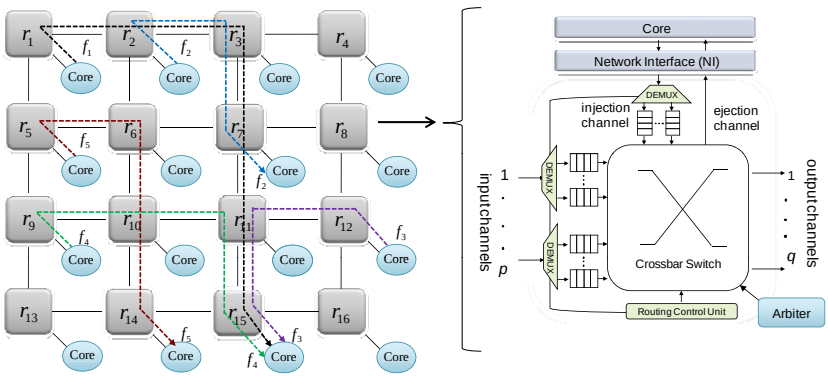


Fig. 2. An example of an NoC with 16 nodes and 5 flows along with the structure of a single node.

shown in Figure 1, $\theta_j = (\sigma_j - L_j)/(p_j - \rho_j)$ and $\alpha_j(t) = L_j + p_j t$ if $t \leq \theta_j$; $\alpha_j(t) = \sigma_j + \rho_j t$, otherwise.

In this paper, we also consider a class of curves, namely pseudoaffine curves [Lenzini et al. 2006], which is a multiple affine curve shifted to the right and given by $\beta = \delta_T \otimes [\otimes_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$. In fact, a pseudoaffine curve represents the service received by single flows in tandems of FIFO multiplexing rate-latency nodes. Due to concave affine curves, it can be rewritten as $\beta = \delta_T \otimes [\wedge_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$, where the non-negative term T is denoted as *offset*, and the affine curves between square brackets as leaky-bucket stages. It is clear that a rate-latency service curve is in fact pseudoaffine, since it can be expressed as $\beta = \delta_T \otimes \gamma_{0,R}$.

Given arrival curve α and service curve β, the delay is bounded by the horizontal deviation between the arrival and service curves.

4. SYSTEM MODEL AND NOTATIONS

As depicted in Figure 2, we consider an NoC architecture in which every node contains a router and a core which performs its own computational, storage or I/O processing functionality, and is equipped with a Network Interface (NI). As you can see in the figure, buffers are arranged to construct VCs in each input channel. To characterize flows based on their defined TSPEC, we assume unbuffered leaky bucket controllers (regulators) which do not buffer the packets, but stall the traffic producers or IPs [Jafari et al. 2010].

Assumptions in this work are listed as follows:

— The NoC architecture can have different topologies.
— Packets have fixed length and traverse the network in a best-effort fashion with virtual-cut-through switching technique using a deadlock-free deterministic routing.
— Routers have only input buffers and VCs.
— Buffers are bounded and the network is lossless.
— The router can have multiple VCs per in-port. VC allocation is deterministic and each VC receives an aggregate service.
— All traffic is the part of TSPEC flows $f = TSPEC(L, p, \sigma, \rho)$ at the entry into the network.

— In each node that guarantees to serve the flow a pseudo affine service curve $\beta = \delta_T \otimes \gamma_{\sigma_x,\rho_x}$, it is assumed that $\rho \leq \rho_x$ and $p \geq \rho_x$.
— Flows are classified into a pre-specified number of aggregates.
— Traffic of each aggregate is buffered and transmitted in the FIFO order, denoted as FIFO multiplexing.
— Different aggregates are buffered separately and each aggregate is guaranteed a rate-latency service curve.
— We use a concrete policy, in this case, round-robin arbitration, to support the assumption on rate-latency service curve. Indeed, it can use some other arbitration policies as well. We also assume a fixed word length of L_w in all of flows.
— The peak rate is limited by the hardware. It is always 1 *flit*/*cycle*.

NoC designers can obtain per flow end-to-end delay bound in NoC architectures by the proposed method in this paper under the mentioned assumptions.

Most of assumptions in this paper have been widely used by [Qian et al. 2009; Jafari et al. 2010]. The system model in this paper is more general than [Qian et al. 2009; Jafari et al. 2010]. In [Qian et al. 2009; Jafari et al. 2010], authors consider a Constant Bit Rate (CBR) flow in NoCs, defined by (σ, ρ) which is a special case of TSPEC. Furthermore, authors in [Jafari et al. 2010] presume the number of VCs for each PC is the same as the number of flows passing through that channel while we have relaxed this limitation in the paper.

We use an example depicted in Figure 2 to explain terminology used in the paper. The figure shows a network with 16 nodes numbered from $1, 2, ..., 16$ connected by links. There are 5 flows in the example denoted as $f_1, ..., f_5$. Multiple flows share the same buffer and channel in the router are scheduled as a flow called aggregate flow. For instance, $f_{\{1,2\}}$ in router 3 is an aggregate flow. A *tagged flow* is the flow that we shall derive its delay bound and other flows that share resources with the tagged flow are *contention flows*. In this example, f_1 is the tagged flow, and f_2, f_3, and f_4 are contention flows. Notations in the paper are listed in Table I.

We use sub-index "(f_i, r_j)" for notations to indicate that they are related to flow f_i in router r_j. For example, $\alpha_{(f_1,r_2)}$ denotes the arrival curve of flow f_1 in router r_2. We also employ sub-index "(s_i, r_j)" to state notations are related to f_{s_i} in router r_j. In this case, f_{s_i} can be one flow or an aggregate flow. For instance, $\beta_{(\{1,2,3\},r_2)}$ indicates the service curve of aggregate flow $f_{\{1,2,3\}}$ in router r_2.

5. PROPOSED THEOREMS

In this section, we review the required theorems, proposed in [Jafari et al. 2011; Jafari et al. 2012], for analyzing performance of VBR flows in a FIFO multiplexing network.

We first represent a theorem for computing delay bound as follows.

Theorem 1. *(Delay Bound) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage γ_{σ_x,ρ_x}, $1 \leq x \leq n$, this means we have:*

$$\beta = \delta_T \otimes [\otimes_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}]$$

and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \geq \rho$ ($\rho^*_\beta = min_{1\leq x\leq n}\rho_x$), then the maximum delay for the flow is bounded by

$$h(\alpha, \beta) = T + \left[\vee_{1\leq x\leq n}\frac{L - \sigma_x + \theta\,(p - \rho_x)^+}{\rho_x}\right]^+ \quad (1)$$

PROOF. We have proved it in [Jafari et al. 2012]. See Appendix A.

Table I. The list of notations

f_i	Flow i
α_i	The arrival curve of f_i
α_i^*	The output arrival curve of f_i
L_i	The maximum transfer size of f_i *(flits)*
p_i	The peak rate of f_i *(flits/cycle)*
σ_i	The burstiness of f_i *(flits)*
ρ_i	The average rate of f_i *(flits/cycle)*
$Src(i)$	The source node of f_i
r_j	Router j
β_j	The service curve of r_j
R	The minimum service rate in a rate-latency service curve
T^l	The maximum processing latency of the arbiter in the router *(cycles)*
T^{HoL}	The maximum waiting time in the FIFO queue of the router *(cycles)*
T^{Total}	The total processing delay which comes from contention flows and equals to the sum of T^l and T^{HoL} *(cycles)*
D_{router}	Time spent for packet routing decision *(cycles)*
L_w	The word length in the flow *(flits)*
C	The channel capacity *(flits/cycle)*
ρ_x	The minimum service rate in a pseudo affine service curve
CF_t	The set of contention flows of tagged flow f_t in the network
s_i	The set of joint flows in an aggregate flow (when the number of elements of s_i is equal to 1, there is only a single flow)
f_{s_i}	An aggregate flow of s_i
$\|s_i\|$	The cardinality of set s_i, which is a measure of the "number of elements of the set"
$S = \{s_i\}$	A set of s_i's in a tandem of routers
s^m	A set which has the maximum cardinality between the sets in S. $s^m = \{s_x \| \|s_x\| = max(\|s_i\|); \forall s_i \in S\}$
f_{s^m}	The flow related to s^m
r^m	The router related to s^m
β^m	The service curve related to s^m
$F^B_{(s_i,r_j)}$	The set of flows which share the same buffer in router r_j with flow f_{s_i}
$\left\|V_{(s_i,r_j)}\right\|$	The number of virtual channels that passing flows from them share the same channel of router r_j with flow f_{s_i}
$F_{(VC_k,PC_i,r_j)}$	The set of flows passing through VC_k in physical channel PC_i of router r_j

In the rest of the paper, we apply Theorem 1 on the end-to-end ESC to calculate LUDB for a tagged flow. Due to our proposed method in Section 6, to obtain the end-to-end ESC, we should able to subtract contention flows from a service curve. To this end, we propose Proposition 1 and Theorem 2. In Proposition 1, we derive ESC with FIFO multiplexing where service curve is a pseudo affine curve. We then use Corollary 1 which is an immediate consequence of Proposition 1 to propose Theorem 2. This theorem is employed for deriving ESC in the underlying system model.

In Proposition 1 and Theorem 2, we obtain ESC with FIFO multiplexing under different assumptions.

Proposition 1. *(Equivalent Service Curve) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage γ_{σ_x,ρ_x}, $\quad 1 \leq x \leq n$, this means we have:*

$$\beta = \delta_T \otimes [\otimes_{1\leq x\leq n} \gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\leq x\leq n} \gamma_{\sigma_x,\rho_x}]$$

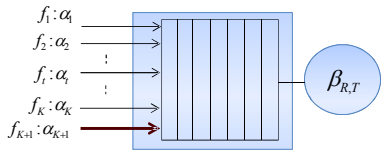


Fig. 3. Computation of equivalent service curve for flow $K+1$ in a rate-latency node.

and let $\alpha = min(L+pt, \sigma+\rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. *If* $\rho_\beta^* \geq \rho$ $(\rho_\beta^* = min_{1\leq x\leq n}\rho_x)$ *and* $p \geq \rho_\beta^\circ$ $(\rho_\beta^\circ = max_{1\leq x\leq n}\rho_x)$, *then the ESC obtained by subtracting arrival curve* α, $\{\beta^{eq}(\alpha,\tau), \quad \tau = h(\alpha,\beta)\} \equiv \beta^{eq}(\alpha)$, *with*

$$\beta^{eq}(\alpha) = \delta_{T+\vee_{1\leq i\leq n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ +\theta} \otimes \left[\otimes_{1\leq x\leq n}\left[\gamma_{\rho_x}\left\{\vee_{1\leq i\leq n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ - \frac{\sigma-\sigma_x-(\rho_x-\rho)\theta}{\rho_x}\right\},\rho_x-\rho\right]\right] \tag{2}$$

PROOF. We have proved it in [Jafari et al. 2012]. See Appendix B.

The following corollary is an immediate consequence.

Corollary 1. *Let* $\beta = \delta_T \otimes \gamma_{\sigma_x,\rho_x}$ *be a pseudo affine curve, with offset* T *and one leaky-bucket stage* γ_{σ_x,ρ_x}, *and let* $\alpha = min(L+pt, \sigma+\rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. *If* $\rho_x \geq \rho$ *and* $p \geq \rho_x$, *then the ESC obtained by subtracting arrival curve* α, β^{eq}

$$\beta^{eq} = \delta_{T+\left[\frac{L-\sigma_x+\theta(p-\rho_x)^+}{\rho_x}\right]^+ +\theta} \otimes \gamma_{0,\rho_x-\rho} \tag{3}$$

PROOF. We can easily obtain this corollary by applying Proposition 1 for service curve β when $n=1$.

We can specifically capitalize on Corollary 1 to obtain a parametric expression for the ESC of a tagged flow passing through a rate-latency node. We assume the number of flows passing through this node is $K+1$. Therefore, for computing equivalent service curve for the tagged flow, we should subtract the arrival curves of other K flows. It can be calculated by iteratively applying Corollary 1 for K times. Without loss of generality, we presume that the tagged flow is flow $K+1$. We now present following theorem:

Theorem 2. *(Equivalent Service Curve for Rate-Latency Service Curve with* $K+1$ *Flows) Consider one node with a rate-latency service curve* $\beta_{R,T} = \delta_T \otimes \gamma_{0,R}$. *Let* $\alpha_i = min(L_i+p_it, \sigma_i+\rho_it) = \gamma_{L_i,p_i} \wedge \gamma_{\sigma_i,\rho_i}$ *be arrival curve of flow* i *and* $p_i \geq R - \sum_{(j=1;j\neq i)}^{K+1}\rho_j$, *where* $1 \leq i \leq K+1$ *and* $K+1$ *is the number of flows passing through that node as shown in Figure 3. Assuming* $\sum_{j=1}^{K+1}\rho_j \leq$ *link rate, where* C *is the link rate, that the ESC for flow* $K+1$ *in the node, obtained by subtracting* K *arrival curves, is:*

$$\beta_{K+1}^{eq} = \delta_{T+\sum_{i=1}^{K}\left(\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+ +\theta_i\right)} \otimes \gamma_{0,R-\sum_{j=1}^{K}\rho_j} \tag{4}$$

PROOF. We have proved it in [Jafari et al. 2012]. See Appendix C.

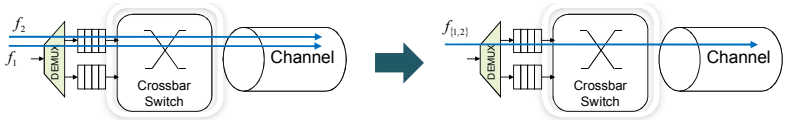


Fig. 4. An example of channel&buffer sharing.

Theorem 3 states how output arrival curve of a VBR flow in a FIFO multiplexing node can be calculated.

Theorem 3. *(Output Arrival Curve with FIFO) Consider a VBR flow, with TSPEC* (L, p, ρ, σ)*, served in a node that guarantees to the flow a pseudo affine service curve* $\beta = \delta_T \otimes \gamma_{\sigma_x,\rho_x}$*. The output arrival curve* α^* *given by:*

$$\alpha^* = \begin{cases} \theta > T & \gamma_{(p\wedge\rho_x)T+\theta(p-\rho_x)^+ + L-\sigma_x, p\wedge\rho_x} \\ & \wedge\gamma_{\sigma-\sigma_x+\rho T,\rho} \\ \theta \leq T & \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{cases} \tag{5}$$

PROOF. We have proved it in [Jafari et al. 2011]. See Appendix D.

We apply this theorem to calculate internal output arrival curves. For instance, in Section 6.2, we obtain the output arrival curve of a crossed flow when it is split into two nested flows.

6. FORMAL METHOD FOR LUDB DERIVATION

We have presented and proved the required theorems for deriving LUDB for VBR flows in on-chip networks based on aggregate scheduling with multiple virtual channels. As mentioned before, to calculate LUDB per flow, we should first obtain the end-to-end ESC which the tandem of routers provides to the flow. For calculating the end-to-end ESC, we propose two following steps:

- **Step 1:** Intra-router ESC
- **Step 2:** Inter-router ESC

In the first step, we consider resource sharing scenarios in the routers and then build analysis models for different resource sharing components. Based on these models, we can derive the intra-router ESC for an individual flow. In the second step, we consider the contention which a flow may experience along its routing path. Therefore, we present recursive algorithm End-to-End ESC to classify and analyze resource sharing models and flow interference patterns. Based on this algorithm, we can derive the end-to-end ESC for a tagged flow passing through the tandem of routers.

6.1. Step1: Intra-router ESC

To compute intra-router ESC for a tagged flow, it is necessary to investigate resource sharing. At each router, we identify three types of resource sharing, namely, *channel sharing*, *buffer sharing*, and *channel&buffer sharing*. *Channel sharing* means that multiple flows share the same outport and thus the output channel bandwidth. *Buffer sharing* means that multiple flows share the same buffer but not channel. In *channel&buffer sharing*, multiple flows share both buffers and channels. They are scheduled as a flow called aggregate flow.

6.1.1. Channel&Buffer Sharing. Figure 4 depicts an example of flows sharing both channel and buffer in the router. As shown in the figure, we consider these flows as an aggregate flow. When an aggregate flow includes the tagged flow, it is called as *tagged*

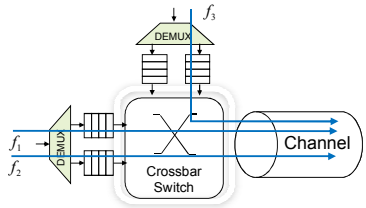


Fig. 5. An example of a channel sharing three flows.

aggregate flow. In this respect, we calculate intra-router ESC for the tagged aggregate flow in the router instead of the tagged flow. In Section 6.2, we show how ESC of the tagged flow is extracted from the ESC of the tagged aggregate flow by removing contention flows one by one. For simplicity, in the rest of the paper, "tagged flow" refers to both tagged flow and tagged aggregate flow.

6.1.2. Channel Sharing. Figure 5 depicts a channel shared between three flows f_1, f_2, and f_3. Since the arbitration policy determines how much the flows influence each other, it has to be known. We assume that, while serving multiple flows, the routers employ round robin scheduling to share the channel bandwidth. Assuming a fixed word length of L_w in all of flows, round robin arbitration means that each flow f_{s_i} in router r_j gets at least a $\frac{C}{|V_{(s_i,r_j)}|}$ of the channel bandwidth, where C is the channel capacity and $|V_{(s_i,r_j)}|$ the number of virtual channels that passing flows from them share the same channel of router r_j with flow f_{s_i}. A flow may get more if other flows use less, but we now know a worst-case lower bound on the bandwidth. Round robin arbitration has good isolation properties because the minimum bandwidth for each flow does not depend on properties of the other flows.

Since network calculus uses the abstraction of service curve to model a network element processing traffic flows [Le Boudec et al. 2004], we can also model a round robin arbiter in router r_j for flow f_{s_i} as a rate-latency server [Gebali et al. 2009] that its function is as $\beta_{(s_i,r_j)} = R_{(s_i,r_j)}(t - T^l_{(s_i,r_j)})^+$, where $R_{(s_i,r_j)}$ is the minimum service rate and $T^l_{(s_i,r_j)}$ is the maximum processing latency of the arbiter in router r_j for flow f_{s_i}. $R_{(s_i,r_j)}$ and $T^l_{(s_i,r_j)}$ are defined as follows:

$$R_{(s_i,r_j)} = \frac{C}{|V_{(s_i,r_j)}|} \tag{6}$$

$$T^l_{(s_i,r_j)} = \left(|V_{(s_i,r_j)}| - 1\right) \times \left(\frac{L_w}{C} + D_{router}\right) \tag{7}$$

where D_{router} is the delay for packet routing decision in a router.

As mentioned in Section 5, a rate-latency service curve is in fact a pseudoaffine. Therefore, $\beta_{(s_i,r_j)}$ can be expressed as $\delta_{\left(|V_{(s_i,r_j)}|-1\right)\times\left(\frac{L_w}{C}+D_{router}\right)} \otimes \gamma_{0,\frac{C}{|V_{(s_i,r_j)}|}}$. Assuming f_1 is the tagged flow in the example, $\beta_{(f_1,r)} = \delta_{2\times\left(\frac{L_w}{C}+D_{router}\right)} \otimes \gamma_{0,\frac{C}{3}}$.

6.1.3. Buffer Sharing. Figure 6 shows a buffer shared between two flows f_1 and f_2. In this type of sharing, in addition to maximum processing latency for link sharing, T^l, we introduce the head-of-Line delay for a tagged flow as below:

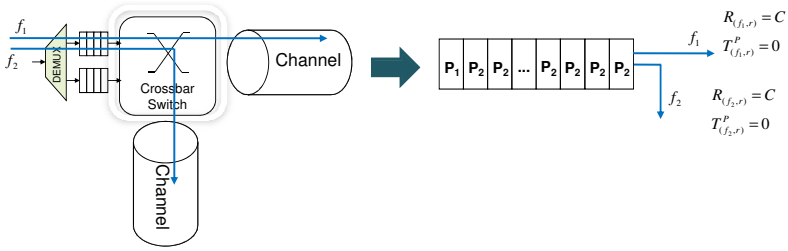


Fig. 6. An example of a buffer sharing two flows.

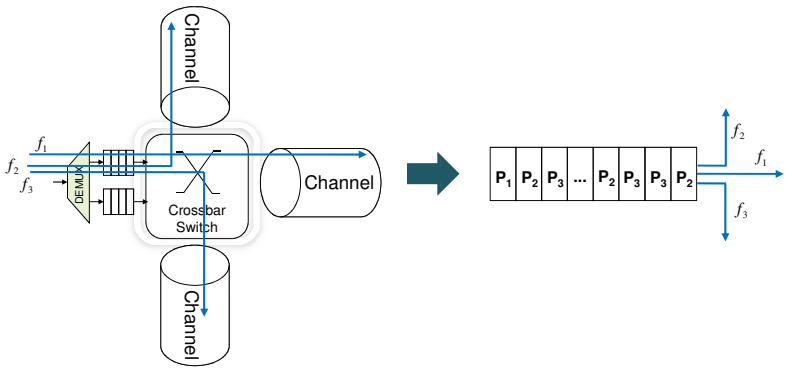


Fig. 7. An example of a buffer sharing three flows.

Head-of-Line delay (HoL): Given a flow comes at time t in a router, the maximum waiting time in the FIFO queue would be in time $t + T^{HoL}$.

Therefore, the total processing delay which comes from contention flows for tagged flow f_{s_i} in router r_j, $T^{Total}_{(s_i,r_j)}$, is equal to $T^l + T^{HoL}$

We assume f_1 in Figure 6 is the tagged flow. According to Equation (7), $T^l_{(f_1,r)} = 0$. From the figure, it is clear that $T^{HoL}_{(f_1,r)}$ is equal to the maximum delay for passing packets of flow f_2 in the buffer. According to [Le Boudec et al. 2004], the maximum delay for flow f_j is bounded by Equation (8).

$$\bar{D}_{(f_j,r)} = T^l_{(f_j,r)} + \frac{L_j + \theta_j(p_j - R_{(f_j,r)})^+}{R(f_j,r)} \tag{8}$$

Therefore, we formulate $T^{HoL}_{(f_1,r)}$ as follows:

$$T^{HoL}_{(f_1,r)} = T^l_{(f_2,r)} - \theta_2 + \frac{L_2 + \theta_2 p_2}{R_{(f_2,r)}} \tag{9}$$

If there is more than one flow sharing the buffer with the tagged flow as shown in Figure 7, HoL delay for tagged flow f_{s_i} in router r_j is given by

$$T^{HoL}_{(s_i,r_j)} = \sum_{\forall f_c \in F^B_{(s_i,r_j)}} T^{HoL(f_c)}_{(s_i,r_j)} \tag{10}$$

where $F^B_{(s_i,r_j)}$ is the set of flows which share the same buffer in router r_j with tagged flow f_{s_i}. $T^{HoL(f_c)}_{(s_i,r_j)}$ is calculated as follows.

$$T^{HoL(f_c)}_{(s_i,r_j)} = T^l_{(f_c,r)} - \theta_c + \frac{L_c + \theta_c p_c}{R_{(f_c,r)}} \tag{11}$$

Therefore router r_j can serve flow f_{s_i} by curve $\beta_{(s_i,r_j)} = \delta_{T^{Total}_{(s_i,r_j)}} \otimes \gamma_{0,R_{(s_i,r_j)}}$, where $T^{Total}_{(s_i,r_j)} = T^{HoL}_{(s_i,r_j)} + T^l_{(s_i,r_j)}$ and $R_{(s_i,r_j)}$ is calculated by Equation (6).

We analyze the buffer space threshold for each VC based on traffic specifications of flows passing through that VC, and also interference between them. The buffer space threshold for virtual channel VC_k in physical channel PC_i of router r_j is given as below:

$$B_{(VC_k,PC_i,r_j)} = \sum_{\forall f_c \in F_{(VC_k,PC_i,r_j)}} \left(\sigma_c + \rho_c T^p_{(f_c,r_j)} + \left(\theta - T^p_{(f_c,r_j)}\right)^+ \left[\left(p_c - R_{(f_c,r_j)}\right)^+ - p_c + \rho_c \right] \right) \tag{12}$$

where $F_{(VC_k,PC_i,r_j)}$ is the set of flows passing through VC_k in physical channel PC_i of router r_j.

