In Networks on Chip - Theory and Practice, Editors: F. Gebali,
H. Elmiligi, M. W. El-Kharashi, CRC Press Taylor & Francis, 2009.

Resource Allocation for QoS On-Chip Communication

Axel Jantsch and Zhonghai Lu
Royal Institute of Technology, Stockholm, Sweden

June 27, 2009

1 Introduction

The provision of communication services with well defined performance characteristics has received significant attention in the NoC community because for many applications it is not sufficient or adequate to simply maximize average performance. It is envisioned that complex NoC based architectures will host complex, heterogeneous sets of applications. In a scenario where many applications compete for shared resources, a fair allocation policy that gives each application *sufficient* resources to meet its delay, jitter and throughput requirements, is critical. Each application, or each part of an application, should obtain exactly those resources needed to accomplish its task, not more nor less. If an application gets too small a share of the resources, it will either fail completely because of a critical deadline miss, or its utility will be degraded, e.g. due to bad video or audio quality. If an application gets more of a resource than needed, the system is over-dimensioned and not cost effective. Moreover, well defined performance characteristics are a prerequisite for efficient composition of components and subsystems into systems [9]. If all subsystems come with QoS properties the system performance can be statically analyzed and, most importantly, the impact of the composition on the performance of individual subsystems can be understood and limited. In the absence of QoS characteristics all subsystems have to be re-verified because the interference with other subsystems may severely affect a subsystems performance and even render it faulty. Thus, QoS is an enabling feature for compositionality.

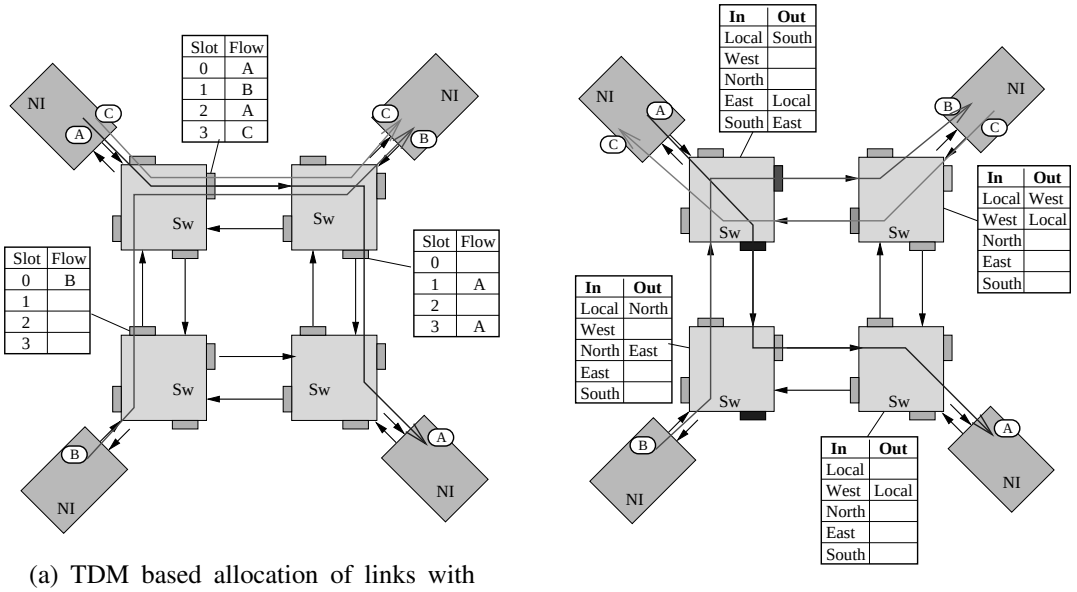
The topic of this chapter is resource allocation schemes that provide the shared NoC communication resources with well defined *Quality of Service (QoS)* characteristics. We exclusively deal with the performance characteristics delay, throughput and, to a lesser extent, delay variations (jitter).

We group the resource allocation techniques into three main categories. *Circuit switching*[1] allocates all necessary resources during the entire life time of a connection. Figure 1(b) illustrates this scheme. In every switch there is a table that defines the connections between input ports and output ports. The output port is exclusively reserved for packets from that particular input port. In this way all the necessary buffers and links are allocated for a connection between a specific

[1]Note, that some authors categorize TDM techniques as a circuit switching scheme. In this chapter we reserve the term circuit switching for the case when resources are allocated exclusively during the entire lifetime of a connection.

source and destination. Before a data packet can be sent, the complete connection has to be setup but once it is setup, the communication is very fast because all contention and stalling is avoided. The table can be implemented as an optimized hardware structure leading to a very compact and fast switch. However, setting up a new connection has a relatively high delay. Moreover, the setup delay is unpredictable because it is not guaranteed that a new connection can be setup at all. Circuit switching is justified only if a connection is stable over a long time and utilizes the resources to a very high degree. With few exceptions such as SoCBUS [19] and Crossroad [4] circuit switching has not widely been used in NoCs because only few applications justify the exclusive assignment of resources to individual connections. Also, the problem of predictable communication is not avoided but only moved from data communication time to circuit setup time. Furthermore, the achievable load of the network as a whole is limited in practice because a given set of circuits blocks the setup of new circuits although there are sufficient resources in the network as a whole.