6.2. Step2: Inter-router ESC

We have analyzed and modeled three kinds of sharing to compute the intra-router ESC. After analyzing per-router resource sharing (intra-ESC), the effects of buffer sharing and channel sharing on tagged flow have been considered and we can view an analysis model which keeps only channel&buffer sharing for tagged flow. This model is called *aggregate analysis model*. For example, suppose that a tagged flow f_1 traverses a tandem of routers, and is multiplexed with contention flows as depicted in Figure 8(a). After analyzing intra-router ESC, aggregate analysis model is shown as 8(b). In this model, $\beta_{(s_i,r_j)}$ indicates that the service curve is related to flow f_{s_i} in router r_j. For instance, $\beta_{(\{1,2\},r_3)}$ is the service curve of flow $f_{\{1,2\}}$ in router r_3. $f_{\{1,2\}}$ indicates to a flow aggregated by flows f_1 and f_2. A set of s_i's in a tandem of routers is denoted as $S = \{s_i\}$. For example, in Figure 8(b), $S = \{\{1\},\{1,2,3\},\{1,2\},\{1\}\}$.

Now, we consider aggregate analysis model to recognize interference patterns and remove contention flows one by one. A tagged flow directly contends with contention flows. Also, contention flows may contend with each other and then contend with the tagged flow again. To consider inter-ESC in the aggregate analysis model, we decompose a complex contention scenario to two basic contention patterns, namely, *Nested* and *Crossed*.Figures 8, 9, 10, and 11 illustrate examples of different kinds of nested contentions and an example of crossed contention is shown in Figure 12. In the following, we will describe these examples with more details.

We use the algebra of sets to recognize the contention scenarios. To facilitate our discussion, we define convenient notations by the example in Figure 8(b). In the example, the tandem of servers is as $\{\beta_{(\{1\},r_1)}, \beta_{(\{1,2,3\},r_2)}, \beta_{(\{1,2\},r_3)}, \beta_{(\{1\},r_4)}\}$ and $S = \{s_i\} = \{\{1\},\{1,2,3\},\{1,2\},\{1\}\}$. We define $s^m = \{s_x \big| |s_x| = max\left(|s_i|\right); \forall s_i \in S\}$, where $|s_x|$ is the cardinality (the number of elements) of set s_x. The service curve, flow, and

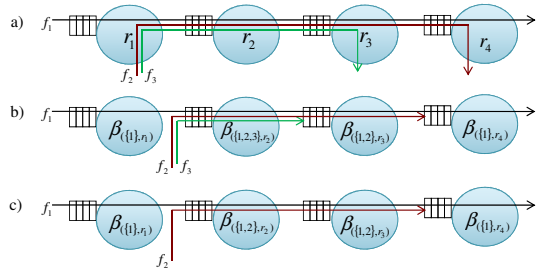


Fig. 8. Analysis for the first type of nested flows.

router related to s^m are denoted as f_{s^m}, β^m, and r^m, respectively. Thus, in Figure 8(b), $s^m = \{1, 2, 3\}$, $f_{s^m} = f_{\{1,2,3\}}$, $r^m = r_2$, and $\beta^m = \beta_{(\{1,2,3\},r_2)}$.

We denote the service curve placed before β^m on the aggregate analysis model by β^{Prev} and related aggregate flow and router as $f_{s^{Prev}}$ and r^{Prev}, respectively. Notation β^{Next} indicates to the service curve placed after β^m, as well. Therefore, due to $\beta^m = \beta_{(\{1,2,3\},r_2)}$ in Figure 8(b), $\beta^{Prev} = \beta_{(\{1\},r_1)}$, $s^{Prev} = \{1\}$, $f_{s^{Prev}} = f_{\{1\}}$, $r^{Prev} = r_1$, $\beta^{Next} = \beta_{(\{1,2\},r_3)}$, $s^{Next} = \{1, 2\}$, $f_{s^{Next}} = f_{\{1,2\}}$, $r^{Next} = r_3$.

Contention recognition procedure in an aggregate analysis model can be generalized as following steps:

(1) Find $s^m = \{s_x || s_x| = max\,(|s_i|)\,; \forall s_i \in S\}$.
(2) if $s^{Prev} \subset s^{Next}$ then the contention is *Nested*; –Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
(3) if $s^{Next} \subset s^{Prev}$ then the contention is *Nested*; –Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
(4) else
  (a) if $s^{Prev} \subset s^m$ and $s^{Next} \not\subset s^m$ then the contention is *Nested*;
    — Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
  (b) if $s^{Next} \subset s^m$ and $s^{Prev} \not\subset s^m$ then the contention is *Nested*;
    — Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
  (c) else, it is *Crossed*.
    — The problem is strictly transformed to the combination of two nested flows

To remove a contention flow from a service curve and derive the new service curve from that, we apply the proposed corollary 1 in Section 5.

When s^m is not unique, each of them can be selected. In this paper, we choose the first one from the left side in the aggregate analysis network.

In the case of $s^{Next} = s^{Prev}$, there are two possibilities:

(1) $s^{Next} = s^{Prev} \neq s^m$: Since $s^{Next} \subset s^{Prev}$ and $s^{Prev} \subset s^{Next}$, the contention is nested as previously described in contention recognition steps.
(2) $s^{Next} = s^{Prev} = s^m$: In this case, three nodes s^{Next}, s^{Prev}, and s^m should be combined as a single server by applying the theorem of *concatenation of network elements* [Le Boudec et al. 2004]. It will be discussed in Section 6.3.

In the following, we give examples for various contention patterns.

6.2.1. Nested Flows. Four different types of nested contention are exemplified as Figures 8, 9, 10, and 11. Flow f_3 is nested in flow f_2 in Figures 8, 9, and 10 and it is also nested in flow f_4 in Figure 11.

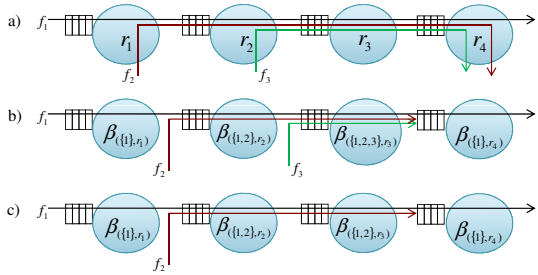


Fig. 9. Analysis for the second type of nested flows.

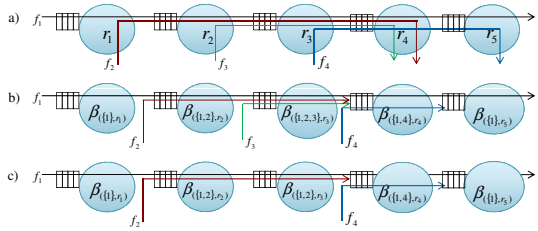


Fig. 10. Analysis for the third type of nested flows.

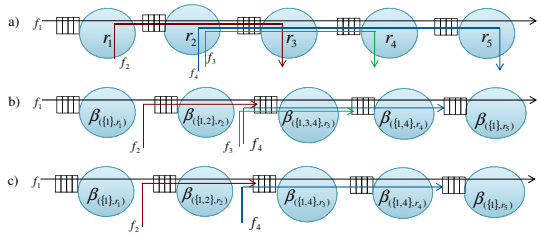


Fig. 11. Analysis for the fourth type of nested flows.

— Figure 8(b) shows the first type of nested flows after applying intra-ESC, in which $s^m = \{1,2,3\}$, $s^{Prev} = \{1\}$, and $s^{Next} = \{1,2\}$. In this case, $s^{Prev} \subset s^{Next}$ and due to step 2 of contention recognition procedure, we remove flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_{\{3\}}$ from $\beta_{(\{1,2,3\},r_2)}$ and derive $\beta_{(\{1,2\},r_2)}$, as depicted in Figure 8(c).
— The second type of nested flows in the aggregate analysis model is depicted in Figure 9. Due to Figure 9(b), $s^m = \{1,2,3\}$, $s^{Prev} = \{1,2\}$, and $s^{Next} = \{1\}$. In this case, $s^{Next} \subset s^{Prev}$ and flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_{\{3\}}$ is eliminated from $\beta_{(\{1,2,3\},r_3)}$ regarding step 3 of contention recognition procedure. Figure 9(c) shows aggregate analysis model after removing f_3.
— Figure 10 shows an example of the third type of nested contention. Based on aggregate analysis model depicted in Figure 10(b), $s^m = \{1,2,3\}$, $s^{Prev} = \{1,2\}$, and

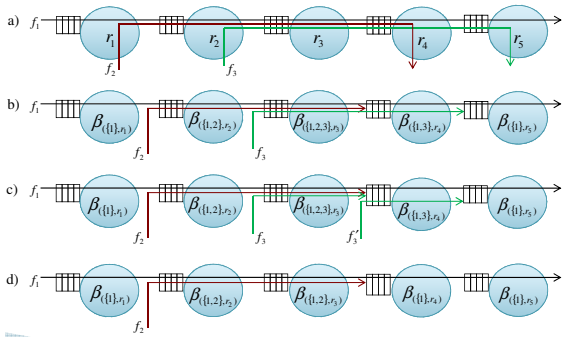


Fig. 12. Analysis for crossed Flows.

$s^{Next} = \{1, 4\}$. Since $s^{Next} \not\subset s^{Prev}$, $s^{Prev} \not\subset s^{Next}$, $s^{Prev} \subset s^m$, and $s^{Next} \not\subset s^m$, due to step 4.a) of contention recognition procedure, the case is nested contention and flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_{\{3\}}$ is removed from $\beta_{(\{1,2,3\},r_3)}$, as shown in Figure 10(c).

— Figure 11 shows a type of nested contention related to step 4.b) of contention recognition procedure. Due to Figure 11(b), $s^m = \{1, 3, 4\}$, $s^{Prev} = \{1, 2\}$, and $s^{Next} = \{1, 4\}$. Since $s^{Next} \not\subset s^{Prev}$, $s^{Prev} \not\subset s^{Next}$, $s^{Next} \subset s^m$, and $s^{Prev} \not\subset s^m$, it is a nested contention and Figure 11(c) shows that flow $f_{\{1,3,4\}-(\{1,3,4\}\cap\{1,4\})} = f_{\{3\}}$ is eliminated from $\beta_{(\{1,3,4\},r_3)}$.

6.2.2. Crossed Flows. Figure 12 shows contention flow f_2 crossed with f_3. Regarding Figure 12(b), $s^m = \{1, 2, 3\}$, $s^{Prev} = \{1, 2\}$, and $s^{Next} = \{1, 3\}$. Since s^{Prev} is not a subset of s^{Next}, and vice versa and also both of them are a subset of s^m, due to step 4.c) of contention recognition procedure, this case is a crossed contention. There are two cross points, one between r_2 and r_3 and the other between r_3 and r_4. We cut f_3 at the second cross point, i.e., at the ingress of r_4, f_3 will be split into two flows, f_3 and $\acute{f}_3$, as shown in Figure 12(c). Then the problem is strictly transformed to the combination of nested flows such that f_3 is nested in flow f_2 and $\acute{f}_3$ in f_1. It is clear that the arrival curve $\alpha_{(f_3,r_3)}$ equals to α_3 and the arrival curve $\alpha_{(\acute{f}_3,r_3)}$ equals to $\alpha^*_{(f_3,r_3)}$. To compute $\alpha^*_{(f_3,r_3)}$, we need to get the ESC of r_3 for f_3, $\beta_{(f_3,r_3)}$. Then, we calculate the output arrival curve of f_3 as $\alpha^*_{(f_3,r_3)} = \alpha_{(f_3,r_3)} \oslash \beta_{(f_3,r_3)}$ by applying the proposed Theorem 3 in Section 5. Now, nested flows f_3 and $\acute{f}_3$ can be removed from the tandem as shown in Figure 12(d).

6.3. End-to-end ESC

We show a high-level analysis flow for deriving the end-to-end ESC in Figure 13 and then present end-to-end ESC algorithm along with more details and one example.

To calculate end-to-end ESC, we first obtain intra-router ESC for the tagged flow in each router. Then we use the theorem of *concatenation of network elements* [Le Boudec et al. 2004] to model nodes sequentially connected and each is offering a service curve on the same aggregate flows $\beta_{(s_i,r_j)}$, $j = 1, 2, ..., n$ as a single server as follows:

$$\beta_{(s_i,r_{1,2,\ldots,n})} = \beta_{(s_i,r_1)} \otimes \beta_{(s_i,r_2)} \otimes ... \otimes \beta_{(s_i,r_n)}$$

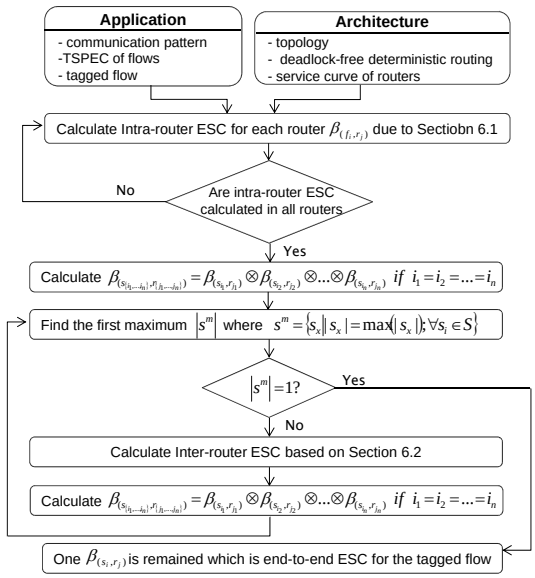


Fig. 13. End-to-end ESC analysis flow.

In the next step, we calculate inter-router ESC by applying contention recognition stages and removing contention flows as described in Section 6.2. After that, the concatenation theorem is applied again to find more equivalent servers and reduce the number of service curves. For instance, after removing contention flow f_3 in Figure 8(c), the service curve of sub-tandem $\{r_2, r_3\}$ for aggregate flow $f_{\{1,2\}}$ is computed as $\beta_{(\{1,2\},r_{2,3})} = \beta_{(\{1,2\},r_2)} \otimes \beta_{(\{1,2\},r_3)}$. If we repeat contention recognition steps, the next contention flow is f_2 nested in f_1. If we similarly remove it from $\beta_{(\{1,2\},r_{2,3})}$ and calculate convolution $\beta_{(\{1\},r_{1,2,3})} = \beta_{(\{1\},r_1)} \otimes \beta_{(\{1\},r_{2,3})}$, the end-to-end ESC of tagged flow f_1 is obtained.

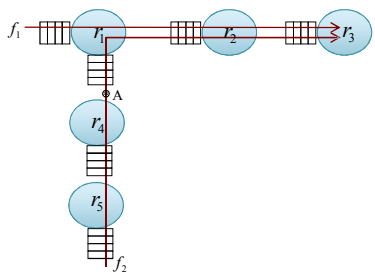


Fig. 14. The example of *joining point*.

Algorithm 1 end-to-end ESC

1: Find the set of contention flows of tagged flow f_t, denoted by CF_t
2: **for** $\forall j \in CF_t$ **do**
3: **if** $Src(j) \notin Path(t)$ **then**
4: Find $joiningnode = JoiningPoint(f_j)$
5: Calculate $X = ESC(f_j, Src(j), joiningnode)$
6: $\alpha_j = \alpha_j \oslash X$
7: **end if**
8: **end for**
9: Calculate intra-router ESC based on Section 6.1.
10: Calculate $\beta_{(s_{i_1}, r_{j_1})} \otimes \beta_{(s_{i_2}, r_{j_2})} \otimes ... \otimes \beta_{(s_{i_n}, r_{j_n})}$ *if* $i_1 = i_2 = ... = i_n$.
11: Find $s^m = \{s_x | |s_x| = max(|s_i|); \forall s_i \in S\}$.
12: **repeat**
13: **if** $s^{Prev} \subset s^{Next}$ **then**
14: Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
15: **else if** $s^{Next} \subset s^{Prev}$ **then**
16: Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
17: **else**
18: **if** $s^{Prev} \subset s^m$ and $s^{Next} \not\subset s^m$ **then**
19: Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m
20: **else if** $s^{Next} \subset s^m$ and $s^{Prev} \not\subset s^m$ **then**
21: Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m.
22: **else**
23: Find $joiningnode = JoiningPoint(f_{(s^m - s^{Prev})})$.
24: Calculate $X = ESC(f_{(s^m - s^{Prev})}, joiningnode, r^{Next})$.
25: $\acute{\alpha}_{(s^m - s^{Prev})} = \alpha_{(s^m - s^{Prev})} \oslash X$
26: Remove $f_{(s^m - s^{Prev})}$ from β^m.
27: Remove $\acute{f}_{(s^m - s^{Prev})}$ from β^{Next}.
28: **end if**
29: **end if**
30: Calculate $\beta_{(s_{i_1}, r_{j_1})} \otimes \beta_{(s_{i_2}, r_{j_2})} \otimes ... \otimes \beta_{(s_{i_n}, r_{j_n})}$ *if* $i_1 = i_2 = ... = i_n$.
31: Find s^m.
32: **until** $|s^m| \neq 1$
33: **return** end-to-end ESC for tagged flow f_t

Algorithm 1 explains the procedure of calculating end-to-end ESC with more details.

— *Joining node*: In Lines $2-8$, the algorithm checks if source node of a contention flow f_i is one of the nodes along the tagged flow's path or not. If it is not, this means that we should calculate input TSPEC of the contention flow f_i in the point joined to the tagged flow's route (point A in Figure 14 when f_1 is the tagged flow). We obtain this point by function $JoiningPoint(f_i)$ and call it *joining node*.

We give an example in Figure 15 to show how to derive an aggregate analysis model and obtain end-to-end ESC by following the proposed algorithm.

Assuming the tagged flow is f_1, line 1 of the algorithm finds CF_t which is $\{f_2, f_3, f_4\}$ in the example.

— **Loop 1 in the algorithm (Lines $2-8$):** In Lines $3-4$, the algorithm obtains *joining node* for each contention flow which its source node is not one of the nodes along the

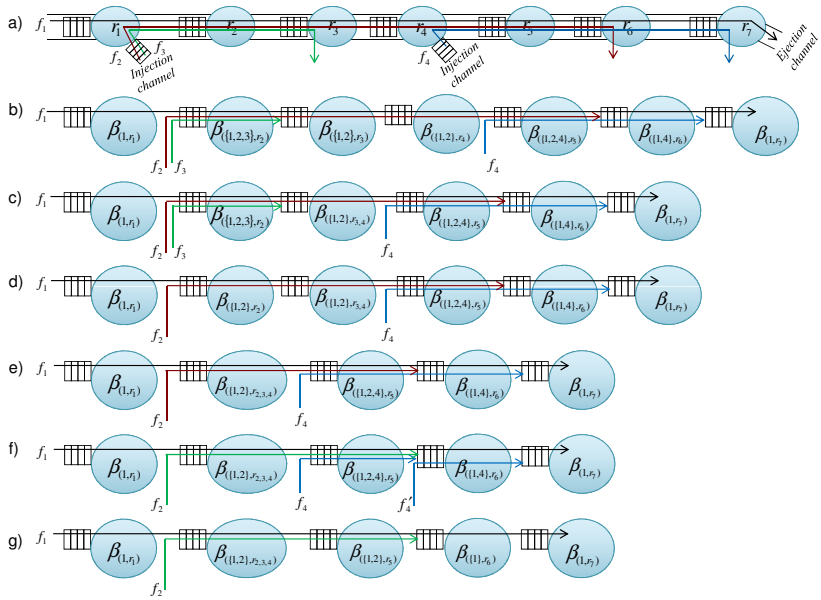


Fig. 15. An example of end-to-end ESC computation.

tandem. Then, end-to-end ESC of flow f_j from the source node to joining node has been derived by recursively calling $ESC(f_j, Src(j), joiningnode)$ in Line 5. Line 6 computes output arrival curve which is input arrival curve to the joining node and input TSPEC is extracted from that. In the example of Figure 15(a), all source nodes of contention flows are in the tagged flow's route and lines $4-6$ are skipped for them.

Line 9 obtains intra-router ESC for the tagged flow due to Section 6.1. Figure 15(b) shows the aggregate analysis model for the example. Due to line 10, $\beta_{(\{1,2\},r_{3,4})} = \beta_{(\{1,2\},r_3)} \otimes \beta_{(\{1,2\},r_4)}$. Figure 15(c) depicts the example in this step. Regarding line 11, $s^m = \{1,2,3\}$.

— **Loop 2 in the algorithm (Lines** $12-32$**):** In Lines $13-29$, we consider different contention scenarios along the route using the algebra of sets. In this step, we intend to remove contention flows one by one due to their effects on the tagged flow as mentioned in Section 6.2. Lines $13-21$ consider nested contentions and lines $22-28$ crossed one.

— **Nested contention in the example:** From Figure 15(c), $s^m = \{1,2,3\}$, $s^{Prev} = \{1\}$, and $s^{Next} = \{1,2\}$. Since $s^{Prev} \subset s^{Next}$, due to line 13, flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_3$ is removed from $\beta_{(\{1,2,3\},r_2)}$ as shown in Figure 15(d).

Lines $30-31$ are the same as lines $10-11$ which calculate concatenation of the nodes on the same aggregate flows and then obtain new s^m, which result in $\beta_{(\{1,2\},r_{2,3,4})} = \beta_{(\{1,2\},r_2)} \otimes \beta_{(\{1,2\},r_{3,4})}$, and $s^m = \{1,2,4\}$ (Figure 15(e)).

— **Crossed contention in the example:** If we repeat contention recognition steps in Loop 2, the next contention in the example is crossed. From Figure 15(e), $s^m = \{1, 2, 4\}$, $s^{Prev} = \{1, 2\}$, and $s^{Next} = \{1, 4\}$. Since neither $s^{Prev} \subset s^{Next}$ nor $s^{Next} \subset s^{Prev}$ and also either $s^{Next} \subset s^m$ and $s^{Prev} \subset s^m$, it goes to the else part (lines $22-28$) of the algorithm. As shown in Figure 15(e), contention flow f_2 is crossed with f_4. There are two cross points, one between $r_{2,3,4}$ and r_5 and the other between r_5 and r_6. Regarding the algorithm, we cut f_4 at the second cross point, i.e., at the ingress of r_6, f_4 will be split into two flows, f_4 and $\acute{f}_4$, as shown in Figure 15(f). Then, the problem is transformed to the combination of two nested scenarios. Apparently the arrival curve $\alpha_{\acute{f}_4}$ of $\acute{f}_4$ is equal to $\alpha^*_{f_4}$ of f_4. To compute $\alpha^*_{f_4}$, we need to get the ESC of f_4 from r_5 to r_6, which is derived regarding lines 23 and 24. Then, line 25 calculates output arrival curve $\alpha^*_{f_4}$ ($\alpha_{\acute{f}_4}$) by applying the proposed Theorem 3 in Section 5. Then, f_4 and $\acute{f}_4$ are removed from r_5 and r_6 due to lines 26 and 27, respectively, as shown in Figure 15(g).

Therefore, according to lines $30-31$, $\beta_{(\{1,2\},r_{2,3,4,5})} = \beta_{(\{1,2\},r_{2,3,4})} \otimes \beta_{(\{1,2\},r_5)}$, $\beta_{(\{1\},r_{6,7})} = \beta_{(\{1\},r_6)} \otimes \beta_{(\{1\},r_7)}$, and $s^m = \{1, 2\}$. We similarly repeat contention recognition and convolution steps until $|s^m| \neq 1$. When $|s^m| = 1$, the end-to-end ESC of tagged flow f_1 is obtained.

6.4. LUDB Derivation

To compute the delay bound for a flow passing a series of nodes, one simple way is to calculate the summation of delay bounds at each node. However, this results in a loose total delay bound. To tighten the worst-case delay bound along the network, the end-to-end service curve of the flow is used as stated in corollary *Pay Bursts Only Once* [Le Boudec et al. 2004]. Hence, we first calculate the end-to-end ESC of the tagged flow based on the proposed algorithm and then obtain LUDB according to Theorem 1. We have implemented algorithms employed in our methodology.

7. EXPERIMENTS

7.1. Experimental Setup

To evaluate the capability of our method, we applied it to a synthetic traffic pattern and a realistic one. Throughout the experiments, we assume an SoC with 500 MHz frequency in which packets traverse the network using the XY routing algorithm. Flows follow TSPEC, $f_i \propto (L_i, p_i, \sigma_i, \rho_i)$, and each node guarantees the service curve of $\beta_{R,T}(t) = \delta_T \otimes \gamma_{0,R}$, where the serving rate R is C *flit/cycle* and the latency T, $\frac{L_w}{C} + D_{router}$ *cycle*. We have implemented the proposed analytical model in C++ to automate analysis steps.

7.2. Synthetic Traffic Pattern

We synthesize a simple traffic pattern as shown in Figure 16 to follow the analytical approach step by step and derive numerical results. The figure depicts a network with 4 flows and 4 routers which serve flows in the FIFO order. f_1 is the tagged flow and f_2 and f_4 are contention flows.