In *Time Division Multiplexing (TDM)* resources are exclusively allocated to a specific user during well defined time periods. If a clock is available as a common time reference, clock cycles, or slots, are often used as allocation units. Figure 1(a) shows a typical TDM scheme where links are allocated to flows in specific time slots. The allocation is encoded in a slot allocation table with one table for each shared resource, a link in this example. The example assumes four different time slots. The tables are synchronized such that a flow, which has slot k in one switch gets slot $(k+1) \mod 4$ in the following switch, assuming it takes one cycle for a packet to traverse a switch.



(a) TDM based allocation of links with slot allocation tables.

(b) Circuit switching based resource allocation.

Figure 1: Resource allocation schemes based on TDM and circuit switching. (Sw=Switch, NI=Network Interface, A,B,C=traffic flows)

The example illustrates two drawbacks of TDM schemes. First, there is a trade-off between granularity of bandwidth allocation and table size. If we have more flows and need a finer granularity for bandwidth allocation, larger tables are required. Second, there is a direct relation between allocated bandwidth and maximum delay. If the bandwidth allocated is k/n with k out of n slots allocated, a packet has to wait $n/k - 1$ cycles in the worst case for the next slot to appear. This is

a problem for low delay, low throughput traffic because it either gets much more bandwidth than needed or its delay is very high.

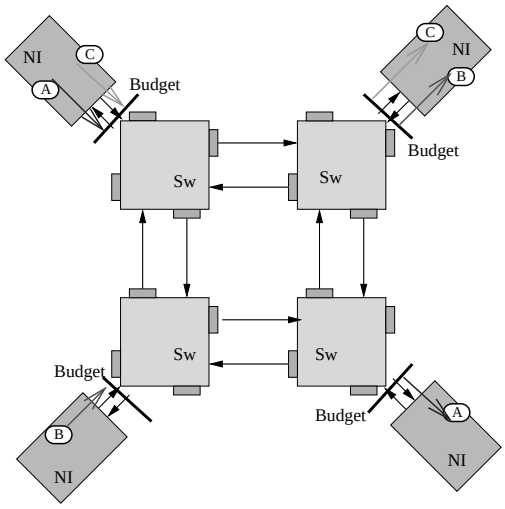


Figure 2: Aggregate resource allocation.

Aggregate resource allocation is a course grained and flexible allocation scheme. Figure 2 shows that each resource is assigned a traffic budget for both sent and received traffic. The reasoning is that, if all resources comply with their budget bounds, the network is not overloaded and can guarantee minimum bandwidth and maximum delay properties for all the flows. Traffic budgets can be defined per resource or per flow and they have to take into account the communication distance to correctly reflect the load in the network. Aggregate allocation schemes are flexible but provide looser delay bounds and require larger buffers than more fine grained control mechanisms such as TDM and circuit switching. This approach has been elaborated in [9] and suitable analysis techniques can be adapted from flow regulation theories in communication networks such as network calculus [10].

In the following sections we will discuss these main three groups of resource allocation in more detail. In section 5 we take up dynamic setup of connections and in section 6 we elaborate some aspects of priority schemes and fairness of resource allocation. Finally we we give an example of how to use a TDM resource allocation scheme in a complex telecom system.

2 Circuit Switching

Circuit switching means that all the necessary resources between a source node and a destination node are exclusively allocated to a particular connection during its entire lifetime. Thus, no arbitration is needed and packets never stall on the way. Consequently, circuit switching allows for very fast communication and small, low-power switches, *if the application is suitable*. In an established connection each packet experiences 1 cycle delay per switch in SoCBUS [19] and 3.48 ns in a 180nm implementation of a Crossroad switch [4]. Hence, the communication delay in established connections is both low and predictable. However, setting up a connection takes more time and is unpredictable. For SoCBUS the authors report a setup time of at least 5 cycles per switch.

Figure 3 shows many, but not all, resources needed for a circuit switched connection. We have the access port in the network interface (NI), up- and downstream buffers in the NI, buffers and crossbars in the switches and the links between the NIs and the switches. If the entire resource chain from the NI input port across the network to the NI output port is reserved exclusively for a specific connection, strong limitations are imposed on setting up other connections. For instance, in this scenario one source node can only have one single sending and one receiving connection at the same time.

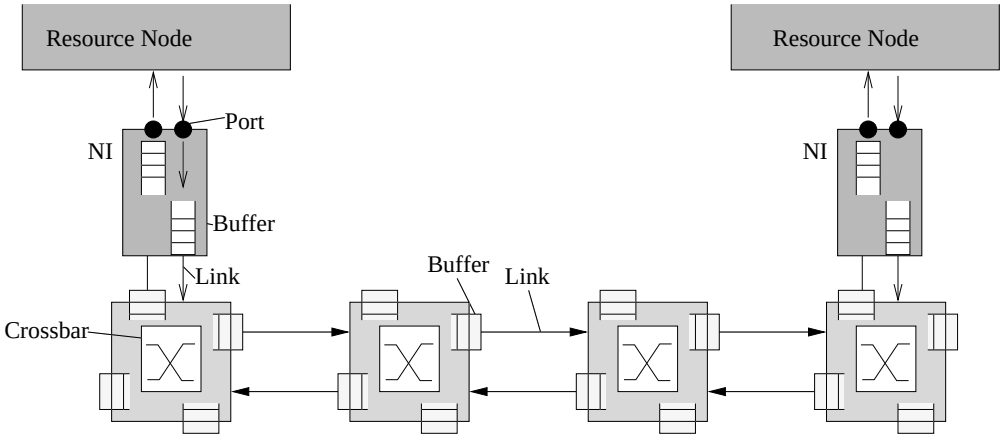


Figure 3: All the resources used for communication between a source and a destination can be allocated in different ways.

Figure 4 illustrates another aspect of the inflexibility of circuit switching. If the links attributed with in use labels are allocated to connections, no new communication from the entire left half of the network to the right half is possible. If these four connections live for a long time, they will completely block a large set of new connections, independent of the routing policy employed, even if they utilize only a tiny fraction of the network or link bandwidth. If restrictive routing algorithms such as deterministic dimension order routing are used, a few allocated links can already stall completely the communication between large parts of the system. For instance, if only one link, link A in figure 4, is used in both nodes connected to Sw 1 and Sw 2 will not be able to communicate to any of the nodes in the right half of the system under X-Y dimension order routing.

Consequently, neither the setup delay nor the unpredictability of the setup time is the most severe disadvantage of circuit switching when compared to other resource allocation schemes, because the connection setup problem is very similar in TDM based techniques (see section 5 for a discussion on circuit setup). The major drawback of circuit switching is its inflexibility and, from a QoS point of view, the limited options for selecting a particular QoS level. For a given source-destination pair the only choice is to setup a circuit switched connection which, once established, gives the minimal delay (1 cycle per hop $\times$ the number of hops in SoCBUS) and the full bandwidth. If an application requires many overlapping connections with moderate bandwidths demands and varying delay requirements, a circuit switched network has little to offer.

Thus, pure circuit switching allocation scheme can be used with benefit in the following two scenarios.

- If the application exhibits a well understood, fairly static communication pattern with a relatively small number of traffic streams with very high bandwidth requirements and long lifetime, these streams can be mapped on circuit switched connections in a cost efficient, power

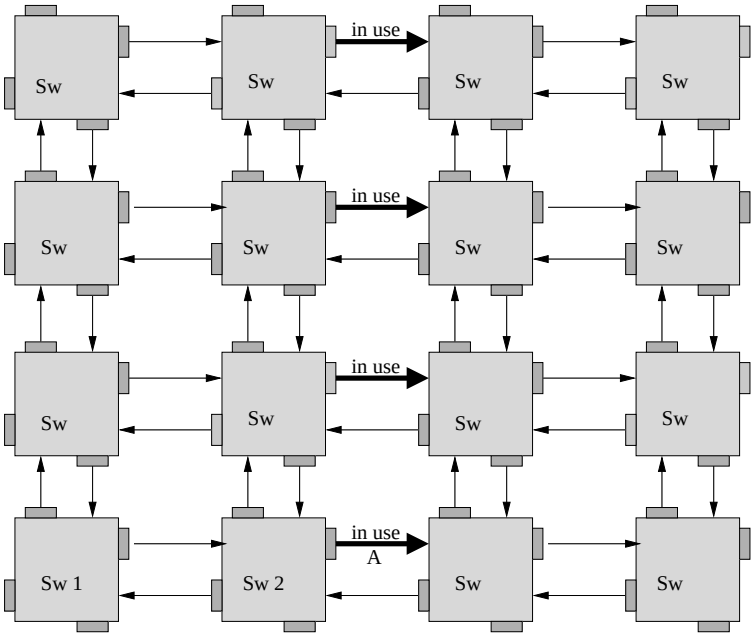


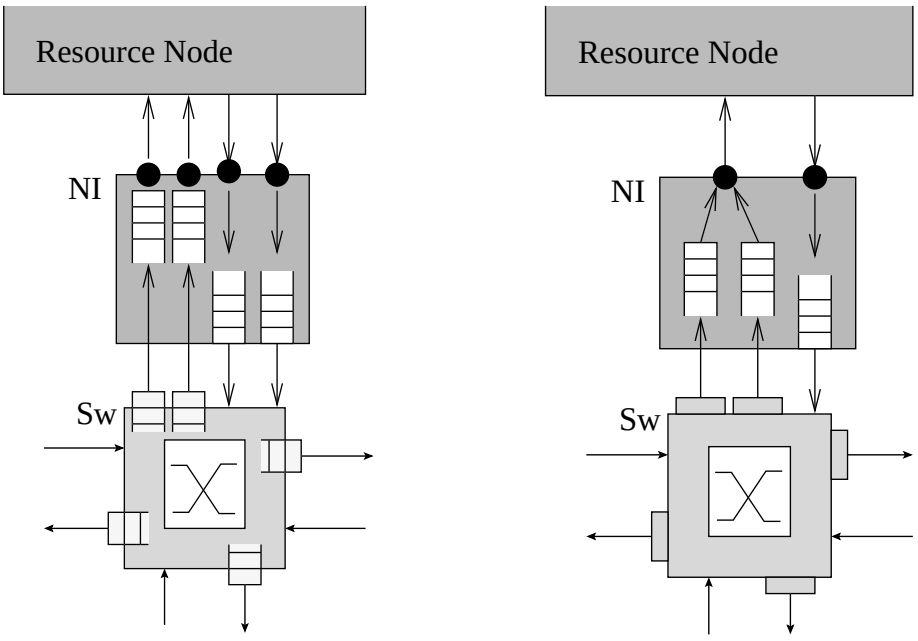
Figure 4: A few active connections may inhibit the setup of new connections although communication bandwidth is available.

efficient and low delay implementation as demonstrated in a study by Chung et al. [4].