7.2.1. *Computation of the end-to-end equivalent service curve.*

Step 1: We first calculate the intra-router ESC for the tagged flow in each node. Then, we can model a flow passing through a series of routers as a series of concatenated pseudoaffine servers.

It is worth mentioning that TSPEC of each flow f_j mentioned above is the TSPEC of the input flow to its source node, for example $f_2 \propto (L_2, p_2, \sigma_2, \rho_2)$ which means $\rho_{(f_2,r_1)} = \rho_2$ and other characteristics can be obtained as well.

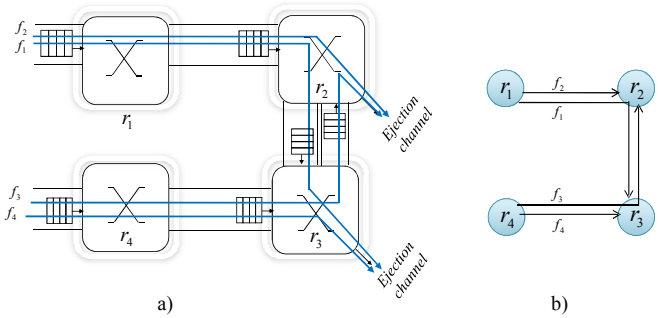


Fig. 16. A synthetic example.

—***In router*** r_1**:** From Equation (6) and (7), the ESC for aggregate flow $f_{\{1,2\}}$ in node 1 is given by:

$$\beta_{(f_{\{1,2\}},r_1)} = \delta_0 \otimes \gamma_{0,C}. \tag{13}$$

—***In router*** r_2**:** $F^B_{(f_1,r_2)} = \{f_2\}$ and due to Equation (6) and (7), $R_{(f_1,r_2)} = C$ and $T^l_{(f_1,r_2)} = 0$. Furthermore, $T^{Total}_{(f_1,r_2)} = T^l_{(f_1,r_2)} + T^{HoL}_{(f_1,r_2)}$ and regarding to Equation (10) and (11), $T^{HoL}_{(f_1,r_2)} = max_{\forall f_c \in F^B_{(f_1,r_2)}} \left(T^{HoL(f_c)}_{(f_1,r_2)}\right) = T^{HoL(f_2)}_{(f_1,r_2)}$ where $T^{HoL(f_2)}_{(f_1,r_2)}$ is calculated as follows:

$$T^{HoL(f_2)}_{(f_1,r_2)} = T^l_{(f_2,r_2)} - \theta_{(f_2,r_2)} + \frac{L_{(f_2,r_2)} + \theta_{(f_2,r_2)} p_{(f_2,r_2)}}{R_{(f_2,r_2)}} \tag{14}$$

where $R_{(f_2,r_2)} = \frac{C}{2}$, $T^l_{(f_2,r_2)} = \frac{L_w}{C} + D_{router}$, because two VCs (one transmits f_2 and the other f_3) are sharing the ejection channel of router r_2. In Equation (14), we should obtain TSPEC of input flow f_2 to r_2 which is TSPEC of output flow f_2 from r_1. Since TSPEC is derived from arrival curve, we obtain arrival curve of output flow f_2 from r_1 by applying the proposed Theorem 3 in Section 5. We assumed $\theta_i \leq T_{(f_i,r_j)}$ *for* $\forall f_i$ ***passing through*** $\forall r_j$. Thus, $\alpha^*_{(f_2,r_1)} = \alpha_{(f_2,r_2)} = \gamma_{\sigma_{(f_2,r_1)}+\rho_{(f_2,r_1)}T_{(f_2,r_1)},\rho_{(f_2,r_1)}}$ where $\rho_{(f_2,r_1)} = \rho_2$ and $\sigma_{(f_2,r_1)} = \sigma_2$. In this respect, we can say $\alpha_{(f_2,r_2)} = \gamma_{\sigma_2+\rho_2 T_{(f_2,r_1)},\rho_2}$. For deriving $T_{(f_2,r_1)}$, we should first obtain ESC for flow f_2 in router r_1, $\beta_{(f_2,r_1)}$, as follows.
From Equation (13), $\beta_{(f_{\{1,2\}},r_1)} = \delta_0 \otimes \gamma_{0,C}$. We then remove f_1 from aggregate flow $f_{\{1,2\}}$ according to Corollary 1 in Section 5, $\beta_{(f_2,r_1)}$ is given by:

$$\beta_{(f_2,r_1)} = \delta_{\left[\frac{L_1+\theta_1(p_1-C)^+}{C}\right]^+ + \theta_1} \otimes \gamma_{0,C-\rho_1} = \delta_{\frac{L_1+\theta_1 p_1}{C}} \otimes \gamma_{0,C-\rho_1} \tag{15}$$

In this respect $T_{(f_2,r_1)} = \frac{L_1+\theta_1 p_1}{C}$, and $\alpha_{(f_2,r_2)} = \gamma_{\sigma_2+\frac{\rho_2(L_1+\theta_1 p_1)}{C},\rho_2}$ which means $\sigma_{(f_2,r_2)} = \sigma_2 + \frac{\rho_2(L_1+\theta_1 p_1)}{C}$, $\rho_{(f_2,r_2)} = \rho_2$, $L_{(f_2,r_2)} = L_{(f_2,r_1)} = L_2$, and $p_{(f_2,r_2)} = p_{(f_2,r_1)} = p_2$. Therefore, Equation (14) is rewritten as below:

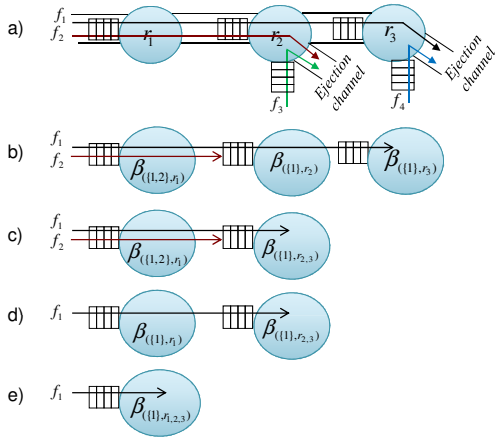


Fig. 17. Analysis steps for the example in Figure 15.

$$T^{HoL(f_2)}_{(f_1,r_2)} = \frac{L_w}{C} + D_{router} - \theta_{(f_2,r_2)} + \frac{L_2 + \theta_{(f_2,r_2)} p_2}{\frac{C}{2}} \tag{16}$$

where $\theta_{(f_2,r_2)} = \frac{\sigma_{(f_2,r_2)} - L_{(f_2,r_2)}}{\rho_{(f_2,r_2)} - \rho_{(f_2,r_2)}} = \frac{\sigma_2 C + \rho_2 L_1 + \rho_2 \theta_1 p_1 - L_2 C}{C(p_2 - \rho_2)}$.
As mentioned before, $R_{(f_1,r_2)} = C$, $T^l_{(f_1,r_2)} = 0$, and $T^{Total}_{(f_1,r_2)} = T^l_{(f_1,r_2)} + T^{HoL}_{(f_1,r_2)}$. Therefore, the ESC for tagged flow f_1 in router 2 is given by:

$$\beta_{(f_{\{1\}},r_2)} = \delta_{0+T^{HoL(f_2)}_{(f_1,r_2)}} \otimes \gamma_{0,C}. \tag{17}$$

— ***In router*** r_3**:** Since VC of f_1 is sharing the ejection channel of r_3 with VC of f_4, due to Equation (6) and (7), $R_{(f_1,r_3)} = \frac{C}{2}$ and $T^l_{(f_1,r_3)} = \frac{L_w}{C} + D_{router}$. Thus, the ESC for tagged flow f_1 in router 3 is given by:

$$\beta_{(f_{\{1\}},r_3)} = \delta_{\left(\frac{L_w}{C} + D_{router}\right)} \otimes \gamma_{0,\frac{C}{2}}. \tag{18}$$

Step 2: Now, we are able to compute per-flow ESC provided by the tandem of routers the tagged flow passes. Figure 17 depicts different steps of computing end-to-end ESC for tagged flow f_1. After calculating intra-router ESC as mentioned in Step 1, we have an *aggregate analysis model* as shown in Figure 17(b). Since we have investigated the effect of flow f_2 on tagged flow f_1 in router r_2, when we calculated $\beta_{(f_1,r_2)}$ in step 1, f_2 is removed from r_2 in Figure 17(b). Similarly, f_3 and f_4 are eliminated from r_2 and r_3, respectively. We then obtain end-to-end ESC for tagged flow f_1 by following Algorithm 1. Due to the algorithm, $\beta_{(\{1\},r_{2,3})}$ in Figure 17(c) is calculated as $\beta_{(\{1\},r_2)} \otimes \beta_{(\{1\},r_3)}$.

We use the theorem of *Concatenation of network elements* [Le Boudec et al. 2004]. Given are two nodes sequentially connected and each is offering a latency-rate service

curve $\beta_{R_i,T_i},\ i = 1 and 2$. These nodes can be represented as a single latency-rate server as follows:

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2} \tag{19}$$

Therefore, $\beta_{(\{1\},r_{2,3})}$ is given by:

$$\beta_{(\{1\},r_{2,3})} = \delta_{\frac{L_w}{C}+D_{router}+T^{HoL(f_2)}_{(f_1,r_2)}} \otimes \gamma_{0,\frac{C}{2}}. \tag{20}$$

In Figure 17(c), $s^m = \{1,2\}$, $s^{Prev} = \{\}$, and $s^{Next} = \{1\}$. The algorithm then removes flow f_2 from aggregate flow $f_{\{1,2\}}$ in router r_1. To this end, we apply the proposed corollary 1 to obtain ESC $\beta_{(\{1\},r_1)}$ by subtracting arrival curve of α_2 from $\beta_{(\{1,2\},r_1)}$, as follows:

$$\beta_{(\{1\},r_1)} = \delta_{\frac{L_2+\theta_2(p_2-C)^+}{C}+\theta_2} \otimes \gamma_{0,C-\rho_2} \tag{21}$$

Figure 17(c) depicts the example after removing arrival curve of flow f_2 from $\beta_{(\{1,2\},r_1)}$. Now, end-to-end ESC can be calculated by:

$$\begin{aligned}\beta_{(\{1\},r_{1,2,3})} &= \beta^{eq}_{f_1} = \beta_{(\{1\},r_1)} \otimes \beta_{(\{1\},r_{2,3})} \\ &= \delta_{\frac{L_w+L_2+\theta_2(p_2-C)^+}{C}+D_{router}+\theta_2+T^{HoL(f_2)}_{(f_1,r_2)}} \otimes \gamma_{0,min\left(\frac{C}{2},C-\rho_2\right)}\end{aligned} \tag{22}$$

Suppose that flows follow TSPEC, $f_1 \propto (1,1,8,0.128)$, $f_2 \propto (1,1,2,0.032)$, $f_3 \propto (1,1,2,0.008)$, and $f_4 \propto (1,1,4,0.128)$. Therefore, θ_j is computed for each flow f_j as $\theta_1 = (\sigma_1 - L_1)/(p_1 - \rho_1) = (8-1)/(1-0.128) = 8.027$, $\theta_2 = 1.033$, $\theta_3 = 1.008$, and $\theta_4 = 3.44$. Also, we assume serving rate $C = 1$ *flit/cycle*, $L_w = 1$ *flit*, and $D_{router} = 1$ *cycle*. We then replace the variables in Equation (22) by numbers as follows:

$$\beta^{eq}_{f_1} = \delta_{9.363} \otimes \gamma_{0,0.5} \tag{23}$$

7.2.2. ***Computation of LUDB***.

According to Theorem 1 and Equation (22), the maximum delay for flow f_1 is bounded by

$$h(\alpha_1, \beta^{eq}_{f_1}) = \left\lceil \frac{L_w + L_2 + \theta_2(p_2 - C)^+}{C} + D_{router} + \theta_2 + T^{HoL(f_2)}_{(f_1,r_2)} + \frac{L_1 + \theta_1\left(p_1 - min\left(\frac{C}{2}, C-\rho_2\right)\right)^+}{min\left(\frac{C}{2}, C-\rho_2\right)} \right\rceil \tag{24}$$

If we substitute the values into variables in the above mentioned equation, $h(\alpha_1, \beta^{eq}_{f_1})$ is equal to $\lceil 19.39 \rceil = 20$.

In what follows, we consider the accuracy of our proposed analytical method through the BookSim simulator [Jiang et al. 2013] and then compare it with the methods without considering the traffic peak rate behavior [Lenzini et al. 2006].

7.2.3. ***Computation of Buffer Size Thresholds***.

As routers are assumed to be input-buffered, we derive buffer size threshold for each input channel in each router by following Eq. (12). In the example of Figure 16, we have assumed one VC per each PC. Therefore, buffer size thresholds are calculated and presented as Table II.

Table II. Buffer size thresholds in the case study with synthetic traffic pattern

	Injection Channel	*Western Channel*	*Northern Channel*	*Eastern Channel*	*Southern Channel*
Router 1	6	–	–	–	–
Router 2	–	11	–	–	3
Router 3	–	8	8	–	–
Router 4	6	–	–	–	–

The buffers size thresholds marked by "-" are not used by flows and thus not relevant for the threshold calculation.

The value of buffer size thresholds per channel depends on the traffic load on that channel which is affected by the number of flows passing through the channel, their traffic specifications, and the contention between them.

7.2.4. ***Simulation Result***.

We investigate the accuracy of the proposed analytical model through BookSim simulator which is a cycle-accurate simulator [Jiang et al. 2013]. The simulation uses the same assumptions as the analytical model. We have considered a 2×2 mesh on-chip interconnect as shown in Figure 16 and input-buffered routers with 12 *flits* in each input channel. It takes 1 clock cycle to pass a flit within a router and 1 clock cycle to transmit a flit over wires between neighboring routers. We also consider the XY routing algorithm to route the data packets among cores.

Simulation result shows that worst-case delay for tagged flow f_1 in the previously mentioned system is equal to 19 *cycles*, which is below the LUBD of 20 *cycles*, predicted by our model.

We also change the value of σ_2 from 2 to 4 to consider more experiments. The LUDB calculated by our analytical model for tagged flow f_1 is equal to 24 *cycles* and the result from the simulation is also 24 *cycles*, again below the analytical LUDB.

7.2.5. ***Comparison***.

If we use (σ, ρ) instead of TSPEC, each flow j would be constrained by arrival curve $\alpha_j = \sigma_j + \rho_j t = \gamma_{\sigma_j,\rho_j}$. Therefore, flows in the example are represented as $f_1 \propto (8, 0.128)$, $f_2 \propto (2, 0.032)$, $f_3 \propto (2, 0.008)$, and $f_4 \propto (4, 0.128)$. We then follow the stages of computing individual delay bounds for a tagged flow as stated before. For this purpose, we can easily revise our proposed theorems for (σ, ρ) flows by substituting σ and ρ into L and p, respectively, in all formulas. We can also apply the method presented in [Lenzini et al. 2006]. With both approaches, the same value for $h(\alpha_1, \beta_{f_1}^{eq})$ is achieved and equals to 26. Thus, our proposed method which calculates $\bar{D}_{VBR}$ has 23% improvement on the accuracy of the delay bound than the method with CBR flows ($\bar{D}_{CBR}$).

To analyze delay sensitivity, Table III depicts LUDB for tagged flow f_1 in a network with CBR flows ($\bar{D}_{CBR}$) and also VBR flows ($\bar{D}_{VBR}$) versus the different values of ser-

Table III. End-to-end delay comparison for tagged flow f_1 under different service rates

	$R_1 = 1$	$R_2 = 0.7$	$R_3 = 0.5$
$R_{f_1}^{eq}$	0.5	0.35	0.25
$T_{f_1,CBR}^{eq}$	10	13.428	18
$T_{f_1,VBR}^{Eq}$	9.363	13.326	18.951
$D_{f_1,CBR}$	26	37	50
$D_{f_1,VBR}$	20	32	48
η_{f_1}	23%	13.5%	4%

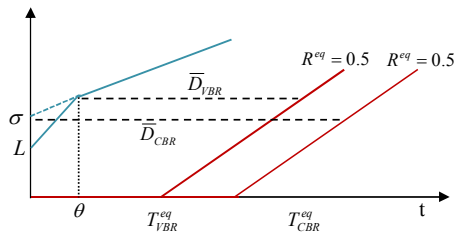


Fig. 18. Comparing $\bar{D}_{VBR}$ and $\bar{D}_{CBR}$ with the same equivalent service curve.

vice rate R, along with values for the end-to-end ESC parameterized by $R^{eq}_{f_1}$, $T^{eq}_{f_1,CBR}$ and $T^{eq}_{f_1,VBR}$. From this table, it is clear that the end-to-end equivalent service rate, $R^{eq}_{f_1}$, is decreasing by reducing R, while the end-to-end processing delays and delay bounds are increasing as well. Also, it is worth mentioning that the improvement percentage (η) decreases because of reduction of $R^{eq}_{f_1}$ and increase of $T^{eq}_{f_1,CBR}$ and $T^{eq}_{f_1,VBR}$. This is due to the relation between these parameters which we will elaborate it in the following.

Figure 18 shows $\bar{D}_{CBR}$ and $\bar{D}_{VBR}$ for $R^{eq} = 0.5$ where $p \geq R^{eq}$ and the end-to-end ESCs are in the form of $\delta_{T^{eq}} \otimes \gamma_{0,R^{eq}}$. According to [Le Boudec et al. 2004], $\bar{D}_{CBR} = T^{eq}_{CBR} + \frac{\sigma}{R^{eq}}$ and with Theorem 1, $\bar{D}_{VBR} = T^{eq}_{VBR} + \frac{L+\theta(p-R^{eq})}{R^{eq}}$. η is calculated as follows.

$$\eta = \frac{\bar{D}_{CBR} - \bar{D}_{VBR}}{\bar{D}_{CBR}} = \frac{\sigma - L - p\theta + R^{eq}\left(T^{eq}_{CBR} - T^{eq}_{VBR} + \theta\right)}{\sigma + T^{eq}_{CBR}R^{eq}} \tag{25}$$

To analyze the behavior of η, we compute the derivative of function η in terms of R^{eq} as follows:

$$\frac{d\eta}{dR^{eq}} = \frac{T^{eq}_{CBR}\left(L + p\theta\right) - \sigma\left(T^{eq}_{VBR} - \theta\right)}{\left(\sigma + T^{eq}_{CBR}R^{eq}\right)^2}$$

From Figure 18, it is obvious that $L + p\theta \geq \sigma$ and $T^{eq}_{CBR} \geq T^{eq}_{VBR} - \theta$ which results $T^{eq}_{CBR}\left(L + p\theta\right) - \sigma\left(T^{eq}_{VBR} - \theta\right) \geq 0$. Thus, $\frac{d\eta}{dR^{eq}} \geq 0$ and η is an increasing function in terms of R^{eq} which means that when R^{eq} increases or decreases, η shows the same behavior as R^{eq}.

Table IV shows $R^{eq}_{f_1}$, $T^{eq}_{f_1,CBR}$ and $T^{eq}_{f_1,VBR}$, $\bar{D}_{CBR}$, and $\bar{D}_{VBR}$ for tagged flow f_1 versus the different values of processing delay T. From this table, it can be seen that the end-to-end processing delays and delay bounds are decreasing by reducing T.

Table IV. End-to-end delay comparison for tagged flow f_1 under different processing delay

	$T_1 = 10$	$T_2 = 2$	$T_3 = 1$	$T_4 = 0.5$	$T_5 = 0.1$
$R^{eq}_{f_1}$	0.5	0.5	0.5	0.5	0.5
$T^{eq}_{f_1,CBR}$	26	10	8	7	6.2
$T^{eq}_{f_1,VBR}$	25.363	9.363	7.363	6.363	5.563
$D_{f_1,CBR}$	42	26	24	23	23
$D_{f_1,VBR}$	39	20	18	17	16
η_{f_1}	7.1%	23%	25%	26.1%	30.4%

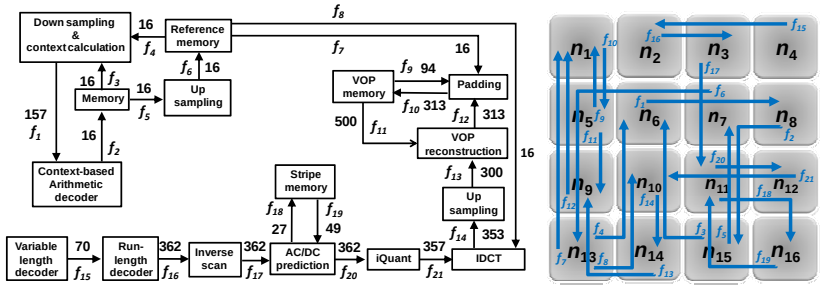


Fig. 19. VOPD Application

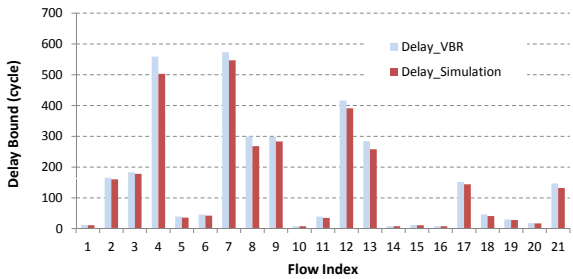


Fig. 20. Comparison of delay bounds from the proposed model and simulator for VOPD application

7.3. Realistic Traffic Pattern

We consider a real-time multimedia application with a random mapping to the tiles of a 4×4 mesh on-chip network. Figure 19 shows the task graph and flow mapping of a Video Object Plane Decoder (VOPD) [Bertozzi et al. 2002] in which each block corresponds to an IP and the numbers near the edges represent the bandwidth (in MBytes/sec) of the data transfer, for a 30 frames/sec MPEG-4 movie with 1920×1088 resolution [Van der Tol et al. 2002]. There are 21 communication flows which characterized by TSPEC. We assume L_i and p_i for all flows are the same and equal to $1\ flit$ and $1\ flit/cycle$, respectively. ρ_i is determined in $flits/cycle$ due to associated bandwidth with flow f_i in Figure 19 and also, σ_i varies between 8 and 128 $flits$ for different flows.

We derive delay bounds from the proposed analytical model, $\bar{D}_{f_i,VBR}$, and BookSim simulator, $\bar{D}_{f_i,Sim}$ for the whole set of flows in Figure 20. In order to have a better insight about the proposed model, for each obtained delay bound, the relative error with respect to simulation result is calculated. The calculations show that the maximum and average relative errors are about 12.1% and 6.8%, respectively, which confirm the accuracy of the proposed model.

which is below the LUBD of 20 cycles, predicted by our model.

As can be observed from Figure 20, a flows may have larger (like f_7) or smaller (like f_{14}) worst-case delay bound than the other flows, which depends on its traffic

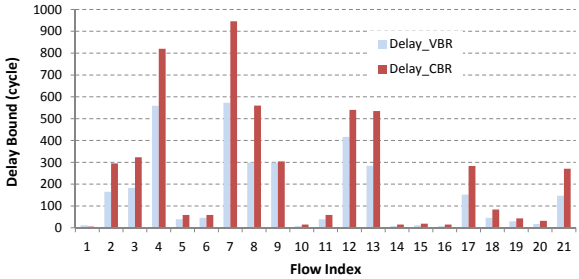


Fig. 21. Comparing $\bar{D}_{VBR}$ and $\bar{D}_{CBR}$ for VOPD application

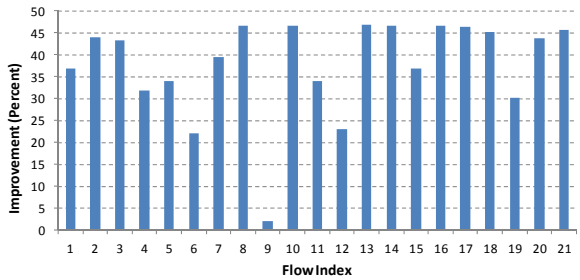


Fig. 22. Improvement percentage of $\bar{D}_{VBR}$ than $\bar{D}_{CBR}$ for VOPD application

specification (TSPEC) and the situation of that flow in the network. For example, if the worst-case delay bound of a particular flow is too large, 1) the flow is probably more limited by its TSPEC parameters for injecting to the network, 2) the flow may have a longer path from its source to destination, or 3) the flow may have more contentions (both direct and indirect) with other flows along its path.

Figure 21 compares the results of applying our analytical model, $\bar{D}_{f_i,VBR}$, and the method with CBR flows, $\bar{D}_{f_i,CBR}$. As you can see in this figure, the proposed model in this paper is more accurate than the method without considering the traffic peak behavior. Figure 22 presents improvement percentage for each flow f_i, η_{f_i}, as defined in Eq. 25 to show the effectiveness of our model. Compared to previous models with two parameters, the proposed method improves the accuracy of the delay bounds up to 46.9% and more than 37% on average over all flows.

Table V presents buffer size threshold of input channels used by flows due to Eq. 12.

Table V. Buffer size thresholds for VOPD application

$B_{(r_1,I)}=1$	$B_{(r_5,I)}=17$	$B_{(r_7,W)}=1$	$B_{(r_9,S)}=259$	$B_{(r_{12},I)}=1$	$B_{(r_{14},N)}=1$
$B_{(r_1,S)}=275$	$B_{(r_5,N)}=1$	$B_{(r_7,N)}=68$	$B_{(r_{10},I)}=1$	$B_{(r_{12},W)}=4$	$B_{(r_{14},E)}=7$
$B_{(r_2,I)}=1$	$B_{(r_5,E)}=7$	$B_{(r_7,E)}=7$	$B_{(r_{10},E)}=68$	$B_{(r_{13},I)}=204$	$B_{(r_{15},I)}=16$
$B_{(r_2,E)}=1$	$B_{(r_5,S)}=262$	$B_{(r_7,S)}=1$	$B_{(r_{10},S)}=84$	$B_{(r_{13},N)}=1$	$B_{(r_{15},N)}=1$
$B_{(r_3,I)}=1$	$B_{(r_6,I)}=1$	$B_{(r_8,I)}=1$	$B_{(r_{11},I)}=17$	$B_{(r_{13},E)}=68$	$B_{(r_{15},E)}=7$
$B_{(r_3,W)}=1$	$B_{(r_6,E)}=1$	$B_{(r_8,W)}=1$	$B_{(r_{11},N)}=77$	$B_{(r_{14},I)}=2$	$B_{(r_{16},I)}=1$
$B_{(r_3,E)}=1$	$B_{(r_6,S)}=17$	$B_{(r_9,I)}=68$	$B_{(r_{11},E)}=1$	$B_{(r_{14},W)}=84$	$B_{(r_{16},N)}=1$
$B_{(r_4,I)}=1$	$B_{(r_7,I)}=1$	$B_{(r_9,N)}=16$	$B_{(r_{11},S)}=16$		

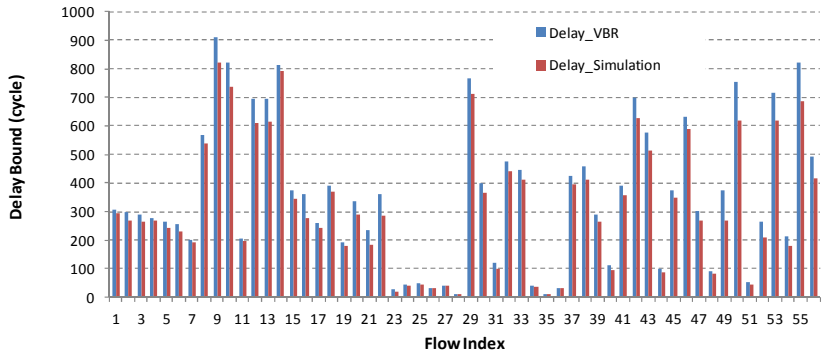


Fig. 23. Comparison of delay bounds from the proposed model and simulator under the transpose traffic pattern

Sub-index (r, L) in Table V refers to input channel L of router r, where r is the number of the router and L is a letter assigned to the input port which is defined as I: *Injection channel*, W: *Western input channel*, N: *Northern input channel*, E: *Eastern input channel*, and S: *Southern input channel*. For example, $B_{(r_3,W)}$ indicates the buffer size threshold in western input channel of router 3.

7.4. Transpose Traffic Pattern

To investigate a larger network, we experiment a 8×8 mesh network under the transpose traffic pattern with 56 communication flows characterized by TSPEC. In this traffic pattern, the node with binary value $a_{n-1}, a_{n-2}, ..., a_1, a_0$ communicates with the node $\bar{a}_{n/2-1}, ..., \bar{a}_0, \bar{a}_{n-1}, ..., \bar{a}_{n/2}$. For all traffic flows, we assume the same values for L_i and p_i which are $1\ flit$ and $1\ flit/cycle$, respectively. For different flows, ρ_i varies between 0.001 and $0.03\ flits/cycle$, and σ_i between 2 and $128\ flits$. Table VI presents the source and destination of flows along with the index assigned to them.

Similar to previous case studies, delay bounds from the proposed analytical model, $\bar{D}_{f_i,VBR}$, and BookSim simulator, $\bar{D}_{f_i,Sim}$ are derived for all flows and presented as Figure 23. As can be seen from this figure, all delays observed in simulations are below the LUDB but not too far, suggesting that the analytical bound is fairly tight since the simulation typically does not exercise the worst case.

Table VI. The list of flows

$f_1 : 0 \longrightarrow 63$	$f_{11} : 20 \longrightarrow 29$	$f_{21} : 25 \longrightarrow 52$	$f_{30} : 55 \longrightarrow 1$	$f_{39} : 29 \longrightarrow 20$	$f_{48} : 44 \longrightarrow 26$
$f_2 : 1 \longrightarrow 55$	$f_{12} : 10 \longrightarrow 46$	$f_{22} : 24 \longrightarrow 60$	$f_{31} : 47 \longrightarrow 2$	$f_{40} : 46 \longrightarrow 10$	$f_{49} : 52 \longrightarrow 25$
$f_3 : 2 \longrightarrow 47$	$f_{13} : 9 \longrightarrow 54$	$f_{23} : 34 \longrightarrow 43$	$f_{32} : 39 \longrightarrow 3$	$f_{41} : 54 \longrightarrow 9$	$f_{50} : 60 \longrightarrow 24$
$f_4 : 3 \longrightarrow 39$	$f_{14} : 8 \longrightarrow 62$	$f_{24} : 33 \longrightarrow 51$	$f_{33} : 31 \longrightarrow 4$	$f_{42} : 62 \longrightarrow 8$	$f_{51} : 43 \longrightarrow 34$
$f_5 : 4 \longrightarrow 31$	$f_{15} : 19 \longrightarrow 37$	$f_{25} : 32 \longrightarrow 59$	$f_{34} : 23 \longrightarrow 5$	$f_{43} : 37 \longrightarrow 19$	$f_{52} : 51 \longrightarrow 33$
$f_6 : 5 \longrightarrow 23$	$f_{16} : 18 \longrightarrow 45$	$f_{26} : 41 \longrightarrow 50$	$f_{35} : 15 \longrightarrow 6$	$f_{44} : 45 \longrightarrow 18$	$f_{53} : 59 \longrightarrow 32$
$f_7 : 6 \longrightarrow 15$	$f_{17} : 17 \longrightarrow 53$	$f_{27} : 40 \longrightarrow 58$	$f_{36} : 22 \longrightarrow 13$	$f_{45} : 53 \longrightarrow 17$	$f_{54} : 50 \longrightarrow 41$
$f_8 : 13 \longrightarrow 22$	$f_{18} : 16 \longrightarrow 61$	$f_{28} : 48 \longrightarrow 57$	$f_{37} : 30 \longrightarrow 12$	$f_{46} : 61 \longrightarrow 16$	$f_{55} : 58 \longrightarrow 40$
$f_9 : 12 \longrightarrow 30$	$f_{19} : 27 \longrightarrow 36$	$f_{29} : 63 \longrightarrow 0$	$f_{38} : 38 \longrightarrow 11$	$f_{47} : 36 \longrightarrow 27$	$f_{56} : 57 \longrightarrow 48$
$f_{10} : 11 \longrightarrow 38$	$f_{20} : 26 \longrightarrow 44$				

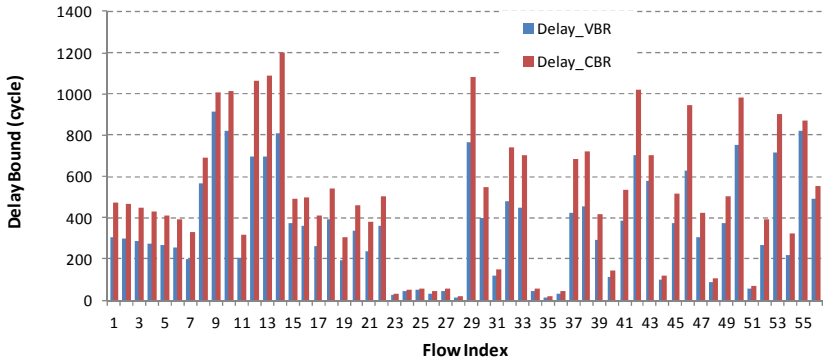


Fig. 24. Comparing $\bar{D}_{VBR}$ and $\bar{D}_{CBR}$ under the transpose traffic pattern

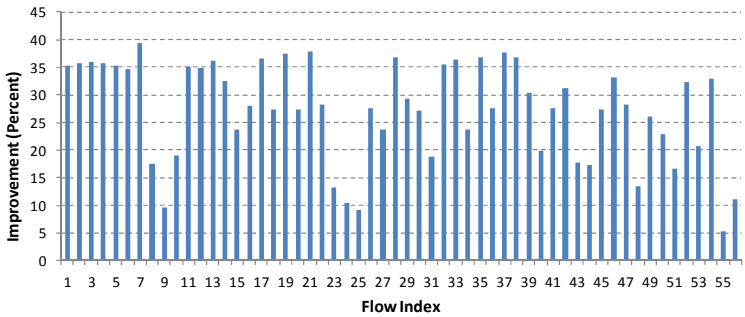


Fig. 25. Improvement percentage of $\bar{D}_{VBR}$ than $\bar{D}_{CBR}$ under the transpose traffic pattern

To consider the accuracy of the analytical model, the relative errors with respect to simulation results are computed. The calculations show that the maximum and average relative errors are about 33.3% and 13%, respectively.

We also calculate per-flow delay bounds from our proposed method, $\bar{D}_{f_i,VBR}$, and CBR analytical model, $\bar{D}_{f_i,CBR}$, as depicted in Figure 24 and compare the results by computation of improvement percentages per flow, η_{f_i}. As shown in Figure 25, our proposed analytical model is up to 39.3% more accurate than CBR analytical model and more than 31% on average over all flows.

7.5. Discussion About Other Metrics

Although the paper targets an analytical model for latency bound, we briefly consider evaluating other metrics including throughput, communication load, energy consumption, and area requirements.

The network throughput is the sum of the data rates that are delivered to all ejection channels in a network and communication load is estimated by utilized bandwidth and

calculated as the sum of the data rates injected to the network. As the paper models the network which is not saturated, the throughput and communication load have the same values. This value is equal to 0.296 $flits/cycle$ for the synthetic example in Section 7.2 and 0.73 $flits/cycle$ for VOPD application in Section 7.3.

Network calculus does not directly evaluate energy consumption and area requirements. However, we can present a comparative discussion between VBR and CBR analyses, which is the main contribution of this work. Since we study the classic input-queuing virtual-channel router, there is nothing new or changed in the structure and design details of routers. In terms of area, what brings difference is in the calculated backlog, which determines the buffer size thresholds. In network calculus, the upper bound on backlog along the network is computed by the sum of the individual bounds on every element [Le Boudec et al. 2004]. Thus, the total required buffer for flow i is bounded by:

$$\bar{B}_i = \sum_{j \in L_{f_i}} \bar{B}_{ij} \tag{26}$$

where $\bar{B}_{ij}$ is the upper bound on the buffer size for flow i in each channel $j \in L_{f_i}$ and L_{f_i} is the set of channels along the path of flow i. $\bar{B}_{ij}$ for VBR traffic flows, $\bar{B}_{ij}^{VBR}$, and CBR traffic flows, $\bar{B}_{ij}^{CBR}$, are given by Eq. (27) and Eq. (28), respectively.

$$\bar{B}_{ij}^{VBR} = \sigma_i + \rho_i T_j + ((\sigma_i - L_j)/(p_i - \rho_i) - T_j)^+ [(p_i - R_j)^+ - p_i + \rho_i] \tag{27}$$

$$\bar{B}_i^{CBR} = \sigma_i + \rho_i T_j \tag{28}$$

In Eq. (27), term $[(p_i - R_j)^+ - p_i + \rho_i]$ is negative because $R_j \geq \rho_i$ and $p_i \geq R_j$ due to channel capacity constraint and the assumption stated in Section 4, respectively. Further, term $((\sigma_i - L_j)/(p_i - \rho_i) - T_j)^+$ is always positive because $a^+ = a$, if $a \geq 0$; $a+ = 0$, otherwise. Therefore, $((\sigma_i - L_j)/(p_i - \rho_i) - T_j)^+ [(p_i - R_j)^+ - p_i + \rho_i] < 0$ which means that $\bar{B}_{ij}^{VBR} \leq \bar{B}_{ij}^{CBR}$. In Section 7.2.3, we have calculated the required buffer size (buffer size threshold) in each input port of routers for a synthetic example. The sum of these values is the total required buffer size, $\bar{B}^{VBR}$, which is equal to 42 ***flits***. If we calculate the total required buffer size for CBR analysis, $\bar{B}^{CBR}$, by Eq. (26) and (28), it would be equal to 51 ***flits***, which is about 21.4% larger than $\bar{B}^{VBR}$. Similarly, $\bar{B}^{VBR}$ is calculated for VOPD application as a realistic traffic pattern by summing buffer size bounds derived in Section 7.3. The calculations show that $\bar{B}^{VBR} = 1673$ ***flits*** and $\bar{B}^{CBR} = 2827$ ***flits***. Therefore VBR analysis leads to about 40.8% reduction of the total required buffers. We have also derived $\bar{B}^{CBR}$ and $\bar{B}^{VBR}$ for the case study represented in Section 7.4 which is a 8×8 mesh network under the transpose traffic pattern. Due to calculations, $\bar{B}^{CBR} = 18256$ ***flits*** and $\bar{B}^{VBR} = 12556$ ***flits*** which shows that the total required buffers is reduced about 31.2% by VBR analysis. As a result, under the same network and application, VBR analysis gives tighter backlog bound than CBR analysis and can thus give more accurate bounds on the buffer requirements. From the design perspective, the tighter backlog bounds lead to the area saving in the router buffers.

Regarding power consumption, the network power comprises router power (buffer, switch, control circuit) and link power which are traffic dependent. It is notable that although VBR analysis derives tighter delay bounds, it does not change the packet transfer behavior, because it is only deriving more accurate analytical delay bounds without any change in design features of the router like switching, control, and link traversal. Therefore, the design decision of the router which our analysis brings impact on is the buffer dimensioning. Assuming the same system model, VBR analysis can indeed derive tighter bounds than CBR analysis on buffer requirements, leading to

power consumption saving. Following a power model for the buffers using, e.g. Orion [Shi et al. 2002], we can safely assume that the power consumption for buffers will decrease proportionally to the buffer size.

8. CONCLUSIONS

In this work, we have derived the analysis procedure to investigate per-flow delay bound. To this end, we have given theorems to calculate end-to-end ESC and internal output arrival curves in a FIFO multiplexing network. Based on the proposed analysis technique, we have conducted case studies of worst-case performance analysis, considered the accuracy of the proposed model through simulation, and compared it with a method without considering the traffic peak behavior. Analysis steps can be applied for larger networks with more flows. We have developed algorithms to automate analysis steps. In the future, we plan to develop network calculus models to investigate different scheduling policies and then compare them. We also plan to extend the proposed analytical method in case of back-pressure in the network. Authors in [Qian et al. 2009] [Zhao et al. 2013] use network calculus to analyze Worst-case Delay Bounds for CBR flows due to back-pressure in the network. It would be interesting to derive possibly tighter delay bound for VBR flows. In this respect, we have to extend the analytical models under a given fixed buffer size rather than to-be-determined bounded buffer size.

REFERENCES

BAKHOUYA, M., SUBOH, S., GABER, J.,EL-GHAZAWI, T., AND NIAR, S. 2011. Performance evaluation and design tradeoffs of on-chip interconnect architectures. *Simulation Modelling Practice and Theory, Elsevier. 19,* 6, 1496-1505.

BAUER, H., SCHARBARG, J. L., AND FRABOUL, C. 2010. Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach. *IEEE Transactions on Industrial Informatics. 6,* 4, 521–533.

BEN-ITZHAK, Y., CIDON, I., AND KOLODNY, A. 2011. Delay Analysis of Wormhole Based Heterogeneous NoC. In *Proceedings of the International Networks-On-Chip Symposium (NOCS)*. 161–168.

BENNETT, J. C. R., BENSON, K.,CHARNY, A.,COURTNEY, W. F., AND LE BOUDEC, J. -Y. 2002. Delay jitter bounds and packet scale rate guarantee for expedited forwarding. *IEEE/ACM Transactions on Networking. 10,* 4, 529–540.

BERTOZZI, D., JALABERT, A., MURALI, S., TAMHANKAR, R., STERGIOU, S., BENINI, L., AND DE MICHELI, G. 2005. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Transactions on Parallel and Distributed Systems. 16,* 2, 113–129.

BISTI, L., LENZINI, L., MINGOZZI, E., AND STEA, G. 2010. DEBORAH: A Tool for Worst-case Analysis of FIFO Tandems. In *Proceedings of ISoLA 2010, LNCS 6415*. 152–168.

BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z., AND WEISS, W. 1998. *An architecture for differentiated services*. IETF RFC 2475.

BOGDAN, P., AND MARCULESCU, R. 2007. Quantum-like effects in network-on-chip buffers behavior. In *Proceedings of the 44th Design Automation Conference (DAC)*. 266–267.

BOGDAN, P., KAS, M., MARCULESCU, R., AND MUTLU, O. 2010. QuaLe: A Quantum-Leap Inspired Model for Non-stationary Analysis of NoC Traffic in Chip Multi-processors. In *Proceedings of the International Networks-On-Chip Symposium (NOCS)*. 241–248.

BOUILLARD, A., JOUHET, L., AND THIERRY, E. 2010. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of Infocom*. 1316–1324

BOUILLARD, A. AND JUNIER, D. 2011. Worst-case delay bounds with fixed priorities using network calculus. In *Proceedings of Valuetools*. 381–390

BOYER, M. 2010. Half-modelling of shaping in FIFO net with network calculus. *RTNS 2010*.

CHANG, C. 2000. *Performance Guarantees in Communication Networks*. Springer-Verlag.

CHARNY, A. AND LE BOUDEC, J. -Y. 2000. Delay Bounds in a Network with Aggregate Scheduling. In *Proceedings of QofIS*. 1–13.

CIUCU, F., AND SCHMITT, J. 2012. Perspectives on Network Calculus - No Free Lunch but Still Good Value. *ACM Sigcomm. 42,* 4, 311–322.

GEBALI, F. AND ELMILIGI, H., EDITORS 2009. *Networks on Chip: Theory and Practice*. Taylor and Francis Group LLC - CRC Press.

JAFARI, F., LU, Z., JANTSCH, A., AND YAGHMAEE, M. H. 2010. Buffer Optimization in Network-on-Chip Through Flow Regulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). 29,* 12, 1973–1986.

JAFARI, F., JANTSCH, A., AND LU, Z. 2011. Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling. In *Proceedings of the International Conference on Computer Design (ICCD'11)*. Amherst, USA, 445–446.

JAFARI, F., JANTSCH, A., AND LU, Z. 2012. Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'12)*. Dresden, Germany, 538–541.

JIANG, Y. 2002. Delay bounds for a network of guaranteed rate servers with FIFO aggregation. *Computer Networks. 40,* 6, 683–694.

JIANG, N., BECKER, D. U., MICHELOGIANNAKIS, G., BALFOUR, J., TOWLES, B., KIM, J., AND DALLY, W. J. 2013. A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 86–96.

KIASARI, A. E., JANTSCH, A., AND LU, Z. 2013. Mathematical formalisms for performance evaluation of networks-on-chip. *ACM Computing Surveys. 45,* 3.

KIEFER, A., GOLLAN, N., AND SCHMITT, J.B. 2010. Searching for Tight Performance Bounds in Feed-Forward Networks. In *Proceedings of MMB/DFT*. 227–241.

HANSSON, A., WIGGERS, M.,MOONEN, A., GOOSSENS, K., AND BEKOOIJ, M. 2008. Applying dataflow analysis to dimension buffers for guaranteed performance in networks on chip. In *Proceedings of NOCS*. 211–212.

LE BOUDEC, J. Y. AND THIRAN, P. 2004. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag.

LEE, S. Real-Time Wormhole Channels. *Parallel Distributed Computer. 63,* 299–311.

LENZINI, L., MARTORINI, L.,MINGOZZI, E., AND STEA, G. 2006. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks. *Performance Evaluation. 63,* 9, 956–987.

LENZINI, L., MINGOZZI, E., AND STEA, G. 2008. A Methodology for Computing End-to-end Delay Bounds in FIFO-multiplexing Tandems. *Elsevier Performance Evaluation. 65,* 11-12, 922–943.

MARTIN, S., MINET, P., AND GEORGE L. 2003. Deterministic End-to-End Guarantees for Real-Time Applications in a DiffServ-MPLS Domain. In *Proceedings of SERA 2003, LNCS 3026*. 51–73.

MARTIN, S. AND MINET, P. 2006. Schedulability analysis of flows scheduled with FIFO: Application to the Expedited Forwarding class. In *Proceedings of IPDPS*. Rhodes Island, 25–29.

MOADELI, M., SHAHRABI, A., VANDERBAUWHEDE, W., AND OULD-KHAOUA M. 2007. An analytical performance model for the spidergon noc. In *Proceedings of 21st AINA*. 1014–1021.

OGRAS, U. Y., HU, J., AND MARCULESCU R. 2005. Key research problems in noc design: A holistic perspective. In *Proceedings of CODES+ISSS 2005*. 69–74.

QIAN, Y., LU, Z., AND DOU, W. 2009. Analysis of Worst-case Delay Bounds for Best-effort Communication in Wormhole Networks on Chip. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*. ACM/IEEE, San Diego, CA, 44–53.

QIAN, Y., LU, Z., AND DOU, Q. 2010. QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin. In *Proceedings of the 28th International Conference on Computer Design (ICCD'10)*. Amerstedam, the Netherlands, 52–59.

RAHMATI, D., MURALI, S., BENINI, L., ANGIOLINI, F., DE MICHELI, G., AND SARBAZI-AZAD, H. A method for calculating hard QoS guarantees for Networks-on-Chip. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'09)*. 579–586.

RAHMATI, D., MURALI, S., BENINI, L., ANGIOLINI, F., DE MICHELI, G., AND SARBAZI-AZAD, H. Computing Accurate Performance Bounds for Best Effort Networks-on-Chip. *IEEE Transactions on Computers (IEEE-TC). 62,* 3, 452–467.

RIZK, A. AND FIDLER, M. 2012. Non-asymptotic End-to-end Performance Bounds for Networks with Long Range Dependent fBm Cross Traffic. *Computer Networks. 56,* 1, 127–141.

SCHMITT, J. B., ZDARSKY, F. A., AND FIDLER, M. 2008. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch .... In *Proceedings of INFOCOM*. 1669–1677.

SHI, Z., AND BURNS A. 2008. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS 2008). IEEE Computer Society*. 161–170.

VAN DER TOL, E.B. AND JASPERS, E.G. T. 2002. Mapping of MPEG4 Decoding on a Flexible Architecture Platform. *SPIE. 4674,* , 1–13.

STILIADIS, D. AND VARMA, A. 1998. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking. 6,* 5, 611–624.

WROCLAWSKI, J. 1997. *The Use of RSVP with IETF Integrated Services*. IETF RFC 2210.

WANG, H. S., ZHU, X.,PEH, L. S., AND MALIK S. 2002. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture (MICRO).*

ZHAO, X. and LU, Z. 2013. Per-flow Delay Bound Analysis Based on a Formalized Micro-architectural Model. In *Proceedings of the 7th ACM/IEEE International Symposium on Networks-on-Chip (NOCS'2013)*, Tempe Arizona, USA, April 2013.

APPENDIX

To prove the proposed theorems, we need to use three theorems defined in network calculus as follows:

Theorem 4. *(Delay Bound [Le Boudec et al. 2004]). Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β, the delay $d(t)$ for all t satisfies: $d(t) \leq h(\alpha, \beta)$.*

The theorem says that the delay is bounded by the horizontal deviation between the arrival and service curves.