- For networks with a small number of hops (up to two), connections can be quickly built up and torn down. The set-up overhead may be compensated by efficient data traversal even if the packet length is only a few words. Several proposals argue for circuit switching implementations based implicitly on this assumption [8, 12, 19]. But even for small sized networks we have the apparent trade-off between packet size and blocking time of resources. Longer packets decrease the relative overhead of connection set-up but block the establishment of other connections for longer time.

For large networks and applications with communications with various different QoS requirements that demand more flexibility in allocating resources, circuit switched techniques are only part of the solution at best.

This inflexibility of circuit switching can be addressed by duplicating some bottleneck resources. For instance, if the resources in the NI are duplicated, as shown in figure 5(a), each node can entertain two concurrent connections in each direction, which increases the overall utilization of the network. A study by Millberg et al. [15] has demonstrated that by duplicating the outgoing link capacity of the network, called Dual Packet Exit (figure 5(b)), the average delay is reduced by 30% and the worst case delay by 50%. Even though that study was not concerned with circuit switching, similar or higher gains are expected in circuit switched approaches. Leroy et al. [11] essentially propose to duplicate the links between switches in an approach they call *spatial division multiplexing (SDM)*. As illustrated in figure 6(a), parts of the link, i.e. subsets of its wires, can be allocated to different connections, thus relaxing the exclusivity of link allocation in other circuit switched schemes. The switch then becomes a sophisticated multiplexer structure that allows the routing of input wires to output wires in a highly flexible and configurable manner. Leroy et al. compare this method to a TDM approach and report 8% less energy consumption, 31% less area and 37% higher delay for their implementation of an SDM switch.



(a) Duplication of selected resources can increase the overall utilization of the network.

(b) Dual Packet Exit doubles the upstream buffers in the NI [15].

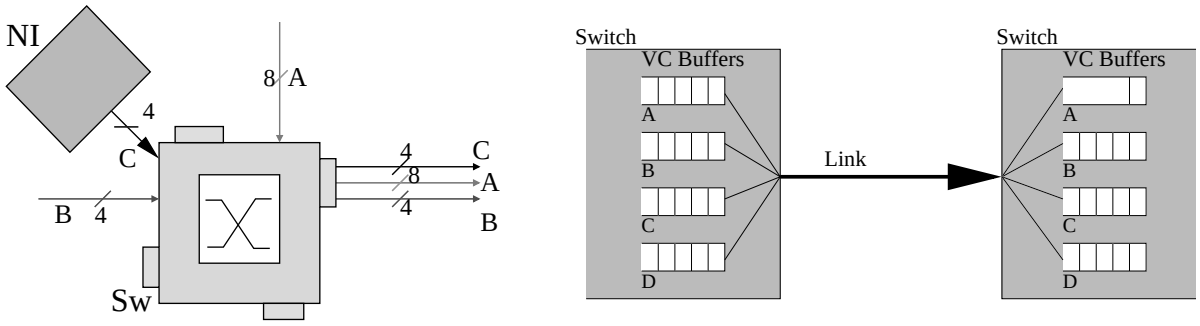
Figure 5: Duplication of NI resources.

Circuit switching can be combined with time sharing by exclusively reserving some resources while sharing others. Those resources that are exclusively allocated can be duplicated to combine maximum flexibility with short delays and efficient implementation. A good example for a mixed approach is the Mango NoC [1], which exclusively allocates buffers in a switch but allows to share the links between switches. In figure 6(b) the four buffers A-D are allocated exclusively to a connection but the link between the two switches is shared among them. Flits from the output buffers in the left hand switch access the link based on a mix of round robin and priority arbitration that allows to calculate predictable end to end delays for each flit.

This clever scheme decouples to some extent delay properties of a connection from throughput properties. The maximum delay of a flit is controlled by assigning priorities. The higher the priority of a packet, the lower the maximum waiting time in the VC buffer for accessing the link. But even low priority flits have a bounded waiting time because they can be stalled by at most one flit of each higher priority connection. Hence, a priority Q flit has to wait at most $Q \cdot F$, where F is the time for one flit to access the link and there are $Q - 1$ higher priority classes. Since there is no other arbitration in the switch, the end-to-end delay (not considering the network interfaces) is bounded by $(Q \cdot F + \Delta) \cdot h$, where Δ is the constant delay in the crossbar and the input buffer of the switch, and h is the number of hops.

The number of VCs determine the granularity of bandwidth allocation and the bandwidth allocated to a connection can be increased by assigning more VCs.

One drawback of this method is that a connection exclusively uses a resource, a VC buffer, and



(a) Spatial Division Multiplexing (SDM) assigns different wires to different connections on the links [11].

(b) A mixed circuit switched and time shared allocation scheme of the Mango NoC [1].

Figure 6: Duplication of switch resources.

to support many concurrent connections, many VCs are required. This drawback is inherited from the exclusive resource allocation of circuit switching but it is a limited problem here because it is confined to the VC buffers. Also, there is a trade-off between high granularity of bandwidth allocation and the number of VCs. But this example demonstrates nicely that the combination of different allocation schemes can offer significant benefits in terms of increased flexibility and QoS control at limited costs.

3 Time Division Multiplexing Virtual Circuits

Next we discuss a less strict reservation method which exclusively allocates a resource for specific connections only in individual time slots. In different time slots the resource is used by different connections. We use the *Time Division Multiplexing (TDM) Virtual Circuit (VC)* techniques developed in Ætherial [7] and Nostrum [16] as examples. For a systematic analysis of TDM properties we follow the theory of *Logic Networks* developed in [13, 14].