Theorem 5. *(Output Flow [Le Boudec et al. 2004]). With the same assumption as in Theorem 4. The output flow is constrained by the arrival curve $\alpha^* = \alpha \oslash \beta$.*

Now, we consider a node which guarantees a minimum service curve to an aggregate flow and also handles packets in order of arrival at the node.

Theorem 6. *(FIFO Service Curves [Le Boudec et al. 2004]). Consider a lossless node serving two flows, 1 and 2, in FIFO order. Assume that packet arrivals are instantaneous and the node guarantees a service curve β to the aggregate of the two flows. Assume that flow 2 has α_2 as an arrival curve. Define the family of functions $\beta^{eq}(t, \alpha_2, \tau) \equiv \beta_1^{eq}(t, \tau)$*

$$\beta_1^{eq}(t,\tau) = [\beta(t) - \alpha_2(t-\tau)]^+_{\{t>\tau\}}$$

For any $\tau \geq 0$ such that $\beta_1^{eq}(t, \tau)$ is wide-sense increasing, then flow 1 is guaranteed the service curve $\beta_1^{eq}(t, \tau)$.

A. PROOF OF THEOREM 1

Theorem 1: (Delay Bound) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage $\gamma_{\sigma_x,\rho_x}, \quad 1 \leq x \leq n$, this means we have:

$$\beta = \delta_T \otimes [\otimes_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}]$$

and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho_\beta^* \geq \rho$ ($\rho_\beta^* = min_{1\leq x\leq n}\rho_x$), then the maximum delay for the flow is bounded by

$$h(\alpha,\beta) = T + \left[\vee_{1\leq x\leq n}\frac{L - \sigma_x + \theta\,(p-\rho_x)^+}{\rho_x}\right]^+$$

Proof. *As stated before, the delay is bounded by the maximum horizontal deviation between the arrival and service curves. Thus, due to Figure 26, if $p \leq min_{1\leq x\leq n}(\rho_x)$, we have:*

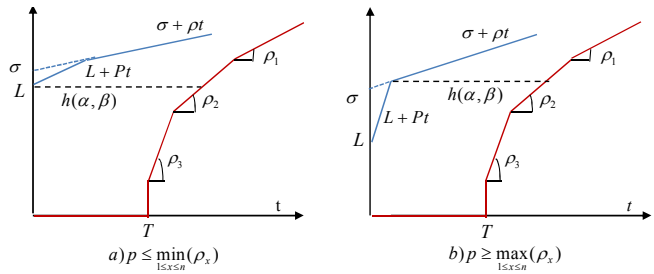


Fig. 26. Computation of delay bound for one VBR flow served by a pseudo affine curve.

$$\begin{cases} L = \sigma_1 + \rho_1 (t_1 - T) \Rightarrow t_1 = T + \dfrac{L - \sigma_1}{\rho_1} \\ L = \sigma_2 + \rho_1 (t_2 - T) \Rightarrow t_2 = T + \dfrac{L - \sigma_2}{\rho_2} \\ \vdots \quad \vdots \quad \vdots \\ L = \sigma_n + \rho_n (t_n - T) \Rightarrow t_n = T + \dfrac{L - \sigma_n}{\rho_n} \end{cases} \tag{29}$$

$$\Rightarrow h(\alpha, \beta) = max_{1 \le x \le n} t_x = T + \left[\vee_{1 \le x \le n} \frac{L - \sigma_x}{\rho_x} \right]^+ \tag{30}$$

If $p \geq max_{1 \le x \le n}(\rho_x)$, due to Figure 26, we have:

$$\begin{cases} L + p\theta = \sigma_1 + \rho_1 (t_1 + \theta - T) \\ \qquad \Rightarrow t_1 = T + \dfrac{L + p\theta - \sigma_1}{\rho_1} - \theta \\ L + p\theta = \sigma_2 + \rho_2 (t_2 + \theta - T) \\ \qquad \Rightarrow t_2 = T + \dfrac{L + p\theta - \sigma_2}{\rho_2} - \theta \\ \qquad \vdots \quad \vdots \quad \vdots \\ L + p\theta = \sigma_n + \rho_n (t_n + \theta - T) \\ \qquad \Rightarrow t_n = T + \dfrac{L + p\theta - \sigma_n}{\rho_n} - \theta \end{cases}$$

$$\Rightarrow h(\alpha, \beta) = max_{1 \le x \le n} t_x = T + \left[\vee_{1 \le x \le n} \frac{L + p\theta - \sigma_x}{\rho_x} - \theta \right]^+$$

$$= T + \left[\vee_{1 \le x \le n} \frac{L - \sigma_x + \theta (p - \rho_x)}{\rho_x} \right]^+ \tag{31}$$

From Eq. 30 and 31, we can say:

$$h(\alpha,\beta) = T + \left[\vee_{1\le x\le n}\frac{L-\sigma_x+\theta\left(p-\rho_x\right)^+}{\rho_x}\right]^+ \tag{32}$$

If $min_{1\le x\le n}(\rho_x) < p < max_{1\le x\le n}(\rho_x)$, we calculate t_x for each γ_{σ_x,ρ_x} and then obtain the maximum one which is $h(\alpha,\beta)$. When $min_{1\le x\le n}(\rho_x) < p < max_{1\le x\le n}(\rho_x)$, this means for some ρ_x, $p > \rho_x$ and for the others $p < \rho_x$. Let us assume $p < \rho_i$ where $i = 1,\ldots,m$ and $p > \rho_j$ where $j = m,\ldots,n$.

For $p < \rho_i$ where $i = 1,\ldots,m$, we have $t_i = T + \left[\frac{L-\sigma_i}{\rho_i}\right]^+$. Since $p < \rho_i$, we can rewrite it as below:

$$t_i = T + \left[\frac{L-\sigma_i+\theta\left(p-\rho_i\right)^+}{\rho_i}\right]^+ \qquad i = 1,\ldots,m \tag{33}$$

When $p > \rho_j$, t_j is given by:

$$t_j = T + \left[\frac{L+p\theta-\sigma_j}{\rho_j} - \theta\right]^+ = T + \left[\frac{L-\sigma_j+\theta\left(p-\rho_j\right)^+}{\rho_j}\right]^+ \qquad j = m,\ldots,n \tag{34}$$

Due to Equation (33) and (34), we can say:

$$t_x = T + \left[\frac{L-\sigma_x+\theta\left(p-\rho_x\right)^+}{\rho_x}\right]^+ \qquad x = 1,\ldots,n \tag{35}$$

which means $h(\alpha,\beta) = max_{1\le x\le n}t_x = T + \left[\vee_{1\le x\le n}\frac{L-\sigma_x+\theta(p-\rho_x)^+}{\rho_x}\right]^+$.

Hence, we proved the theorem.

B. PROOF OF PROPOSITION 1

Proposition 1: (Equivalent Service Curve) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage γ_{σ_x,ρ_x}, $\quad 1 \le x \le n$, this means we have:

$$\beta = \delta_T \otimes \left[\otimes_{1\le x\le n}\gamma_{\sigma_x,\rho_x}\right] = \delta_T \otimes \left[\wedge_{1\le x\le n}\gamma_{\sigma_x,\rho_x}\right]$$

and let $\alpha = min(L+pt, \sigma+\rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \ge \rho$ ($\rho^*_\beta = min_{1\le x\le n}\rho_x$) and $p \ge \rho^\circ_\beta$ ($\rho^\circ_\beta = max_{1\le x\le n}\rho_x$), then the ESC obtained by subtracting arrival curve α, $\{\beta^{eq}(\alpha,\tau), \quad \tau = h(\alpha,\beta)\} \equiv \beta^{eq}(\alpha)$, with

$$\begin{aligned}\beta^{eq}(\alpha) = \delta_{T+\vee_{1\le i\le n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ + \theta} \otimes \Big[\otimes_{1\le x\le n}\Big[\\ \gamma_{\rho_x\left\{\vee_{1\le i\le n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ - \frac{\sigma-\sigma_x-(\rho_x-\rho)\theta}{\rho_x}\right\},\rho_x-\rho}\Big]\Big]\end{aligned} \tag{36}$$

Proof. *Let us apply Theorem 6 to service curve β as follows.*

$$\begin{aligned}\beta^{eq}(\alpha,\tau) = &\left[\delta_T \otimes \left[\otimes_{1\le x\le n}\gamma_{\sigma_x,\rho_x}\right]\right. \\ &\left.-min\left(L+p\left(t-\tau\right), \sigma+\rho\left(t-\tau\right)\right)\right]\end{aligned} \tag{37}$$

Eq. (37) is wide-sense increasing for any $\tau \geq 0$. *Since we assumed* $\tau = h(\alpha, \beta)$, *due to Proposition 1, we have:*

$$\tau = T + \left[\vee_{1 \leq x \leq n} \frac{L - \sigma_x + \theta \left(p - \rho_x\right)^+}{\rho_x} \right]^+ \tag{38}$$

Without losing generality, we follow proof for $n = 1$. *Therefore, by Eq. (38) we have:*

$$\tau - T = \left[\frac{L - \sigma_x + \theta \left(p - \rho_x\right)^+}{\rho_x} \right]^+ \tag{39}$$

We then apply Theorem 6 to service curve $\acute{\beta}$ *(*$\acute{\beta}$ *is* β *when* $n = 1$*) as follows.*

$$\begin{aligned} \beta^{\acute{e}q}(\alpha, \tau) &= \delta_T \otimes \gamma_{\sigma_x, \rho_x} - min\left(L + p\left(t - \tau\right), \sigma + \rho\left(t - \tau\right)\right) \\ &= \sigma_x + \rho_x \left(t - T\right) - min\left(L + p\left(t - \tau\right), \sigma + \rho\left(t - \tau\right)\right) \end{aligned} \tag{40}$$

We now consider two situations including $0 \leq t - \tau \leq \theta$ *and* $t - \tau > \theta$.
If $0 \leq t - \tau \leq \theta \Rightarrow min\left(L + p\left(t - \tau\right), \sigma + \rho\left(t - \tau\right)\right) = L + p\left(t - \tau\right)$. *Let us assume* $\acute{t} = t - \tau \Rightarrow t - T = \acute{t} + (\tau - T)$. *From Eq. 39, we can say* $t - T = \acute{t} + \left[\frac{L - \sigma_x + \theta\left(p - \rho_x\right)^+}{\rho_x}\right]^+$.

$$\begin{aligned} \beta^{\acute{e}q}(\alpha, \tau) &= \sigma_x + \rho_x \left(\acute{t} + \left[\frac{L - \sigma_x + \theta \left(p - \rho_x\right)^+}{\rho_x} \right]^+ \right) \\ &\quad - \left(L + p\acute{t}\right) \\ &= \sigma_x + \rho_x \acute{t} + \left[L - \sigma_x + \theta \left(p - \rho_x\right)^+ \right]^+ - L - p\acute{t} \\ &\leq - \left(p - \rho_x\right) \acute{t} + \theta \left(p - \rho_x\right)^+ \end{aligned}$$

Since $p \geq \rho_\beta^\circ$ *and* $\acute{t} \leq \theta$, *we have:*

$$\begin{aligned} \beta^{\acute{e}q}(\alpha, \tau) &= - \left(p - \rho_x\right) \acute{t} + \theta \left(p - \rho_x\right)^+ \\ &\leq - \left(p - \rho_x\right) \theta + \theta \left(p - \rho_x\right) \leq 0 \end{aligned}$$

Therefore, $\beta^{\acute{e}q}(\alpha, \tau) = 0$ *where* $0 \leq t - \tau \leq \theta$. *By definition of the service curve, we can say that if* $0 \leq t \leq \theta + \tau$ *then* $\beta^{\acute{e}q}(\alpha, \tau) = 0$, *and this means that the offset of* $\beta^{\acute{e}q}(\alpha, \tau)$ *is equal to* $\tau + \theta$.

If $t - \tau > \theta \Rightarrow min\left(L + p\left(t - \tau\right), \sigma + \rho\left(t - \tau\right)\right) = \sigma + \rho\left(t - \tau\right)$. *Therefore,* $\beta^{\acute{e}q}(\alpha, \tau) = \sigma_x + \rho_x \left(t - T\right) - \left(\sigma + \rho\left(t - \tau\right)\right)$. *If* $\rho_x \tau$ *is added to and subtracted from* $\beta^{\acute{e}q}(\alpha, \tau)$, *we have*

$$\begin{aligned} \beta^{\acute{e}q}(\alpha, \tau) &= \sigma_x + \rho_x \left(t - T\right) - \left(\sigma + \rho\left(t - \tau\right)\right) + \rho_x \tau - \rho_x \tau \\ &= \sigma_x - \sigma + \rho_x \left(\tau - T\right) + \left(\rho_x - \rho\right)\left(t - \tau\right) \\ &= \delta_\tau \otimes \gamma_{\sigma_x - \sigma + \rho_x (\tau - T), \rho_x - \rho} \end{aligned} \tag{41}$$

Since we concluded that the offset of $\beta^{\acute{e}q}(\alpha, \tau)$ *is* $\tau + \theta$, *we add* $\left(\rho_x - \rho\right)\theta$ *to Eq. 41 and then subtract it. We obtain:*

$$\begin{aligned}\beta^{\acute{e}q}(\alpha,\tau) &= \sigma_x - \sigma + \rho_x(\tau - T) + (\rho_x - \rho)(t - \tau) \\ &\quad + (\rho_x - \rho)\theta - (\rho_x - \rho)\theta \\ &= \sigma_x - \sigma - \rho\theta + \rho_x(\tau + \theta - T) + (\rho_x - \rho)(t - \tau - \theta) \\ &= \delta_{\tau+\theta} \otimes \gamma_{\sigma_x - \sigma - \rho\theta + \rho_x(\tau+\theta-T), \rho_x - \rho} \end{aligned} \tag{42}$$

Thus, the offset of $\beta^{eq}(\alpha,\tau)$ is equal to $\tau+\theta$. Furthermore, each leaky bucket-stage in $\beta^{eq}(\alpha,\tau)$ can be computed as $\gamma_{\acute{\sigma_j},\acute{\rho_j}}$, with $\acute{\sigma_j} = \sigma_x - \sigma - \rho\theta + \rho_x(\tau+\theta-T)$ and $\acute{\rho_j} = \rho_j - \rho$. Therefore, we have $\beta^{eq} = \delta_{\tau+\theta} \otimes [\otimes_{1\le x\le n} \gamma_{\acute{\sigma_x},\acute{\rho_x}}]$ and by substituting (38) into β^{eq}, we prove the proposition.

C. PROOF OF THEOREM 2

Theorem 2: (Equivalent Service Curve for Rate-Latency Service Curve with $K+1$ Flows) Consider one node with a rate-latency service curve $\beta_{R,T} = \delta_T \otimes \gamma_{0,R}$. Let $\alpha_i = min(L_i + p_i t, \sigma_i + \rho_i t) = \gamma_{L_i,p_i} \wedge \gamma_{\sigma_i,\rho_i}$ be arrival curve of flow i and $p_i \ge R - \sum_{j=1}^{i-1}\rho_j$, where $1 \le i \le K+1$ and $K+1$ is the number of flows passing through that node as shown in Figure 3. The ESC for flow $K+1$ in the node, obtained by subtracting K arrival curves, is:

$$\beta^{eq}_{K+1} = \delta_{T+\sum_{i=1}^{K}\left(\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+ + \theta_i\right)} \otimes \gamma_{0,R-\sum_{j=1}^{K}\rho_j} \tag{43}$$

Proof. *We use the simplest form of mathematical inductive proof method. It proves that a statement involving a number N holds for all values of N. The proof consists of two steps:*

Base Step: *In this step, we show that the statement holds when $N=1$. In order to verify this, we compute the ESC obtained by subtracting one arrival curve ($N=1$), offered by Proposition 2:*

$$\beta^{eq}_{2} = \delta_{T+\left[\frac{L_1+\theta_1(p_1-R)^+}{R}\right]^+ + \theta_1} \otimes \gamma_{0,R-\rho_1} \tag{44}$$

If we apply Proposition 1 for a rate-latency service curve $\beta_{R,T}$ where $n=1$, $\sigma_x = 0$ and $\rho_x = R$, Eq. 44 is easily obtained. Therefore, the statement holds when $N=1$.

Inductive Step: *In this step, we show if the statement holds for some N, then the statement also holds when $N+1$ is substituted for N. Assume that β^{eq}_{N+1} is an ESC for flow $N+1$, obtained by subtracting N arrival curves as represented in Eq. 43. We shall compute ESC β^{eq}_{N+2} for flow $N+2$. Therefore, in this case we should subtract $N+1$ arrival curves. After subtracting N arrival curves, the ESC for aggregate flow $\{N+1, N+2\}$ will be equal to β^{eq}_{N+1}. Therefore, for computing β^{eq}_{N+2}, it is enough to subtract flow $N+1$ from β^{eq}_{N+1} by applying Proposition 1.*

From β^{eq}_{N+1}, we can say n, ρ_x, σ_x and T_x in Proposition 1 are as $n=1$, $\rho_x = R - \sum_{j=1}^{N}\rho_j$, $\sigma_x = 0$, and $T_x = T + \sum_{i=1}^{N}\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+ + \sum_{j=1}^{N}\theta_j$. Also, α in Proposition 1 is equal to $\alpha_{N+1} = min(L_{N+1} + p_{N+1}t, \sigma_{N+1} + \rho_{N+1}t)$. After applying

Proposition 1 and computing some straightforward algebraic manipulation, β^{eq}_{N+2} is given by:

$$\beta^{eq}_{N+2} = \delta_{T+\sum_{i=1}^{N+1}\left(\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+ +\theta_t\right)} \otimes \gamma_{0,R-\sum_{j=1}^{N+1}\rho_j} \tag{45}$$

which proves the inductive step.

D. PROOF OF THEOREM 3

Theorem 3: (Output Arrival Curve with FIFO) Consider a VBR flow, with TSPEC (L, p, ρ, σ), served in a node that guarantees to the flow a pseudo affine service curve equal to $\beta = \delta_T \otimes \gamma_{\sigma_x,\rho_x}$. The output arrival curve α^* given by:

$$\alpha^* = \begin{cases} \theta > T & \gamma_{(p\wedge\rho_x)T+\theta(p-\rho_x)^+ + L-\sigma_x, p\wedge\rho_x} \\ & \wedge\gamma_{\sigma-\sigma_x+\rho T,\rho} \\ \theta \leq T & \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{cases} \tag{46}$$

Proof. *From Theorem 5, the output flow is constrained by the arrival curve $\alpha^* = \alpha \oslash \beta = sup_{u\geq 0}\{\alpha(t+u) - \beta(u)\}$. Thus, $\alpha^* = sup_{u\geq 0}\{min(\sigma+\rho(t+u), L+p(t+u)) - \sigma_x - \rho_x(u-T)^+\}$*

We now consider two different cases including $\theta \leq T$ and $\theta > T$. (1) If $\theta \leq T$, we have:

$$\begin{aligned}
\alpha^* &= sup_{u\geq 0}\{min(\sigma+\rho(t+u), L+p(t+u)) - \sigma_x \\
&\quad -\rho_x(u-T)^+\} \\
&= sup_{0\leq u\leq T}\{min(\sigma+\rho(t+u), L+p(t+u)) - \sigma_x\} \\
&\quad \vee sup_{u>T}\{min(\sigma+\rho(t+u), L+p(t+u)) \\
&\quad -\sigma_x - \rho_x u + \rho_x T\} \\
&= \{min(\sigma+\rho(t+T), L+p(t+T)) - \sigma_x\} \vee \\
&\quad sup_{u>T}\{min(\sigma+\rho(t+u), L+p(t+u)) - \sigma_x \\
&\quad -\rho_x u + \rho_x T\} \\
&= \{\sigma+\rho(t+T) - \sigma_x\} \vee sup_{u>T}\{\sigma+\rho(t+u) - \sigma_x \\
&\quad -\rho_x u + \rho_x T\} \\
&= \{\sigma+\rho(t+T) - \sigma_x\} \vee sup_{u>T}\{\sigma+\rho t + \rho_x T - \sigma_x \\
&\quad +u(\rho-\rho_x)\}
\end{aligned}$$

Since $\rho \leq \rho_x$ and thus $\rho - \rho_x$ is negative, u in the second term should get its lowest possible value to achieve supremum. Thus, we have

$$\begin{aligned}
&= \{\sigma+\rho(t+T) - \sigma_x\} \vee \{\sigma+\rho(t+T) - \sigma_x\} \\
&= \sigma+\rho(t+T) - \sigma_x = \gamma_{\sigma-\sigma_x+\rho T,\rho}
\end{aligned} \tag{47}$$

(2) If $\theta > T$, we have:

$$\begin{aligned}\alpha^* &= sup_{u\geq 0}\left\{min\left(\sigma+\rho\left(t+u\right), L+p\left(t+u\right)\right)-\sigma_x- \right.\\ &\qquad \left.\rho_x\left(u-T\right)^+\right\}\\ &= sup_{0\leq u\leq T}\left\{min\left(\sigma+\rho\left(t+u\right), L+p\left(t+u\right)\right)-\sigma_x\right\}\\ &\qquad \vee sup_{u>T}\left\{min\left(\sigma+\rho\left(t+u\right), L+p\left(t+u\right)\right)-\sigma_x\right.\\ &\qquad \left.-\rho_x u+\rho_x T\right\}\\ &= \left\{min\left(\sigma+\rho\left(t+T\right), L+p\left(t+T\right)\right)-\sigma_x\right\}\vee sup_{u>T}\left\{\right.\\ &\qquad min\left(\sigma+\rho\left(t+u\right)-\sigma_x-\rho_x u+\rho_x T, L+p\left(t+u\right)\right.\\ &\qquad \left.\left.-\sigma_x-\rho_x u+\rho_x T\right)\right\} \end{aligned} \tag{48}$$

For completing the proof, we need to consider the second term in right side of Eq. (48) in detail. Therefore, we call it $Term_2$ *in the following:*

$$\begin{aligned}Term_2 =& sup_{u>T}\left\{min\left(\sigma+\rho\left(t+u\right)-\sigma_x-\rho_x u+\rho_x T,\right.\right.\\ &\left.\left. L+p\left(t+u\right)-\sigma_x-\rho_x u+\rho_x T\right)\right\}\end{aligned}$$

For solving $Term_2$*, we consider two different situations including* $t+u\leq\theta$ *and* $t+u\geq\theta$*. Thus, if* $t+u\geq\theta$*, we have* $u>T$ *and* $t+u\geq\theta$*.*

$$\begin{aligned}\Rightarrow Term_2 &= sup_{u>T}\left(\sigma+\rho\left(t+u\right)-\sigma_x-\rho_x u+\rho_x T\right)\\ &= sup_{u>T}\left(\sigma+\rho t+\rho_x T-\sigma_x+\left(\rho-\rho_x\right)u\right)\\ &= \sigma+\rho t+\rho_x T-\sigma_x+\left(\rho-\rho_x\right)T\\ &= \sigma+\rho\left(t+T\right)-\sigma_x=\gamma_{\sigma-\sigma_x+\rho T,\rho}\end{aligned} \tag{49}$$

If $t+u\leq\theta$*, we have* $u>T$ *and* $t+u\leq\theta\Rightarrow u\leq\theta-t$*.*

$$\begin{aligned}\Rightarrow Term_2 &= sup_{T<u\leq\theta-t}\left(L+p\left(t+u\right)-\sigma_x-\rho_x u+\rho_x T\right)\\ &= sup_{T<u\leq\theta-t}\left(L+pt+\rho_x T-\sigma_x+\left(p-\rho_x\right)u\right)\end{aligned}$$

Selecting an appropriate value for u *depends on if* $(p-\rho_x)$ *is positive or negative. Therefore, we have two different situations including* $p>\rho_x$ *and* $p\leq\rho_x$*. If* $p>\rho_x\Rightarrow(p-\rho_x)$ *is positive and* u *should be the highest possible value to have supremum value. Thus, due to* $u=\theta-t$*,* $Term_2=L+\rho_x\left(t+T\right)-\sigma_x+\theta\left(p-\rho_x\right)$*. If* $p\leq\rho_x\Rightarrow(p-\rho_x)$ *is negative. Therefore,* u *gets its lowest value and* $Term_2$ *is equal to* $L+p\left(t+T\right)-\sigma_x$*.*

$$\Rightarrow Term_2 = L+\left(p\wedge\rho_x\right)\left(t+T\right)-\sigma_x+\theta\left(p-\rho_x\right)^+ \tag{50}$$

From Eq. 48, 49 and 50, if $\theta>T$*, we have:*

$$\begin{aligned}\alpha^* &= min\left(L+\left(p\wedge\rho_x\right)\left(t+T\right)-\sigma_x+\theta\left(p-\rho_x\right)^+,\right.\\ &\qquad \left.\sigma+\rho\left(t+T\right)-\sigma_x\right)\\ &= \gamma_{(p\wedge\rho_x)T+\theta(p-\rho_x)^++L-\sigma_x,p\wedge\rho_x}\wedge\gamma_{\sigma-\sigma_x+\rho T,\rho}\end{aligned} \tag{51}$$

Hence, we prove the theorem.

Paper 14

Weighted Round Robin Configuration for Worst-Case Delay Optimization in Network-on-Chip

F. Jafari
A. Jantsch
Z. Lu

Submitted to IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD).

Weighted Round Robin Configuration for Worst-Case Delay Optimization in Network-on-Chip

Fahimeh Jafari*, Axel Jantsch†, and Zhonghai Lu*
*KTH Royal Institute of Technology, Sweden
†Vienna University of Technology (TU Wien), Austria

***Abstract*—We propose an approach for computing the end-to-end delay bound of individual Variable Bit-Rate (VBR) flows in a FIFO multiplexer with aggregate scheduling under Weighted Round Robin (WRR) policy. To this end, we use the theorems proposed based on network calculus to derive per-flow end-to-end Equivalent Service Curve (ESC) employed for computing Least Upper Delay Bounds (LUDBs) of individual flows. Since real time applications are going to meet guaranteed services with lower delay bounds, we optimize weights in WRR policy to minimize LUDBs while satisfying performance constraints. We formulate two constrained delay optimization problems, namely, Minimize-Delay and Multi-objective optimization. Multi-objective optimization has both total delay bounds and their variances as minimization objectives. The proposed optimizations are solved using a genetic algorithm. A realistic case study exhibits 15.4% reduction of total worst-case delays and 40.3% reduction on the sum of variances of delays when compared with round robin policy. The optimization algorithm has low run-time complexity, enabling quick exploration of large design spaces. We conclude that an appropriate weight allocation can be a valuable instrument for delay optimization in NoC designs.**

I. Introduction

Many multi-core Systems on Chip (SoC) require different levels of service for different applications. Real-time applications have stringent performance requirements; the correctness relies on not only the communication result but also the end-to-end delay bound. A data packet received too late could be useless. In other words, the Least Upper Delay Bound (LUDB) for each packet must not exceed its deadline. In such systems, it is desirable to minimize the end-to-end delay bound of the traffic streams satisfying their QoS requirements. Therefore, the first important consideration is to derive the LUDB for a given communication flow. Since, in such systems, resources are shared among multiple communication flows, we analysis the interference in the shared resources for a given flow. To this end, based on the Network Calculus theory [1], we have presented and proved required theorems based on network calculus in [2][3]. We then presented a methodology [4] to consider resource sharing scenarios and also derive end-to-end Equivalent Service Curve (ESC) and LUDB by applying the proposed theorems. We assume that all traffic can be well characterized as flows and scheduled as aggregates which means multiple flows are scheduled as an aggregate flow. For a given flow, we study the maximum interference of all other flows based on the Network Calculus. Our proposed models [4] have been defined under Round Robin (RR) policy. RR policy uses the same service level for each connection while multiple service levels allow to better adapt to the application requirements by providing different bandwidth and latency guarantees. A Weighted Round Robin (WRR) scheduling policy assigns weights to concurrent communications to define multiple service levels. Higher service levels have greater weights and do not preempt lower ones. It is important for designers to find appropriate weights in WRR policy such that the corresponding service levels can support QoS requirements for each communication connection. It is desirable to also optimize delay and throughput in the network.

In this paper, we extend our earlier proposed methodology for RR [4] to WRR policy. We then address an optimization problem of minimizing the total delay bounds subject to the performance constraints of the applications running on the SoC. Moreover, to avoid an unfair service in which some flows have to wait for a very long time, we investigate another goal which is minimizing the variances of delay bounds in different flows. As both mentioned objective functions are worthwhile for the real-time applications, we formulate them as a multi-objective problem under QoS constraints. Finally, we show the benefits of the proposed method and quantify performance improvement.

Regarding optimization problems presented in this paper, random variables appear in the formulation of the optimization problem which causes random objective functions. Such optimization problems are usually solved by metaheuristic methods which do not guarantee an optimal solution. However, they usually find high-quality solutions in reasonable time [5]. There is a wide variety of metaheuristics like simulated annealing, tabu search, iterated local search, evolutionary computation, and genetic algorithms. We compare the performance of several metaheuristics (*pure random search*, *markov monotonous search*, *adaptive search*, and *genetic algorithm*) and conclude that a genetic algorithm based method is most suitable.

The rest of this paper is organized as follows. Section II discusses related works. Section III introduces the basics of Network Calculus. Section IV is devoted to the underlying system model and notations in our analysis. Section V introduces the major features of our formal method for analyzing the contention scenarios and computation of LUDB along with an example. The proposed optimization problems and corresponding solutions are represented in Section VI and VII. Section VIII implements the algorithms for solving

978-3-9810801-8-6/DATE12/©2012 EDAA

the proposed optimization problems. Experimental results are reported in Section IX. Finally, Section X concludes the paper.

II. Related Work

A. Performance Evaluation of Real-time Services

In networks employing aggregate scheduling, the performance analysis of real-time services is a challenging and complex issue. Aggregate scheduling arises in many cases such as on-chip networks and large-size networks. For instance, The Differentiated Services (DiffServ) [6] is an architecture based on aggregate scheduling in the Internet. Bennet et al. provide a survey on the subject [7].

Charny and Boudec [8] derive a closed-form delay bound in a generic network assuming the fluid model. An extended model is proposed [9] to look into packetization effects. The main limitation of these models is that they work well only for small utilization factors in a generic network configuration. Lenzini et al. [10] describe a methodology for obtaining per-flow worst-case delay bound in tandem networks of rate-latency nodes traversed by leaky-bucket shaped flows. This method yields better bounds than those previously proposed. Qian et al. [12] present analytical models for traffic flows under strict priority queueing and weighted round robin scheduling in on-chip networks. They then derive per-flow end-to-end delay bounds using these models.

All previous works on this subject investigate computing delay bounds only for average behavior of flows and they do not consider peak behavior, which results in less accurate bounds. Since a considerable number of real time applications are transmitted by VBR traffics, we have proposed a methodology presented [4] to consider performance analysis for VBR traffic characterized by (L, p, σ, ρ) in on-chip networks employing aggregate resource management. As such, this method achieves more accurate delay bounds.

In this paper, we extend our proposed method for weighted round robin policy. Then we regulate weights in each router to minimize delay bounds while satisfying performance constraints.

B. Optimization Method

We formulate optimization problems to optimize the weights in weighted round robin policy with respect to worst-case delay bounds. Since the proposed constrained problems are stochastic optimization problem, metaheuristics can be an efficient method for providing good solution quickly.

In recent years, a great interest has been devoted to metaheuristics. The term metaheuristic is commonly associated with random search algorithms. One pioneer contribution is the proposition of the Pure Random Search (PRS) which is a simple stochastic search algorithm, presented by Brooks in 1958 [15]. Different techniques of local random search (markov monotonous search) were proposed by White in 1971 [16]. The simulated annealing method was introduced by Kirkpatrick et al. in 1982 [17] which makes it possible for the system to escape local optima. Previous metaheuristics do not explicitly use memory, except the selection of the best solutions. The actual first usage of memory in modern metaheuristics is probably due to Tabu search proposed by Glover [18] in 1986. Farmer et al. proposed the artificial immune system [19] as a novel approach inspired by the specifications of the immune system which uses memory and learning to solve a problem. In 1988, Koza registered his first patent on genetic programming, published in 1992 [20]. The basic idea is to use the genetic principle to gradually produce the best programs for a given problem. A well known book on genetic algorithms was published by Goldberg in 1989 [21]. In 1992, Dorigo completed his PhD thesis, in which he innovates ant colony optimization [22]. In 1993, the first algorithm based on bee colonies was proposed by Walker et al. [23]. Another significant progress is the development of the particle swarm optimization by Kennedy and Eberhart in 1995 [24]. In 1997, Storn and Price proposed differential evolution [25] as a vector-based evolutionary algorithm which is more efficient than previous algorithms in many applications. In 2002, Passino introduced an optimization algorithm based on bacterial foraging [26] which is a common solution for various optimization problems such as transport modeling and scheduling. Then, Simon proposed a biogeography-based optimization algorithm in 2008 [27].

The considerable development of metaheuristics is because of the significant increase in the processing power of the computers, and the development of massively parallel architectures.

In this paper, we solve the proposed optimization problems using genetic algorithm. To show that GA has a good run-time efficiency for solving the proposed optimizations, in Section IX-C, we present a comparative study between commonly used metaheuristics such as pure random search, markov monotonous search (local random search), adaptive search, and genetic algorithm.

III. Network Calculus Background

Network calculus is a collection of results which gives deep insights into deterministic queuing systems found in communication networks [1]. It can be used for example to analyze flow problems encountered in networking, model schedulers, and compute worst case bounds used in guaranteed services. Network calculus uses min-plus algebra to convert non-linear queueing systems into linear systems. The algebra structure of min-plus is $(\Re \cup \{+\infty\}, \wedge, +)$ in which the "multiplication" operation is $+$, and the "addition" operation is $\wedge$. $\wedge$ represents the minimum operation, $f \wedge g = min(f, g)$. The min-plus convolution, denoted by $\otimes$, is defined as $(f \otimes g)(t) = inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}$; where two functions f and g are wide-sense increasing functions.

Arrival curve and service curve are the most significant concepts in network calculus. An arrival curve defines an upper bound on the cumulative arrival process and a service curve defines a lower bound on the cumulative service process. Network calculus uses the abstraction of arrival curve to characterize a traffic flow f_j which is an infinite stream of unicast traffic sent from a source node and also employs the abstraction of

a service curve to model a network element processing traffic flows. In this paper, we use Traffic SPECification (TSPEC) [28] for characterizing traffic to look into both the average and peak behaviors of a flow. With TSPEC, the arrival curve of flow f_j is defined as $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate ($p_j \geq \rho_j$), σ_j the burstiness ($\sigma_j \geq L_j$), and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. A well-formulated service model to reflect the service capability of a node is the rate-latency function defined as $\beta_{R,T} = R(t-T)^+$, where R is the minimum service rate and T the maximum processing latency of the node. We use x^+ to denote the function $x^+ = x$ if $x > 0; x^+ = 0$, otherwise. More notations in network calculus, employed through our analysis models in this paper, are introduced as follows.

$\vee$ represents the maximum operation, $f \vee g = max(f, g)$. *Burst delay* function $\delta_T(t) = +\infty$, if $t > T$; $\delta_T(t) = 0$, otherwise. *Affine function* $\gamma_{b,r}(t) = b + rt$, if $t > 0$; $\gamma_{b,r}(t) = 0$, otherwise. Therefore, $\delta_T \otimes \gamma_{b,r}(t) = b + r(t-T)$. $\oslash$ represents the min-plus deconvolution as $(f \oslash g)(t) = sup_{s \geq 0}\{f(t+s) - g(s)\}$. A pseudoaffine curve represents the service received by single flows in tandems of FIFO multiplexing rate-latency nodes [10] and defined as $\beta = \delta_T \otimes [\otimes_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$. Due to concave affine curves, it can be rewritten as $\beta = \delta_T \otimes [\wedge_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$, where T is denoted as *offset*, and the affine curves between square brackets as leaky-bucket stages.

The following theorems are used in this paper to derive LUDB per flow. We have proposed and proved these theorems in [2][3].

Theorem 1. *(Output Arrival Curve with FIFO) Consider a VBR flow, with TSPEC (L, p, ρ, σ), served in a node that guarantees to the flow a pseudo affine service curve $\beta = \delta_T \otimes \gamma_{\sigma_x, \rho_x}$. The output arrival curve α^* given by:*

$$\alpha^* = \begin{cases} \theta > T & \gamma_{(p \wedge \rho_x)T + \theta(p-\rho_x)^+ + L - \sigma_x, p \wedge \rho_x} \\ & \wedge \gamma_{\sigma - \sigma_x + \rho T, \rho} \\ \theta \leq T & \gamma_{\sigma - \sigma_x + \rho T, \rho} \end{cases} \quad (1)$$

where $\theta = (\sigma - L)/(p - \rho)$.

Proof. We have proved it in [2]. □

Theorem 2. *Let $\beta = \delta_T \otimes \gamma_{\sigma_x, \rho_x}$ be a pseudo affine curve, with offset T and one leaky-bucket stage $\gamma_{\sigma_x, \rho_x}$, and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho_x \geq \rho$ and $p \geq \rho_x$, then the ESC obtained by subtracting arrival curve α, β^{eq}*

$$\beta^{eq} = \delta_{T + \left[\frac{L - \sigma_x + \theta(p-\rho_x)^+}{\rho_x}\right]^+ + \theta} \otimes \gamma_{0, \rho_x - \rho} \quad (2)$$

Proof. We have proved it in [3]. □

Theorem 3. *(Delay Bound) Let β be a pseudo affine curve, with offset T and n leaky-bucket stages $\gamma_{\sigma_x, \rho_x}$, $1 \leq x \leq n$, this means we have:*

$$\beta = \delta_T \otimes [\otimes_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}] = \delta_T \otimes [\wedge_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$$

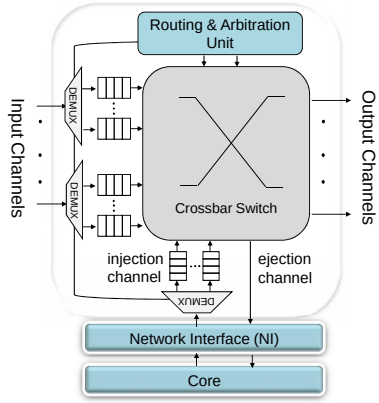


Fig. 1. The structure of a single node in NoC architecture

and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \geq \rho$ ($\rho^*_\beta = min_{1 \leq x \leq n} \rho_x$), then the maximum delay for the flow is bounded by

$$h(\alpha, \beta) = T + \left[\vee_{1 \leq x \leq n} \frac{L - \sigma_x + \theta\,(p - \rho_x)^+}{\rho_x}\right]^+ \quad (3)$$

Proof. We have proved it in [3]. □

IV. System Model

Figure 1 shows an NoC architecture in which every node contains an core equipped with a Network Interface (NI) and a router with input and output channels. Assumptions in this paper are given as follows:

- The NoC architecture can have different topologies.
- A flow consists of packets and each packet is broken into flits. We consider the arbitration granularity of one word with a fixed word length of L_w for all flows. L_w is assumed to be 1 *flit*.
- Packets have fixed length and traverse the network in a best-effort fashion with virtual-cut-through switching technique using a deadlock-free deterministic routing.
- Routers have only input buffers and Virtual Channels (VCs).
- The router can have multiple VCs per in-port. VC allocation is deterministic and each VC receives an aggregate service.
- Buffers are bounded due to the threshold calculated in Eq. (10) and the network is lossless.
- All traffic is modeled as TSPEC flows $f = TSPEC(L, p, \sigma, \rho)$ at the entry into the network.
- To characterize flows based on their defined TSPEC, we assume unbuffered leaky bucket controllers (regulators) which do not buffer the packets, but stall the traffic producers or IPs [11].

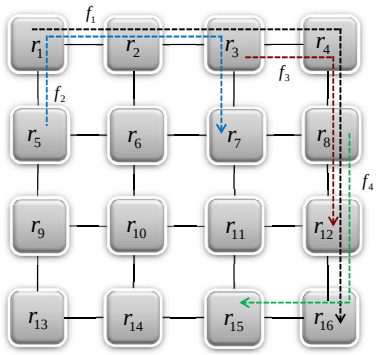


Fig. 2. An example of an NoC with 16 nodes and 4 flows.

- We assume weighted round robin arbitration and model it by a rate-latency service curve as $\beta = \delta_T \otimes \gamma_{0,R}$, it is assumed that $\rho \leq R$ and $p \geq R$.
- Flows are classified into a pre-specified number of aggregates.
- Traffic of each aggregate is buffered and transmitted in the FIFO order, denoted as FIFO multiplexing.
- Different aggregates are buffered separately and each aggregate is guaranteed a rate-latency service curve.
- The hardware limits the peak rate to 1 *flit/cycle*.

Figure 2 depicts an example with 16 nodes and 4 flows. Multiple flows which share the same buffer and channel in the same router, for example f_1 and f_2 in router 2, are scheduled as an aggregate flow denoted as $f_{\{1,2\}}$. The *tagged flow* is a flow for which the delay bound is derived and the other flows which compete with the tagged flow for the same resource are called *contention flows*. In the example, if f_1 is the tagged flow, f_2, f_3, and f_4 would be contention flows. Table I presents notations in this work.

Notations with sub-index "(f_i, r_j)" indicate that they are related to flow f_i in router r_j. For instance, $\alpha_{(f_1,r_2)}$ indicates the arrival curve of f_1 in router r_2. Using f_{s_i} instead of f_i in the sub-index means that the notation is related to the f_{s_i} which can be one flow or an aggregate flow. For example, $\beta_{(\{1,2\},r_2)}$ refers to aggregate flow $f_{\{1,2\}}$ in router r_2.

TABLE I
THE LIST OF NOTATIONS

f_i	Flow i
$F^{RPV}_{(j,i,k)}$	The set of flows passing through VC k in physical channel i of router j
$F_{(j,l,s,k)}$	The set of flows passing from VC s of input channel l to output channel k in router j
α_i	The arrival curve of f_i
α^*_i	The output arrival curve of f_i
Input PC#	The number assigned to an input physical channel
Output PC#	The number assigned to an output physical channel
VC#	The number assigned to an input virtual channel
$InPC$	The set of input physical channels in each router
$OutPC$	The set of output physical channels in each router
$InVC$	The set of input virtual channels in each input physical channel
L_i	The maximum transfer size of f_i *(flits)*
p_i	The peak rate of f_i *(flits/cycle)*
σ_i	The burstiness of f_i *(flits)*
ρ_i	The average rate of f_i *(flits/cycle)*
$Src(i)$	The source node of f_i
r_j	Router j
β_j	The service curve of r_j
R	The minimum service rate in a rate-latency service curve
T^l	The maximum processing latency of the arbiter in the router *(cycles)*
T^{HoL}	The maximum waiting time in the FIFO queue of the router *(cycles)*
T^{Total}	The total processing delay which comes from contention flows the router and equals to the sum of T^l and T^{HoL}
D_{router}	Time spent for packet routing decision *(cycles)*
L_w	The word length in the flow *(flits)*
C	The channel capacity *(flits/cycle)*
CF_t	The set of contention flows of tagged flow f_t
s_i	The set of joint flows in an aggregate flow (when the number of elements of s_i is equal to 1, there is only a single flow)
f_{s_i}	An aggregate flow of s_i
$\|s_i\|$	The cardinality of set s_i, which is a measure of the "number of elements of the set"
$S = \{s_i\}$	A set of s_i's in a tandem of routers
s^m	A set which has the maximum cardinality between the sets in S. $s^m = \{s_x \| \|s_x\| = max(\|s_i\|); \forall s_i \in S\}$
f_{s^m}	The flow related to s^m
r^m	The router related to s^m
β^m	The service curve related to s^m
$F^B_{(s_i,r_j)}$	The set of flows which share the same buffer in router r_j with flow f_{s_i}
$w_{(j,l,s,k)}$	The weight assigned to node r_j, input Physical Channel (PC) l, input VC s, and output PC k
L_{WR}	The length of a round in WRR policy

V. LUDB Derivation for WRR Policy

In [2] and [3], based on network calculus, we have presented and proved the required theorems to derive delay bounds for flows constrained by dual leaky bucket (VBR flows) in on-chip FIFO networks with aggregate scheduling and multiple virtual channels. In [4], we have applied the theorems to obtain per-flow LUDB under the same system model assuming round robin scheduling policy. In this section, we extend this method to weighted round robin policy.

To derive delay bound per flow passing a series of nodes, one simple way is to calculate the summation of delay bounds at each node, which results in a loose delay bound. A corollary called *Pay Bursts Only Once* is known to give a tighter upper estimate on delay bounds, when an end-to-end service curve is obtained prior to delay computations. This accounts for bursts of the tagged flow only once instead of at each link independently. This principle also holds in aggregate scheduling networks. To this end, we propose the two following steps to derive the end-to-end service curve for a tagged flow:

- **Step 1:** *Intra-router ESC*: This step derives intra-router ESCs for each router through which the tagged flow is passing. Different resource sharing scenarios in each router are distinguished and *intra-router analysis models* are built.

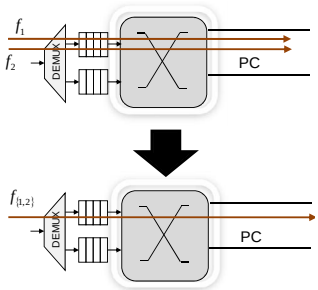


Fig. 3. An example of channel&buffer sharing

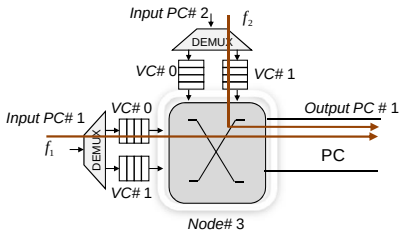


Fig. 4. An example of channel sharing

- **Step 2:** *Inter-router ESC*: In this step, according to the intra-router analysis models, we present a set-theoretic approach which recognizes and investigates different contention scenarios that a flow may experience along its routing path and in turn derive an end-to-end ESC for the tagged flow.

For extending our proposed analytical method to weighted round robin policy, we should expand the first step while the second step keeps the same principles. Similarly, to support some other arbitration policies, only the first step must be modified.

A. Intra-router ESC

In this step, we consider three types of resource sharing, including channel&buffer sharing, channel sharing, and buffer sharing.

1) Channel&Buffer Sharing: As shown in Figure 3, multiple flows share both the same buffer and channel in the router, and are scheduled as a flow called aggregate flow. An aggregate flow including the tagged flow is named as tagged aggregate flow. In this case, intra-ESC is derived for the tagged aggregate flow instead of the tagged flow. In Section V-B, due to contention scenarios, we will remove contention flows from the ESC of a tagged aggregate flow in order to extract the ESC of the tagged flow.

2) Channel Sharing: Figure 4 depicts an example of a channel shared between two flows f_1 and f_2. The WRR arbiter associates a weight $w_{(j,l,s,k)}$ in cycles on each aggregate/single flow f_{s_i} passing from input VC s of input Physical Channel (PC) l in router r_j to output PC k. The value of the weight assigned to a channel depends on flows passing through that channel. Then, the router will try to give the flow a period of $w_{(j,l,s,k)}$ cycles before moving to the next node. In each round, for a non-empty VC buffer encountered, the router serves up to corresponding configured weight in cycles. The maximum length of a round consequently equals to $\sum_{l,s} w_{(j,l,s,k)}$ cycles, denoted as L_{WR}. The least service offered to one flow in a VC is completely dependent on the weight of that VC and the sum of all other weights. With the WRR scheduling, the worst case appears for a flow when it just misses its slot in the current round. Consequently it will have to wait for its slot assigned at the next round. In the worst case, each flow f_{s_i} passing from input VC s of input PC l in router r_j to output PC k will have to wait up to $\left(\sum_{p,q} w_{(j,p,q,k)} - w_{(j,l,s,k)}\right) \times \left(\frac{L_w}{C} + D_{router}\right)$ cycles before to be served, and get at least a $\frac{w_{(j,l,s,k)}}{\sum_{p,q} w_{(j,p,q,k)}} \times C$ of the channel bandwidth, where C is the channel capacity, L_w the word length, and D_{router} the delay for packet routing decision in a router. A flow may get more service rate if other flows use less, but we now know a worst-case lower bound on the bandwidth. Based on network calculus theory, we can use the abstraction of service curve to model a weighted round robin arbiter in router r_j for flow f_{s_i} as a rate-latency server $\beta_{(s_i,r_j)} = R_{(s_i,r_j)}(t - T^l_{(s_i,r_j)})^+$, where $R_{(s_i,r_j)}$ is the minimum service rate and $T^l_{(s_i,r_j)}$ is the maximum processing latency of the arbiter in router r_j for flow f_{s_i}. $R_{(s_i,r_j)}$ and $T^l_{(s_i,r_j)}$ are defined as follows:

$$R_{(s_i,r_j)} = \frac{w_{(j,l,s,k)}}{\sum_{p,q} w_{(j,p,q,k)}} \times C \tag{4}$$

$$T^l_{(s_i,r_j)} = \left(\sum_{p,q} w_{(j,p,q,k)} - w_{(j,l,s,k)}\right) \times \left(\frac{L_w}{C} + D_{router}\right) \tag{5}$$

In the example of Figure 4:

$$R_{(f_1,r_3)} = \frac{w_{(3,1,0,1)}}{w_{(3,1,0,1)}+w_{(3,2,1,1)}} \times C$$

$$T^l_{(f_1,r_3)} = w_{(3,2,1,1)} \times \left(\tfrac{L_w}{C} + D_{router}\right)$$

3) Buffer Sharing: Figure 5 depicts a buffer shared between two flows f_1 and f_2. In this type of sharing, we introduce two kinds of delay for a tagged flow including:

- *Head-of-Line delay (HoL)* is the maximum waiting time of the packet in the FIFO queue, which is denoted by T^{HoL}.
- *Processing delay* is the maximum processing latency of the router's arbiter for the flow, which is denoted by T^l.