3.1 Operation and Properties of TDM VCs

In a network, we are concerned with two shared resources: buffers in switches and links (thus link bandwidth) between switches. The allocation for the two resources may be coupled or decoupled. In coupled buffer and link allocation, the allocation of buffers leads to associated link allocation. In decoupled buffer and link allocation, the allocation of buffers and that of links are independent. In this section, we consider the coupled buffer and link allocation using TDM. The consequence of applying the TDM technique to coupled buffer and link allocation is the reservation of exclusive slots in using both buffers and links. When packets pass these buffers along their routing path in reserved time slots, they encounter no contention, like going through a virtually dedicated circuit, called *Virtual Circuit (VC)*.

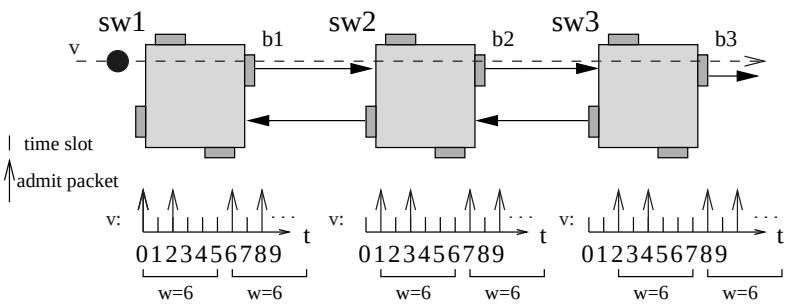


Figure 7: An example of packet delivery on a VC

On one hand, to guarantee a portion of link bandwidth for a traffic flow, the exclusive share of buffers and links must be reserved before actual packet delivery. On the other hand, traffic on a VC must be admitted into the network in a disciplined way. A certain number of packets are admitted in pre-calculated slots within a given window. This forms an *admission pattern* that is repeated without change throughout the lifetime of the VC. We call the window *admission cycle*. In the network, VC packets synchronously advance one hop per time slot. Since VC packets encounter no contention, they never stall, using consecutive slots in consecutive switches. As illustrated in Figure 7, VC v passes switches sw_1, sw_2 and sw_3 through $\{b_1 \rightarrow b_2 \rightarrow b_3\}$. On v, two packets are injected into the network every six slots (we say that the window size is 6). Initially, the slots of buffer b_1 at the first switch sw_1 occupied by the packet flow are 0 and 2. Afterwards, this pattern repeats, b_1's slots 6 and 8, 12 and 14, and so on are taken. In the second switch sw_2, the packets occupy b_2's slots 1 and 3, 7 and 9, and 13 and 15, and so on. In switch sw_3, they occupy b_3's slots 2 and 4, 8 and 10, and 14 and 16, and so on.

As such, TDM VC makes two assumptions: (1) network switches share the same notion of time. They have the same clock frequency but may allow phase difference [17]; (2) buffer and link allocations are coupled, as stated previously. Since packets are transmitted over these shared resources without stall and in a time-division fashion, we need only one buffer per link. This buffer may be situated at the input or output of a switch. As can be observed in Figure 7, we have assumed the buffer is located at the output. In terms of QoS, TDM VC provides strict guarantees in delay and bandwidth with low cost. Compared with circuit switching, it utilizes resources in a shared fashion (but with exclusive time slots), thus more efficient. As with circuit switching, it must be established before communication can start. The establishment can be accomplished through configuring a routing table in switches. Routing for VC packets is performed by looking up these tables to find the output port along the VC path.

Before discussing VC configuration, we introduce two representative TDM VCs proposed for on-chip networks, the Æthereal VC [7] and the Nostrum VC [16].

3.2 On-Chip TDM VCs

Figure 8 shows two VCs, v_1 and v_2, and the respective routing tables for the switches. The output links of a switch are associated with a buffer or register. A routing table (t, in, out) is equivalent to a routing or slot allocation function $\mathcal{R}(t, in) = out$, where t is time slot, in an input link, and out an output link. v_1 passes switches sw_1 and sw_2 through $\{b_1 \rightarrow b_2\}$; v_2 passes switches sw_3 and sw_2 through $\{b_3 \rightarrow b_2\}$. The Æthereal NoC [7] proposes this type of VC for QoS. Since the path of such a VC is not a loop, we call it an open-ended VC.