Therefore, total delay for tagged flow f_i in router r_j is calculated as $T^{Total}_{(f_i,r_j)} = T^{HoL}_{(f_i,r_j)} + T^l_{(f_i,r_j)}$.

$T^l_{(f_i,r_j)}$ and $R_{(f_i,r_j)}$ can be calculated according to Equation (5) and (4), respectively. To show how $T^{HoL}_{(f_i,r_j)}$ is calculated, we consider the example in Figure 5 and assume that f_1 is the tagged flow. As depicted in the figure, $T^{HoL}_{(f_1,r)}$ is equal to the maximum delay for passing packets of flow f_2 in the buffer.

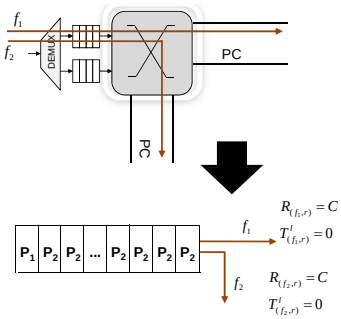


Fig. 5. An example of buffer sharing

According to [1], the maximum delay for flow f_j is bounded by Equation (6).

$$\bar{D}_{(f_j,r)} = T^l_{(f_j,r)} + \frac{L_j + \theta_j(p_j - R_{(f_j,r)})^+}{R(f_j,r)} \tag{6}$$

Therefore, $T^{HoL}_{(f_1,r)}$ is given as follows:

$$T^{HoL}_{(f_1,r)} = T^l_{(f_2,r)} - \theta_2 + \frac{L_2 + \theta_2 p_2}{R_{(f_2,r)}} \tag{7}$$

In the case of more than one flow sharing the same buffer with the tagged flow, HoL delay for tagged flow f_{s_i} in router r_j is calculated as belows:

$$T^{HoL}_{(s_i,r_j)} = \sum_{\forall f_c \in F^B_{(s_i,r_j)}} T^{HoL(f_c)}_{(s_i,r_j)} \tag{8}$$

where $F^B_{(s_i,r_j)}$ is the set of flows which share the same buffer in router r_j with tagged flow f_{s_i}. Also $T^{HoL(f_c)}_{(s_i,r_j)}$ is given by

$$T^{HoL(f_c)}_{(s_i,r_j)} = T^l_{(f_c,r)} - \theta_c + \frac{L_c + \theta_c p_c}{R_{(f_c,r)}} \tag{9}$$

Therefore router r_j can give flow f_{s_i} service bounded by curve $\beta_{(s_i,r_j)} = \delta_{T^{Total}_{(s_i,r_j)}} \otimes \gamma_{0,R_{(s_i,r_j)}}$, where $T^{Total}_{(s_i,r_j)}$ is equal to $T^{HoL}_{(s_i,r_j)} + T^l_{(s_i,r_j)}$ and $R_{(s_i,r_j)}$ is calculated by Equation (4).

We analyze the buffer space threshold for each VC based on traffic specifications of flows passing through that VC, and also interference between them. The buffer space threshold for virtual channel k in physical channel i of router j is given as below:

$$B_{(j,i,k)} = \sum_{\forall f_c \in F^{RPV}_{(j,i,k)}} \left(\sigma_c + \rho_c T^p_{(f_c,r_j)} + \left(\theta - T^p_{(f_c,r_j)}\right)^+ \times \left[\left(p_c - R_{(f_c,r_j)}\right)^+ - p_c + \rho_c \right] \right) \tag{10}$$

where $F^{RPV}_{(j,i,k)}$ is the set of flows passing through VC k in physical channel i of router j.

B. Inter-router ESC

In this step, we aim to extract ESC of the tagged flow by removing the contention flows from the ESC of the tagged aggregate flows. We have described this stage in elaborate detail through paper [4]. Here, we show the procedure of deriving end-to-end ESC for a tagged flow with the help of the example in Figure 2. Assuming flow f_1 is the tagged flow, its routing path is shown in a tandem of routers in Figure 6(a).

After analyzing per-router resource sharing scenarios and deriving intra-router ESCs, we can view an analysis model which keeps per-router ESCs of a tagged flow or tagged aggregate flow as shown in Figure 6(b). This model is called *aggregate analysis model*. In this model, $\beta_{(s_i,r_j)}$ indicates that the service curve is related to flow f_{s_i} in router r_j. For instance, $\beta_{(\{1,2\},r_2)}$ is the service curve of aggregate flow $f_{\{1,2\}}$ in router r_2. A set of s_i's in a tandem of routers is denoted as $S = \{s_i\}$. For example, in Figure 6(b), $S = \{\{1\},\{1,2\},\{1\},\{1,3\},\{1,3,4\},\{1,4\},\{1\}\}$.

We use the theorem of *concatenation of network elements* [1] to model nodes sequentially connected and each is offering a rate-latency service curve to each of the aggregate flows $\beta_{(s_i,r_j)}$, $j = 1,2,...,n$ as a rate-latency server as follows:

$$\beta_{(s_i,r_{1,2,...,n})} = \beta_{(s_i,r_1)} \otimes \beta_{(s_i,r_2)} \otimes ... \otimes \beta_{(s_i,r_n)} \tag{11}$$

where the minimum service rate and the maximum processing latency in an equivalent rate-latency server are defined as follows:

$$\begin{aligned} R_{(s_i,r_{1,2,...,n})} &= \min\left(R_{(s_i,r_1)}, R_{(s_i,r_2)}, ..., R_{(s_i,r_n)}\right) \\ T^l_{(s_i,r_{1,2,...,n})} &= T^l_{(s_i,r_1)} + T^l_{(s_i,r_2)} + ... + T^l_{(s_i,r_n)} \end{aligned} \tag{12}$$

In Figure 6(b), sequentially connected service curves for the same aggregate flows do not exist. Thus, we can directly go to the next step which considers contention scenarios.

As illustrated in Figures 6(a), contention flow f_2 is nested in flow f_1 and contention flow f_3 is crossed with f_4. To consider contentions in this model and obtain inter-router ESC, we decompose a complex contention scenario to basic contention patterns and then remove contention flows one by one. The contention scenarios can be classified into two basic patterns, namely, *nested* and *crossed*. We apply the algebra of sets to recognize contention scenarios. Convenient notations are defined through the example in order to facilitate our discussion. To recognize the contention scenarios, we first find $s^m = \{s_x | |s_x| = max(|s_i|); \forall s_i \in S\}$, where $|s_x|$ is the cardinality (the number of elements) of set s_x. In other words, s^m is $s_x \in S$ with the maximum cardinality. The service curve, flow, and router related to s^m are denoted as f_{s^m}, β^m, and r^m, respectively. Thus, these notations in Figure 6(b) are given by $S = \{\{1\},\{1,2\},\{1\},\{1,3\},\{1,2,3,4\},\{1,4\},\{1\}\}$, $s^m = \{1,3,4\}$, $f_{s^m} = f_{\{1,3,4\}}$, $r^m = r_8$, and $\beta^m = \beta_{(\{1,3,4\},r_8)}$.

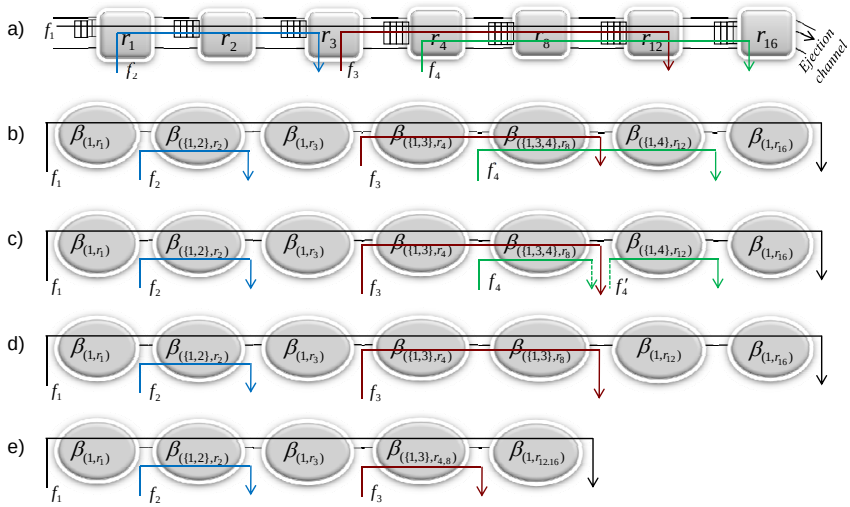


Fig. 6. An example of end-to-end ESC computation

The set placed before s^m in S is called s^{Prev} and the set after that s^{Next}. In this respect, the related aggregate flow, service curve, and router to s^{Prev} are denoted as $f_{s^{Prev}}$, β^{Prev}, and r^{Prev}, respectively. $f_{s^{Next}}$, β^{Next}, and r^{Next} are related to s^{Next} as well. Therefore, due to $s^m = _{\{1,3,4\}}$ in Figure 6(b), $s^{Prev} = \{1,3\}$, $\beta^{Prev} = \beta_{(\{1,3\},r_4)}$, $f_{s^{Prev}} = f_{\{1,3\}}$, $r^{Prev} = r_4$, $s^{Next} = \{1,4\}$, $\beta^{Next} = \beta_{(\{1,4\},r_{12})}$, $f_{s^{Next}} = f_{\{1,4\}}$, and $r^{Next} = r_{12}$.

Now, we can recognize contention scenarios as below:

1) if $s^{Next} \subset s^{Prev}$ then the contention is *nested*;
   - Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
2) else if $s^{Prev} \subset s^{Next}$ then the contention is *nested*;
   - Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
3) else
   - if $s^{Next} \subset s^m$ then the contention is *nested*;
     - Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
   - else if $s^{Prev} \subset s^m$ then the contention is *nested*;
     - Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
   - else, it is *crossed*.
     - The problem is strictly transformed to the combination of two nested flows

Regarding Figure 6(b), $s^m = \{1,3,4\}$, $s^{Prev} = \{1,3\}$, and $s^{Next} = \{1,4\}$. Since s^{Prev} is not a subset of s^{Next}, and vice versa, due to contention recognition procedure, this case is a crossed contention. There are two cross points, one between r_4 and r_8 and the other between r_8 and r_{12}. We cut f_4 at the second cross point, i.e., at the ingress of r_{12}, f_4 will be split into two flows, f_4 and $\acute{f}_4$, as shown in Figure 6(c). Then the problem is strictly transformed to the combination of nested flows such that f_4 is nested in flow f_3 and $\acute{f}_4$ in f_1. It is clear that the arrival curve $\alpha_{(f_4,r_{12})}$ equals to output arrival curve f_4 in router r_8, $\alpha^*_{(f_4,r_8)}$. To compute $\alpha^*_{(f_4,r_8)}$, we need to get the ESC of r_8 for f_4, $\beta_{(f_4,r_8)}$. Then, we calculate the output arrival curve of f_4 as $\alpha^*_{(f_4,r_8)} = \alpha_{(f_4,r_8)} \oslash \beta_{(f_8,r_8)}$ and remove nested flows f_4 and $\acute{f}_4$ from the tandem as shown in Figure 6(d). Deriving output arrival curve and removing the contention flows are done by applying our proposed Theorems 1 and 2 in [2][3].

After subtracting each contention flow from the ESC, we should apply the concatenation theorem again to find more equivalent servers and reduce the number of service curves. For instance, after removing contention flows f_4 and $\acute{f}_4$, the example looks like Figure 6(d). In this figure, the service curve of sub-tandem $\{r_4, r_8\}$ for aggregate flow $f_{\{1,3\}}$ is computed as $\beta_{(\{1,3\},r_{4,8})} = \beta_{(\{1,3\},r_4)} \otimes \beta_{(\{1,3\},r_8)}$ and also $\beta_{(1,r_{12,16})}$ is calculated as $\beta_{(1,r_{12})} \otimes \beta_{(1,r_{16})}$. The aggregate analysis model with new equivalent servers is shown in Figure 6(e).

If we repeat contention recognition steps, the next contention flows are f_2 and f_3 nested in flow f_1. Due to Figure 6(e), we have two options for s^m, one is $\{1,2\}$ and the other one $\{1,3\}$. When s^m is not unique, each of them can be selected. In this paper, we choose the first one from the left side in the aggregate analysis model. Therefore, $s^m = \{1,2\}$, $s^{Prev} = \{1\}$, and $s^{Next} = \{1\}$. In this case, $s^{Prev} \subset s^{Next}$ and also $s^{Prev} \subset s^{Next}$. Thus, it satisfies conditions 1 and 2 in contention recognition steps which state the contention is nested. It does not matter that which condition is followed since the results are the same. We particularly follow the first condition which states flow $f_{s^m - (s^m \cap s^{Next})}$ should be removed from β^m. In this example, we eliminate flow $f_{\{1,2\}-(\{1\}\cap\{1,2\})} = f_{\{2\}}$ from $\beta_{(\{1,2\},r_2)}$ to derive $\beta_{(1,r_2)}$ by applying Theorem 2 proposed in [4]. After that, convolution

$\beta_{(\{1\},r_{1,2,3})} = \beta_{(\{1\},r_1)} \otimes \beta_{(\{1\},r_2)} \otimes \beta_{(\{1\},r_3)}$ is calculated.

We similarly repeat contention recognition and convolution steps until $|s^m| \neq 1$. When $|s^m| = 1$, it means, the end-to-end ESC of tagged flow is obtained. In the example, if we perform contention recognition steps one more time by removing $f_{\{3\}}$ from $\beta_{(\{1,3\},r_{4,8})}$ to derive $\beta_{(1,r_{4,8})}$ and then calculate $\beta_{(\{1\},r_{1,2,3,4,8,12,16})} = \beta_{(\{1\},r_{1,2,3})} \otimes \beta_{(\{1\},r_{4,8})} \otimes \beta_{(\{1\},r_{12,16})}$, the example looks like Figure 7 and $\beta_{(\{1\},r_{1,2,3,4,8,12,16})}$ would be the end-to-end ESC of tagged flow f_1.

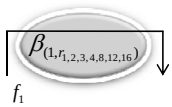


Fig. 7. The final stage of end-to-end ESC computation

Algorithm 1 presents all stages of deriving the end-to-end ESC for a given tagged flow as described through the example. The only difference between this algorithm and the one presented for RR [4] results from the different methods proposed for calculating intra-router ESCs (Line 9).

Now, we can obtain LUDB from end-to-end ESC according to Theorem 3 proposed in [3]. We have automated our proposed analytical approach as a tool for worst-case performance analysis.

VI. Optimization Problem Formulation

The members of a workshop on on-chip communication challenges in 2006 did agree that latency is one of the most critical challenges for on-chip interconnection network architectures [13]. From the design perspective, there exists a huge search space to explore the network for minimizing latency. Thus, to design a low latency on-chip network, designers need to investigate optimization problems and make appropriate decisions. The general problem is defined as below:

General Problem Definition
Given Architecture specifications, application parameters, and traffic characteristics (e.g. TSPEC in this paper);
Find A set of decision variables;
Such that network delay is minimized and performance constraints are satisfied.

In this formulation, decision variables can include finding an efficient application mapping to processing cores, traffic regulation parameters (e.g. peak rate, burstiness, and packet injection rates to the network), switch architecture, a resource allocation strategy (e.g., bandwidth of channels, etc.), weight configuration in WRR policy, and a routing algorithm.

In this paper, we find optimal weight configuration in WRR policy to minimize total worst-case delay in the network. Weight allocation is actually one of resource allocation strategies in which a flow with larger weight gets more bandwidth or higher service level. The weight of each non-empty VC is selected due to traffic specifications of flows passing through that VC, and also interference between them. In Section IX, we show that an unoptimized weight configuration increases network delay.

On the other hand, the faster transmission delay is not necessarily better in a shared communication channel since faster delivery requires higher link bandwidth reservation and may incur a larger delay for another contention flow in a shared channel, leading to an intolerable delay. To avoid throttling for some communications, we investigate another objective function which is minimizing the variances of delay bounds in different flows. As both mentioned goals are worthwhile for the real-time applications, we formulate them as a multi-objective problem in Section VI-B.

Algorithm 1 End-to-End ESC Algorithm

1: Find the set of contention flows of tagged flow f_t, denoted by CF_t
2: **for** $\forall j \in CF_t$ **do**
3: **if** $Src(j) \notin Path(t)$ **then**
4: Find $joiningnode = JoiningPoint(f_j)$
5: Calculate $X = ESC(f_j, Src(j), joiningnode)$
6: $\alpha_j = \alpha_j \oslash X$
7: **end if**
8: **end for**
9: Calculate intra-router ESC for WRR based on Section V-A.
10: Calculate $\beta_{(s_{i_1},r_{j_1})} \otimes \beta_{(s_{i_2},r_{j_2})} \otimes ... \otimes \beta_{(s_{i_n},r_{j_n})}$ *if* $i_1 = i_2 = ... = i_n$.
11: Find $s^m = \{s_x \| s_x| = max(|s_i|); \forall s_i \in S\}$.
12: **repeat**
13: **if** $s^{Prev} \subset s^{Next}$ **then**
14: Remove $f_{s^m-(s^m \cap s^{Next})}$ from β^m
15: **else if** $s^{Next} \subset s^{Prev}$ **then**
16: Remove $f_{s^m-(s^m \cap s^{Prev})}$ from β^m.
17: **else**
18: **if** $s^{Prev} \subset s^m$ **then**
19: Remove $f_{s^m-(s^m \cap s^{Prev})}$ from β^m
20: **else if** $s^{Next} \subset s^m$ **then**
21: Remove $f_{s^m-(s^m \cap s^{Next})}$ from β^m.
22: **else**
23: Find $joiningnode = JoiningPoint(f_{(s^m-s^{Prev})})$.
24: Calculate $X = ESC(f_{(s^m-s^{Prev})}, joiningnode, r^{Next})$.
25: $\acute{\alpha}_{(s^m-s^{Prev})} = \alpha_{(s^m-s^{Prev})} \oslash X$
26: Remove $f_{(s^m-s^{Prev})}$ from β^m.
27: Remove $\acute{f}_{(s^m-s^{Prev})}$ from β^{Next}.
28: **end if**
29: **end if**
30: Calculate $\beta_{(s_{i_1},r_{j_1})} \otimes \beta_{(s_{i_2},r_{j_2})} \otimes ... \otimes \beta_{(s_{i_n},r_{j_n})}$ *if* $i_1 = i_2 = ... = i_n$.
31: Find s^m.
32: **until** $|s^m| \neq 1$
33: **return** end-to-end ESC for tagged flow f_t

A. Delay Optimization

As stated before, our objective is to choose appropriate weights in weighted round robin policy, assigned to channels on the path of flows, so as to minimize the sum of LUDBs while satisfying acceptable performance in the network. It is worth mentioning that $w_{(j,l,s,k)} = 0$ when no flow is passing from virtual channel s of input channel l to output channel k in router j. Thus, the delay bound minimization problem, *Minimize-Delay*, can be formulated as follows.

Given a set of flows $F = \{f_i \propto (L_i, p_i, \sigma_i, \rho_i)\}$, routing matrix R, the number of weight cycles L_{WR}, *find* the weights in weighted round robin policy as $w_{(j,l,s,k)}$ for $\forall i \in N$, $\forall j \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$, such that

$$\min_{w_{(j,l,s,k)}} \sum_{\forall f_i \in F} D_i \tag{13}$$

subject to:

$$\sum_{l,s} w_{(j,l,s,k)} = L_{WR} \qquad \forall j \in N; \forall k \in OutPC \tag{14}$$

$$\frac{L_{WR} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)} \leq L_{WR} \qquad \forall j \in N, \forall l \in InPC, \forall s \in InVC, \forall k \in OutPC \tag{15}$$

where $w_{(j,l,s,k)}$ for $\forall j \in N$, $\forall l \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$ are optimization variables.

Eq. (13) is the objective function of this optimization problem which minimizes total LUDBs. Constraint (14) says that the sum of weights assigned to flows which pass through through the same output channel k in router j, the same weighted round robin scheduler, is equal to L_{WR}. Although we have assumed the same value of L_{WR} for all arbiters, the optimization problem can be easily adapted to different values of the sum of weights. To reach acceptable performance in the network, the share of $w_{(j,l,s,k)}$ from L_{WR} should be proportionate to $\frac{\sum_{m \in F_{(j,l,s,k)}} \rho_m}{C}$, where $F_{(j,l,s,k)}$ is the set of flows which pass through virtual channel s of input channel l to output channel k in router j. Therefore, we can consider $\frac{\sum_{m \in F_{(j,l,s,k)}} \rho_m}{C}$ as a criterion of minimum guaranteed performance for flows in $F_{(j,l,s,k)}$. In this respect, we have $\frac{\sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq \frac{w_{(j,l,s,k)}}{L_{WR}}$ which means $\frac{L_{WR} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)}$ as stated in Constraint (15). It is also clear that the value of each weight should be less than the number of weight cycles which means $w_{(j,l,s,k)} \leq L_{WR}$.

By following the equations described in Section V, the effect of optimization variables on the objective function of the defined problem is obvious.

In the literature, problem (13) is called a stochastic and nonlinear optimization problem [14]. We solve it using genetic algorithms because of their well-known robustness and ability to solve large and complex discrete optimization problems.

B. Multi-objective Optimization Problem

In order to avoid an intolerable delay of some flows due to processing and transmission of other flows, we would like to find appropriate weights in weighted round robin policy so that variance of delay bounds in the network is minimized. Using a general variance formula, we can calculate variances of the delay bounds as $\frac{1}{|F|} \times \sum_{\forall f_i \in F} (E(D) - D_i)^2$. Hence, another optimization problem can be formulated to minimize both the total delay bounds and their variance while satisfying the constrains (14) and (15), as follows.

Given a set of flows $F = \{f_i \propto (L_i, p_i, \sigma_i, \rho_i)\}$, routing matrix R, the number of weight cycles L_{WR}, *find* the weights in weighted round robin policy as $w_{(j,l,s,k)}$ for $\forall j \in N$, $\forall l \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$, such that

$$\min_{w_{(j,l,s,k)}} \sum_{\forall f_i \in F} D_i \tag{16}$$

$$\min_{w_{(j,l,s,k)}} \frac{1}{|F|} \times \sum_{\forall f_i \in F} (E(D) - D_i)^2 \tag{17}$$

subject to:

$$\sum_{l,s} w_{(j,l,s,k)} = L_{WR} \qquad \forall j \in N; \forall k \in OutPC \tag{18}$$

$$\frac{L_{WR} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)} \leq L_{WR} \qquad \forall j \in N, \forall l \in InPC, \forall s \in InVC, \forall k \in OutPC \tag{19}$$

Although the solution of multi-objective optimization problems consists of a set of solutions, the user needs only one solution. The decision about which solution is best depends on the *decision maker* and there is no a universally accepted definition of *optimum* as in single-objective optimizations [29]. A multi-objective problem is often solved by composing the objective function as the weighted sum of the objectives which is in general known as the *weighted-sum* or *scalarization* method. In this approach, a relative preference factor of the objectives should be known in advance. In more detail, the weighted-sum method minimizes a positively weighted sum of the objectives, that is,

$$min(\gamma_1 f_1 + \gamma_2 f_2) \tag{20}$$

where γ_1 and γ_2 are the weighting coefficients representing the relative importance of the objectives.

The simplicity and efficiency of this method makes it an appropriate option for solving multi-objective optimizations with complex and nonsmooth objective functions. Therefore, we convert our proposed multi-objective problem into a scalar optimization problem with equal weighting coefficients. Since the problem is still a nonsmooth and stochastic optimization, we use the genetic algorithm to solve it.

VII. Solution Method

The proposed optimization problems have complex and highly nonlinear objective functions. Moreover, due to Eq. (4) and (12), minimization functions of decision variables appear in the formulation of per-flow LUDBs and in turn in the objective formulation which cause random objective functions.