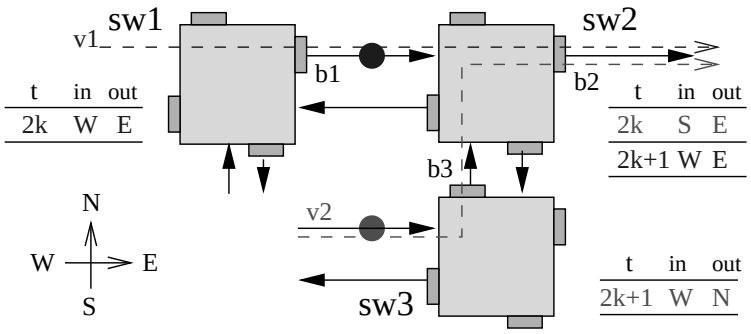


Figure 8: Open-ended virtual circuits

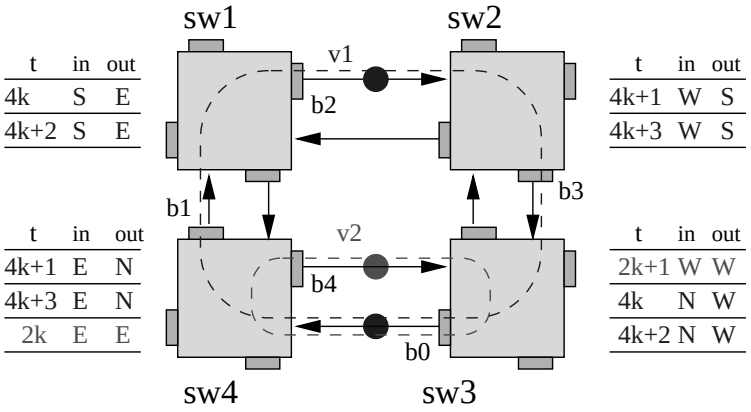


Figure 9: Closed-loop virtual circuits

The Nostrum NoC [16] also suggests TDM VC for QoS. However, a Nostrum VC has a cyclic path, i.e., a closed loop. On the loop, at least one *container* is rotated. A container is a *special packet* used to carry data packets, like a vehicle carrying passengers. The reason to have a loop is due to the fact that Nostrum uses deflection routing [3] whereas switches have no buffer queues. If a packet arrives at a switch, it must be switched out using one output port the next cycle. A Nostrum switch has $k+1$ inports/outports, k of which are network ports connected to other switches and one of which is a local duplex port for admitting/sinking packets into/from the network. If k network packets arrive at a switch but none of them has reached its destination, none of the packets will be sunk and they will occupy all the k network output ports the next cycle. This situation makes any packet admission at this time impossible because there is no output port available. This problem is solved by a looped container. The looped container ensures that there is always an output port or link available for locally admitting a VC packet into the container and thus the network. VC packets are loaded into the container from a source, and copied (for multicast) or unloaded at the destination, by-passing other switches. Similarly to open-ended VCs, containers as VC packet carriers enjoy higher priority than best-effort packets and must not contend with each other.

3.3 TDM VC Configuration

As introduced above, TDM VC requires an establishment phase to configure routing tables. The entries for the routing tables are globally orchestrated such that no simultaneous use of shared

buffers and links is possible, i.e., free from contention. This process is called TDM VC configuration. We can loosely formulate the problem as follows: *Given a specification set of n VCs, each with a set of source and destination nodes and minimum bandwidth requirement, determine visiting nodes in sequence for each VC and exact time slots when VC packets visit each node.* Note that here we only use bandwidth as constraint in the formulation, but apparently, other design constraints such as delay, jitter and power can be added into the formulation, if needed. Also, we do not include a cost function as an optimization criterion.

VC configuration is a complex problem, which can be elaborated as two sequential but orthogonal sub-problems:

1. *Path selection*: Since a network is rich in connectivity, given a set of source and destination nodes [2], there exist diverse ways to traverse all nodes in the set. Minimal routes are typically preferable. However, in some scenarios, non-minimal routes are also useful to balance traffic and have the potential to enable more VCs configurable. Allowing non-minimal routes further complicates the problem. In both cases, we need to explore the network path diversity. This involves an exponentially increased search space. Suppose each VC has m alternative paths, configuring n VC routes has a search space of m^n.
2. *Slot allocation*: Since VC packets can not contend with each other, VCs must be configured such that an output link of a switch is allocated to one VC per slot. Again, to find optimal slot allocation, i.e., reserving sufficient but not more than necessary bandwidth, requires exploring an exponentially increased design space.

VC configuration is typically an iterative process, starting with the path selection, slot allocation and then repeating the two sequential steps until a termination condition is reached such that solutions are found, or solutions cannot be found within a time threshold. VC configuration is a combinatorial optimization problem. Depending on the problem size, one can use different techniques to solve it, discovering whether there exists any, optimal or suboptimal solutions. If the problem size is small, standard search techniques such as branch-and-bound, backtracking may be used to constructively search the solution space, finding optimal or suboptimal solutions; If the problem size is large, heuristics such as dynamic programming, randomized algorithms, simulated annealing and genetic algorithms may be employed to find suboptimal solutions within reasonable time.

No matter what search techniques we use, any solution must satisfy three conditions: (1) all VCs are contention free; (2) all VCs are allocated sufficient slots in a time wheel, thus guaranteeing sufficient bandwidth; (3) the network must be deadlock free and livelock free. Condition (3) is important for networks with both best-effort and TDM VC traffic. The pre-allocated bandwidth must leave room to route best-effort traffic without deadlock and livelock. For example, a critical link should not have all its bandwidth reserved, making it unusable for best-effort packets. In the following sections, we focus on (1) and (2), discussing how to guarantee contention free and allocate sufficient bandwidth. This discussion is based on the logical network theory [13, 14]. The theory is equally suited for open and closed VCs; in the following we use closed VCs to illustrate the concepts.

[2] We allow that a VC may comprise more than one source and one destination node.

3.4 The Theory of Logical Network for TDM VCs

One key of success for synchronous hardware design is that we are able to reason about the logic behavior for each signal at each and every cycle. In such a way, the logic design is fully deterministic. Traffic flows delivered on TDM VCs exhibit well-defined synchronous behavior, fully pipelined and moving one hop per cycle through pre-allocated resources. To precisely and collectively express the resources used or reserved by a guaranteed service flow, we define a *logical network (LN)* as an infinite set of associated *(time slot, buffer)* pairs with respect to a buffer on the flow path. This buffer is called *reference buffer*. When VCs overlap, we use a shared buffer as the reference buffer because it is the point of interest to avoid contention. Since LNs use exclusive sets of resources, VCs allocated to different LNs are free from conflict.

LNs may be constructed in two steps, *slot partitioning* and *slot mapping*. Slot partitioning is performed in the time domain while slot mapping in the space domain. We exemplify the LN construction using Figure 9, where two VCs v_1 and v_2 are to be configured. The loop length of v_1 is 4 and a container re-visits the same buffer every 4 cycles. Assuming uniform link bandwidth 1 packet/cycle, the bandwidth granularity of v_1 is $1/4$ packet/cycle and the packet admission cycle on v_1 is 4. Since two containers are launched, v_1 offers a bandwidth of $1/2$ packet/cycle. Similarly, v_2 with one container supports bandwidth of $1/2$ packet/cycle, and the packet flow on v_2 has an admission cycle of 2.

1. *Slot partitioning*: As b_0 is the only shared buffer of v_1 and v_2, $v_1 \cap v_2 = \{b_0\}$, we use b_0 as the reference buffer, denoted as $Ref(v_1, v_2) = b_0$. Since v_1 and v_2 use b_0 once every two slots, their bandwidth equals to $1/2$. Thus, we partition the slots of b_0 into two sets, an even set $s_0^2(b_0)$ for $t = 2k$ and an odd set $s_1^2(b_0)$ for $t = 2k + 1$, as highlighted in Figure 10 by the underlined number set $\{0, 2, 4, 6, 8, \cdots\}$ and $\{1, 3, 5, 7, 9, \cdots\}$, respectively. The notation $s_\tau^T(b)$ represents pairs $(\tau + kT, b)$, which is the τth slot set of the total T slot sets, $\forall k \in \mathbb{N}$, $\tau \in [0, T)$ and $T \in \mathbb{N}$. The pair (t, b) refers to the slot of b at time instant t. Notation $s^T_{\tau_1,\tau_2,\cdots,\tau_n}(b)$ collectively represents a set of pair sets $\{(\tau_1 + kT, b), (\tau_2 + kT, b), \cdots, (\tau_n + kT, b)\}$.

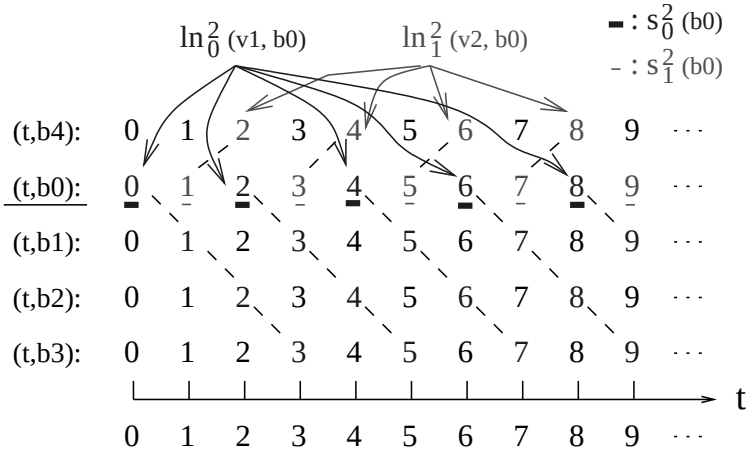


Figure 10: LN construction by partitioning and mapping slots in the time and space domain

2. *Slot mapping*: The partitioned slot sets can be mapped to slot sets of other buffers on a VC regularly and unambiguously because a VC packet or container advances one hop along its path each and every slot. For example, v_1 packets holding slot t at buffer b_0, i.e., pair (t, b_0),

will consecutively take slot $t+1$ at b_1 (pair $(t+1, b_1)$), slot $t+2$ at b_2 (pair $(t+2, b_2)$), and slot $t+3$ at b_3 (pair $(t+3, b_3)$). In this way, the slot partitionings are propagated to other buffers on the VC. In Figure 10, after mapping the slot set $s_0^2(b_0)$ on v_1 and $s_1^2(b_0)$ on v_2, we obtain two sets of slot sets $\{s_0^2(b_0), s_1^2(b_1), s_0^2(b_2), s_1^2(b_3)\}$ and $\{s_1^2(b_0), s_0^2(b_4)\}$, as marked by the dashed diagonal lines. We refer to the logically networked slot sets in a set of buffers of a VC as a *LN*. Thus a LN is a composition of associated *(time slot, buffer)* pairs on a VC with respect to a buffer. We denote the two LNs as $ln_0^2(v_1, b_0)$ and $ln_1^2(v_2, b_0)$, respectively. The notation $ln_\tau^T(v, b)$ represents the τth LN of the total T LNs on v with respect to b. Figure 10 illustrates the mapped slot sets for $s_0^2(b_0)$ and $s_1^2(b_0)$ and the resulting LNs. We may also observe that, slot mapping is a process of assigning VCs to LNs. LNs can be viewed as the result of VC assignment to slot sets, and a LN is a function of a VC. In our case, v_1 subscribes to $ln_0^2(v_1, b_0)$ and v_2 to $ln_1^2(v_2, b_0)$.