Such optimization problems are usually solved by meta-heuristic methods which make few assumptions about the

Algorithm 2 A General Scheme of GA in Pseudo-code

```
P1 ← Generate random population of n chromosomes
Evaluate the fitness f(x) for each x ∈ P1
repeat                                   ▷ Create a new population
    Selection: Select two parents from a population.
    Crossover: With a crossover probability cross over the
      parents to form a new offspring (children).
    Mutation: With a mutation probability mutate new
      offspring at each locus (position in chromosome).
    Accepting: Place new offspring in a new population
until the new population is not complete
Use new generated population for a further run.
if the end condition is satisfied then
    return The best solution in current population
else
    Go to step 2
end if
```

problem being solved and do not guarantee an optimal solution. However, they can usually find a good solution [5].

Among different types of metaheuristics, we choose genetic algorithms to solve the proposed optimization problems because they are most appropriate for large and complex non-linear models specially where the objective function is discontinuous, stochastic, very rugged and complex, noisy, or has many local optima [30], [31], [32]. Moreover, they have been proven to be effective at avoiding getting trapped in local optima and discovering the global optimum in even a problem with very complex objective functions [31]. GAs tend to be computationally expensive for the solutions of optimization problems with nonlinear equality and inequality constraints [32], which does not occur in our proposed problems. Although a GA does not always find a global optimum to a problem, it almost always finds high-quality solutions [31].

GA generates solutions to optimization problems mimicking the process of natural evolution such as inheritance, mutation, selection, and crossover. Algorithm 2 presents a general scheme of GA in pseudo-code. The algorithm is started with an initial population of solutions represented by *chromosomes*. A chromosome contains the solution as a set of parameters in form of *genes*. A gene is a position or set of positions in a chromosome, represented as a simple string or other data structures. The algorithm selects solutions, called *parents*, from the population and produces a new solution, called *offspring*, to form a new population. Although parents can be selected in many different ways, the main idea is that better parents according to their *fitness* hopefully will produce better offspring. *Crossover* and *mutation* are two basic operators of GA which produce a new offspring. This process is repeated until some condition, such as the number of populations or improvement of the best solution, is satisfied.

A method for encoding potential solutions of the problem is needed. There are different approaches to encode solutions like binary encoding, value encoding, permutation encoding, and tree encoding.

Algorithm 3 Genetic Algorithm

```
Pop1 ← Initilization_FirstPopulation()
Encoded_Pop1 ← Encoding(Pop1)
Temp_Pop ← Encoded_Pop1
for i=1 to Iteration# do
    New_Pop[0] ← Elitism(Lb, Ub)
    for j=1 to Pop_Size do
        Cross_Rate ← MersenneTwister()
        if (Cross_Rate ≤ Cross_Prob) then
            Chromosome1 ← Selection(Lb, Ub)
            Chromosome2 ← Selection(Lb, Ub)
            Offspring ← Crossover(Chromosome1, Chromosome2)
        else
            Offspring ← Selection(Lb, Ub)
        end if
        Mut_Rate ← MersenneTwister()
        if (Mut_Rate ≤ Mut_Prob) then
            Offspring ← Mutation(Offspring)
        end if
        New_Pop[j] ← Offspring
    end for
    Temp_Pop ← New_Pop
end for
Decoded_Pop ← Encoding(Temp_Pop)
Optimal_Weight ← Minimum(Decoded_Pop)
return Optimal_Weight
```

VIII. Implementation

We present Algorithm 3 to detail the procedure of deriving optimal weights for the proposed optimization problems. Parent is introduced as a vector of decision variables of weights, which presents the current solution for this round and offspring is a new vector generated from the parent which may be the next solution. The algorithm uses a binary representation of *chromosomes* as fixed-length strings over the alphabet $\{0, 1\}$, such that they are well suited to handle the optimization problems. It uses function $Encoding()$ to map solutions $\vec{w} \in W$ to a binary string $\{0, 1\}^l$ and defines function $Decoding()$ to do the reverse. To this end, real-valued vector $\vec{w} \in \Re^n$ is presented by a chromosome in form of a binary string $\vec{x} \in \{0, 1\}^l$. The chromosome is logically divided into n segments (*gene*) of equal length S_{gene} as $(w_1...w_n)$, where S_{gene} is gene size and $l = n \times S_{gene}$. Each gene w_i is decoded to yield the corresponding integer value, and the integer value is in turn linearly mapped to its interval of real values, denoted as $[Lb_i, Ub_i] \subset \Re$, where Lb_i and Ub_i indicate lower and upper bound constraints on w_i, respectively. In this work, we use a *gray code* interpretation of the binary string. The main advantage of gray codes is that they are different by only one bit.

Figure 8 shows an example of the decoding process for string segments of length $S_{gene} = 8$ which allows the representations of integers $\{0, 1, ..., 255\}$. As shown in the figure,

function $Decoding()$ first converts a given gray code to an integer value $p_i \in \{0, ..., 2^{S_{gene}} - 1\}$ and then maps p_i linearly to its corresponding interval $[Lb_i, Ub_i]$ as $Lb_i + \frac{Ub_i - Lb_i}{2^{S_{gene}} - 1} \times p_i$.

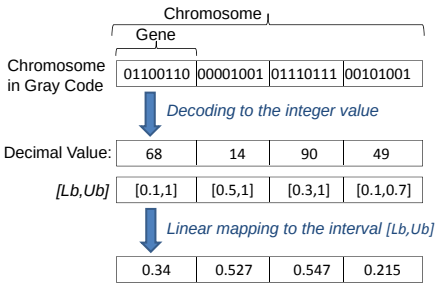


Fig. 8. An example of decoding and linear mapping

After encoding, the algorithm starts producing a new population in Line 5-20. Function $Elitism()$ in Line 5 copies the best chromosome of the current population to the new population, so the best chromosome found can survive. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution. To create other new offsprings, three basic operators including *selection*, *crossover*, and *mutation* are applied as follows.

Selection in GA means how to select parents for crossover or mutation. The main idea is to select the better parents in hope that the better parents will produce better offspring. Thus, function $Selection()$ in the algorithm selects randomly two chromosomes from the current population, evaluates their fitness values, and finally returns the one which has the smaller fitness value as one of parents. Another parent is selected in the same way.

$Cross\_Prob$ in Line 8 is the crossover probability which states how often a crossover is performed. If there is a crossover, two parents' chromosomes are selected and offspring is made from their crossover. If there is no crossover, offspring is the exact copy of a chromosome from the old population. Due to $Cross\_Prob$, the new generation is a mix of offsprings made by crossovers and chromosomes from the old population. Although crossovers have the tendency to improve chromosomes, it has been shown to be beneficial to keep part of the old population.

Crossover selects genes from parents' chromosomes and creates a new offspring. There are different ways to make a crossover. This algorithm chooses randomly two crossover points and everything before the first point and after the second point is copied from the first parent and the section between the two crossover points is copied from the second parent. Figure 10 shows an example of crossover applied in this algorithm (| denotes the crossover point).

After crossover, *mutation* is performed. Mut_Prob in Line 16 is the mutation probability which states how often a chromosome is mutated. If mutation is performed, parts of chromosome are changed. If there is no mutation, the offspring

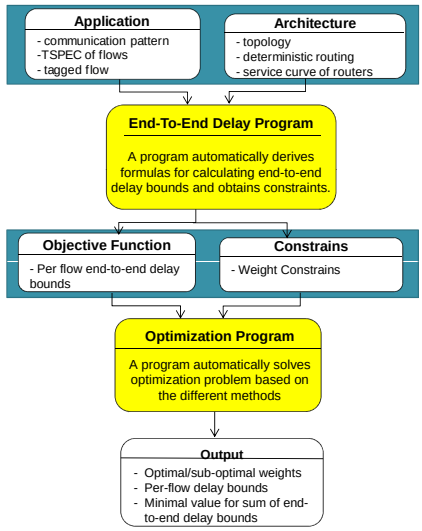


Fig. 9. The flow chart of the developed tool

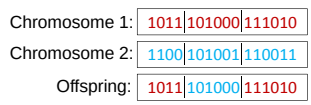


Fig. 10. An example of crossover

is copied after crossover without any change. Mutation is made to prevent an entire population being trapped in a local optimum. Mutation in Algorithm 3 changes the new offspring by randomly switching a few bits. It is worth mentioning that the mutation should not occur very often, because then GA will convert into a random search. Figure 11 shows an example of mutation used in the algorithm.

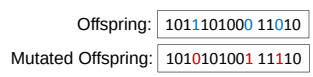


Fig. 11. An example of mutation

This process repeats for a specified number of iterations.

As shown in Figure 9, we have developed a tool in C++, divided into two main sub-tools including "End-to-End Delay Program" and "optimization Program". The former derives per-flow worst-case bounds by applying the proposed formal approach in Section V. The bounds are represented as functions of weights in WRR policy. The latter optimizes weights in WRR policy based on the optimization problem formulated in Section VI. Input for the first sub-tool includes an application communication graph, specification of flows,

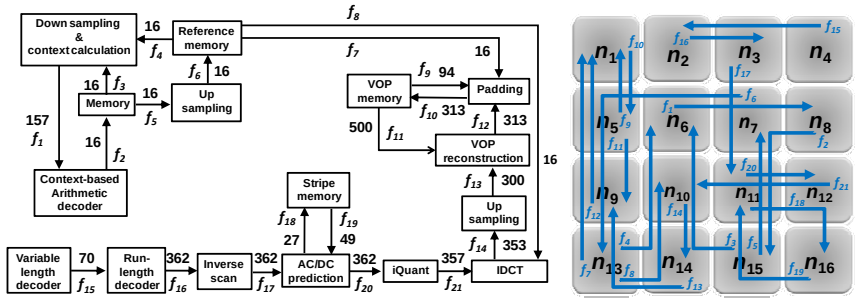


Fig. 12. VOPD Application

topology graph, routing matrix, and characteristics of routers. The outputs from the first sub-tool along with the set of system constraints will be inputs for the second part.

IX. Experimental Results

To evaluate the capability of our method, we applied it to a real-time multimedia application with a random mapping to the tiles of a 4×4 mesh on-chip network. Figure 12 shows the task graph and flow mapping of a Video Object Plane Decoder (VOPD) [33] in which each block corresponds to an IP and the numbers near the edges represent the bandwidth (in MBytes/sec) of the data transfer, for a 30 frames/sec MPEG-4 movie with 1920×1088 resolution [34]. There are 21 communication flows characterized by TSPEC.

Hence, each flow i is characterized by $(L_i, p_i, \sigma_i, \rho_i)$. We assume L_i and p_i for all flows are the same and equal to 1 $flit$ and 1 $flit/cycle$, respectively. ρ_i is determined in $flits/cycle$ due to associated bandwidth with flow f_i in Figure 12 and σ_i varies between 8 and 128 $flits$ for different flows. The length of a round in WRR scheduling, L_{WR}, is assumed to be 10 *cycles*.

A. Delay Optimization

As mentioned before, decision variables in the proposed optimization problems are the weights on shared channels. Due to shared channels in VOPD application, 20 weights are formulated in the optimizations as a weight vector W defined as below:

$$W = \big(w_{(6,3,0,4)}, w_{(10,2,0,0)}, w_{(14,0,0,2)}, w_{(13,3,0,2)}, w_{(12,0,0,2)}, w_{(9,4,0,0)}, w_{(4,3,0,4)}, w_{(4,0,0,2)}, w_{(8,2,0,0)}, w_{(8,4,0,2)}, w_{(6,2,0,4)}, w_{(10,4,0,0)}, w_{(14,3,0,2)}, w_{(13,1,0,2)}, w_{(12,3,0,2)}, w_{(9,3,0,0)}, w_{(4,0,0,4)}, w_{(4,4,0,2)}, w_{(8,4,0,0)}, w_{(8,0,0,2)}\big) \quad (21)$$

The "End-to-End Delay Program" calculates per-flow worst-case bounds as functions of weights for each flow in VOPD application and derives corresponding constraints. The "Optimization Program" formulates *Minimize-Delay* problem and derives weights for VOPD application.

To show how these weights affect the communication delay, we consider four different schemes:

- Random Scheme: The weights are selected randomly.
- *Round Robin Scheme*: The weights have the same values to represent round robin policy.
- *Optimized Scheme*: The weights are optimized based on the optimization problem (13).
- *Unoptimized Scheme*: The weights are not optimized and there are many unoptimized configurations. In this scheme, we allocate weights so as to maximize the optimization problem (13) instead of minimization.

Then, the total maximum delay are calculated for different schemes and depicted in Table II. From this table, we can see that the optimized scheme leads to about 15.4%, 48.8%, and 81.1% reduction in total maximum delay when compared with *Round Robin*, *Random*, and *Unoptimized* schemes, respectively. The results show that although WRR is able to make better performance in terms of latency than RR scheduling, if the weights are not allocated properly, it may be worse. Therefore, an appropriate weight configuration makes WRR able to reduce total and average maximum delay by balancing the allocation of shared network bandwidth to different traffic flows with respect to their specifications and contentions for

TABLE II
How Good Are Optimized Weights?

Scheme Type	Weight Vector	Total Worst-case Delay (*cycles*)	Average Worst-case Delay (*cycles*)
Optimized	(2, 8, 8, 2, 6, 6, 4, 2, 3, 6, 8, 2, 2, 8, 4, 4, 6, 8, 7, 4)	3671	174
Round Robin	(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	4237	202
Random	(1, 4, 2, 7, 2, 3, 9, 5, 8, 5, 9, 6, 8, 3, 8, 7, 1, 5, 2, 5)	7177	343
Unoptimized	(1, 1, 1, 9, 1, 1, 9, 9, 9, 5, 9, 9, 9, 1, 9, 9, 1, 1, 1, 5)	19432	926

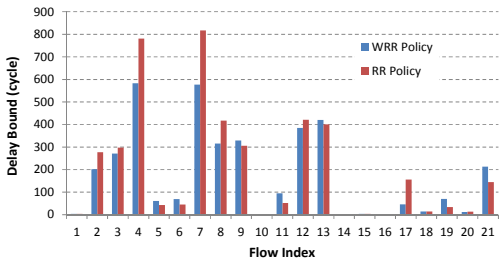


Fig. 13. Maximum worst-case delay for every flow

shared resources.

To better understand the effects of the weights, per-flow delay bounds for RR and WRR with *Optimized* scheme are revealed in Fig 13. This figure shows that flows in WRR can experience longer or shorter delays than RR scheme which depends on their assigned weights leading to different service levels (the amount of network bandwidth allocated to each flow). However, from Table II, we can see that the total and average worst-case delay are decreased in WRR with *Optimized* scheme because the weights are assigned in a way to minimize total delay, satisfy performance constraints, and reduce contentions for shared resources leaving room for other contention interfering flows. Therefore, WRR can be used to control the per-flow delay bound by controlling its assigned weight.

It is worth mentioning that if RR policy is better for flows due to the defined optimization problem, the solution method sets the weights equal to each other, namely makes WRR into RR.

B. Multi-objective Optimization

In the multi-objective optimization minimizing delay and variance, we have calculated two parameters: *Total Worst-case Delay* and *Variance* listed in Table III. As can be observed from Table III, Minimize-Delay problem guarantees that weight allocation is carried out in favor of minimizing total worst-case delay while there is no such guarantee for the sum of variances over various flows. In contrast, the Multi-objective optimization provides a trade-off between such parameters.

Although we have assumed the same importance for total delay and variance in the multi-objective problem by considering the same weighting coefficients in Equation (20), it is possible for designers to change the value of the weighting coefficients γ_1 and γ_2 to specify another relative importance of objective functions.

C. Comparing with Other Solution Methods

As a comparative study, we implement three other metaheuristics, namely *Pure Random Search (PRS)* [15], *Markov Monotonous Search (MMS)* [16], *Adaptive Search (AS)* [16] to compare them with the genetic algorithm in terms of runtime and efficiency. These algorithms belong to a category of metaheuristics called trajectory-based methods. A trajectory-based algorithm works on single solutions at any time, namely, it starts from an initial state (initial solution) and follows a trajectory to reach a successor solution which may or may not belong to the neighborhood of the current solution. Population-based metaheuristics, on the contrary, deal with a set (a population) of solutions in each iteration and in turn provide an intrinsic method for exploring the search space. The way of manipulating the population has a significant impact on the performance of these methods. Genetic algorithms belong to this category. We also extend *PRS*, *MMS*, and *AS* to support a population of solutions instead of a single solution. Hereby, they produce m solutions in every iteration and select n solutions for the next iteration. The extended versions of *PRS*, *MMS*, and *AS* are called *PRS* $(m+n)$, *MMS* $(m+n)$, and *AS* $(m+n)$. Table IV presents the iteration number and run time required for solving the optimization problem (Eq. 13).

The results show that all metaheuristics presented in this table obtain the same solution for the problem. Therefore, we can say with some confidence that the solution is of high quality.

The table shows that the genetic algorithm has a shorter execution time with fewer iterations. GA is no exhaustive optimization method. However, as it is well known that GAs provide an efficient and robust method for solving problems

TABLE III
HOW GOOD IS MULTIOBJECTIVE OPTIMIZATION?

	Weight Vector	Total Worst-case Delay (*cycles*)	Variance
Round Robin	$(5,5,5,5,5,5,5,5,5,5,$ $5,5,5,5,5,5,5,5,5,5)$	4237	59324.49
Minimize-Delay	$(2,8,8,2,6,6,4,2,3,6,$ $8,2,2,8,4,4,6,8,7,4)$	3671	35416.67
Multi-objective	$(6,9,9,1,6,7,2,1,1,6$ $4,1,1,9,4,3,8,1,1,6)$	4045	29320.71

TABLE IV
COMPARISON OF THE RUN TIME BETWEEN DIFFERENT METHODS

Optimal point obtained by the methods		
Optimal Weight Vector		Total Delay
$(2,8,8,2,6,6,4,2,3,6,8,2,2,8,4,4,6,8,7,4)$		3671 *cycles*
Performance in different methods		
	Iteration#	Time *(sec)*
PRS	$100,000$	2.71
MMS	$100,000$	2.8
AS	$100,000$	2.85
PRS $(10+10)$	$5,000$	13
MMS $(10+10)$	$5,000$	13.37
AS $(10+10)$	$5,000$	12.83
GA	250	1.05

in which the objective function is discontinuous, nondifferentiable, or highly nonlinear and due to the results from table IV, we believe that GA is a well suited solution method for our problem.

X. Conclusions

In this work, we have extended our proposed analytical methodology [4] for deriving per-flow delay bound under RR policy to WRR scheduling and then compared them. We have developed algorithms to automate analysis steps. It is notable that the proposed methodologies for both RR and WRR do not deal with the back-pressure, but we have calculated the buffer size thresholds to make sure the back-pressure does not occur in the network. Due to our proposed analytical models, we have presented two optimization problems for weight allocation in WRR scheduling, first one for minimizing total worst-case delays, second one for minimizing both total worst-case delays and their variance under performance requirements to control per-flow delay bound. We have also demonstrated that the proposed model exerts significant impact on communication performance. The algorithm for solving the proposed minimization problems runs very fast. For the case study, the optimized solution is found within about one second. In the future, we intend to investigate other scheduling policies. We also plan to extend the proposed analytical method in case of back-pressure in the network. Zhao and Lu [35] propose analytical models to derive worst-case bounds for constant bit rate flows due to back-pressure in the network.

References

[1] J. Y. L. Boudec and P. Thiran, *"Network Calculus: A Theory of Deterministic Queuing Systems for the Internet"*, Number 2050 in LNCS, 2004.

[2] F. Jafari, A. Jantsch, and Z. Lu, "Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling", *in Proc. the International Conference on Computer Design (ICCD)*, pp. 445-446, 2011.

[3] F. Jafari, A. Jantsch, Z. Lu, "Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling", *in Proc. Design, Automation and Test in Europe Conference (DATE)*, pp. 538-541, 2012.

[4] F. Jafari, Z. Lu, and A. Jantsch, "Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels", Submitted to ACM Transactions on Design Automation of Electronic Systems (TODAES).

[5] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", *ACM Comput. Surv.*, Vol. 35, No. 3, pp. 268-308 , 2003.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An architecture for differentiated services", *IETF RFC 2475*, 1998.

[7] J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, J.-Y. Le Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding", *IEEE/ACM Transactions on Networking* Vol. 10, No. 4, pp. 529-540, 2002.

[8] A. Charny and J.L. Boudec, "Delay Bounds in a Network with Aggregate Scheduling", *in Proc. QofIS*, pp.1-13, 2000,

[9] Y. Jiang, "Delay bounds for a network of guaranteed rate servers with FIFO aggregation", *Computer Networks*, Vol. 40, No. 6, pp. 683-694, 2002.

[10] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks", *Performance Evaluation*, Vol. 63, No. 9, pp. 956-987, 2006.

[11] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, "Buffer Optimization in network-on-Chip through Flow Regulation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 29, No. 12, pp 1973-1986, Dec. 2010.

[12] Y. Qian, Z. Lu, and Q. Dou, "QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin", *in Proc. the 28th International Conference on Computer Design (ICCD)*, pp. 52-59, 2010.

[13] J.D. Owens et al., "Research Challenges for On-Chip Interconnection Networks", IEEE Micro, Vol. 27, No. 5, 2007, pp. 96-108.

[14] D. P. Bertsekas, "Stochastic optimization problems with nondifferentiable cost functionals", *Journal of Optimization Theory and Applications*, Vol. 12, No. 2, pp. 218-231, 1973.

[15] S. H. Brooks, "A discussion of random methods for seeking maxima", *The computer journal*, Vol. 6, No. 2, 1958.

[16] R. White, "A survey of random methods for parameter optimization", *SIMULATION*, Vol. 17, pp. 197-205, 1971.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, New Series, Vol. 220, No. 4598, pp. 671-680, 1983.

[18] F. Glover, "Future paths for integer programming and links to artificial intelligence", *Computers and Operations Research*, Vol. 13, No. 5, pp. 533-549, 1986.

[19] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning", *Journal Physica D archive*, Vol. 2, No. 1-3, pp. 187-204, 1986.

[20] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, (first ed.) The MIT Press, 1992.

[21] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine learning, Studies in Computational Intelligence*, (first ed.) Addison-Wesley Longman Publishing Co., 1989.

[22] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[23] A. Walker, J. Hallam, D. Willshaw, "Bee-havior in a mobile robot: the construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment", *in Proc. IEEE International Conference on Neural Networks (ICNN)*, Vol. 3, IEEE Service Center, pp. 1451-1456, 1993.

[24] J. Kennedy, R. Eberhart, "Particle swarm optimization", *in Proc. IEEE International Conference on Neural Networks (ICNN)*, Vol. 4, pp. 1942–1948, 1995.

[25] R.M. Storn, K.V. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, Vol. 11, pp. 341-359, 1997.

[26] K.M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control", *IEEE Control Systems Magazine*, Vol. 22, pp. 52-67, 2002.

[27] D. Simon, "Biogeography-based optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 12, pp. 702-713, 2008.

[28] J. Wroclawski. The Use of RSVP with IETF Integrated Services, September 1997. RFC 2210, IETF.

[29] CA. Coello Coello, "A comprehensive survey of evolutionary based multiobjective optimization techniques", *Knowl Inform Syst: An Int J*, Vol. 3, pp. 269-308, 1999.

[30] S. Mardle and S. Pascoe, "An overview of genetic algorithms for the solution of optimisation problems", Computers in Higher Education Economics Review, Vol. 13, No. 1, 1999.

[31] P. Bajpai and M. Kumar, "Genetic Algorithm – an Approach to Solve Global Optimization Problems", Indian Journal of Computer Science and Engineering, Vol. 1 No. 3, pp. 199-206.

[32] J. Guan, M. M. Aral, "Progressive genetic algorithm for solution of optimization problems with nonlinear equality and inequality constraints", Applied Mathematical Modelling, Vol.23, No. 4, pp. 329–343, 1999.

[33] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip", *IEEE Transaction on Parallel and Distributed Systems*, VOL. 16, NO. 2, February 2005 113-129

[34] E.B. van der Tol and E.G. Jaspers, "Mapping of MPEG4 Decoding on a Flexible Architecture Platform", SPIE, Vol. 4674, 2002,pp. 1-13.

[35] X. Zhao and Z. Lu. "Per-flow Delay Bound Analysis Based on a Formalized Micro-architectural Model", *in Proc. ACM/IEEE International Symposium on Networks-on-Chip (NoCS'2013)*, Tempe Arizona, USA, April 2013.