As $ln_0^2(v_1, b_0) \cap ln_1^2(v_2, b_0) = \emptyset$, v_1 and v_2 are conflict free. In addition, the bandwidth supply of both VCs equals to $1/2$ packet/cycle, $BW(ln_0^2(v_1, b_0)) = BW(ln_1^2(v_2, b_0)) = 1/2$ packet/cycle.

Suppose that v_1 and v_2 are two overlapping VCs with D_1 and D_2 being their admission cycles, respectively, we have proved a set of important theorems, which we summarize as follows:

- The maximum number N_{ln} of LNs, which v_1 and v_2 can subscribe to without conflict, equals to $GCD(D_1, D_2)$, the Greatest Common Divisor (GCD) of D_1 and D_2. The bandwidth that a LN possesses equals $1/N_{ln}$ packet/cycle.
- Assigning v_1 and v_2 to different LNs is the sufficient and necessary condition to avoid conflict between them.
- If v_1 and v_2 have multiple (more than one) shared buffers, these buffers must satisfy *reference consistency* in order to be free from conflict. If so, any of the shared buffers can be used as the reference buffer to construct LNs. Two shared buffers b_1 and b_2 are termed *consistent* if it is true that, "v_1 and v_2 packets do not conflict in buffer b_1" if and only if "v_1 and v_2 packets do not conflict in buffer b_2". The sufficient and necessary condition for them to be consistent is that the distances of b_1 and b_2 along the two VCs, denoted $d_{b_1\vec{b}_2}(v_1)$ and $d_{b_1\vec{b}_2}(v_2)$, respectively, satisfy $d_{b_1\vec{b}_2}(v_1) - d_{b_1\vec{b}_2}(v_2) = kN_{ln},\ k \in \mathbb{Z}$. Furthermore, instead of pair-wise checking, the reference consistency can be linearly checked.

3.5 Application of the Logical Network Theory for TDM VCs

Guided by the LN theory, slot allocation is a procedure of computing and consuming LNs, by which VCs are assigned to different LNs. To begin this procedure, we need to know the admission cycle and bandwidth demand of VCs. We draw a diagram for allocating slots for two VCs in Figure 11. It has three main steps:

Step 1: Reference consistency check. If two VCs have more than one shared buffer, this step checks whether they are consistent.

Step 2: Compute available LNs. The available LNs (thus bandwidth) are computed to check if they can satisfy VC's bandwidth requirement.

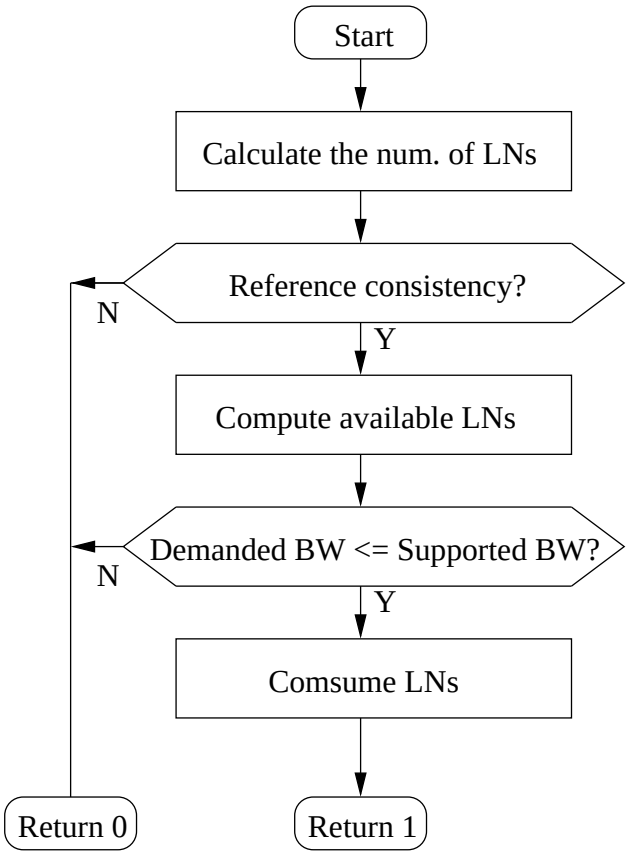


Figure 11: LN-oriented slot allocation

Step 3: Consume LNs. This allocates and claims the exclusive slots (slot sets).

As an example in Figure 12, two VCs v_1 and v_2 comprise a buffer set, $\{b_1, b_2, b_5, b_6\}$ and $\{b_2, b_3, b_4, b_6\}$, respectively, and their bandwidth demand is $BW(v_1) = 3/8$ and $BW(v_2) = 7/16$. Following the procedure, the slot allocation is conducted as follows:

Step 1: The number N_{ln} of LNs for v_1 and v_2 equals to $N_{ln}(v_1, v_2) = GCD(8,\ 16) = 8$. Since $v_1 \cap v_2 = \{b_2, b_6\}$, the distance between b_2 and b_6 along v_1 is $d_{b_2\vec{b}_6}(v_1) = 2$. Similarly, the distance between b_2 and b_6 along v_2 is $d_{b_2\vec{b}_6}(v_2) = 2$. Thus the difference between the two distances is $d_{b_2\vec{b}_6}(v_1, v_2) = d_{b_2\vec{b}_6}(v_1) - d_{b_2\vec{b}_6}(v_2) = 0$. Therefore, b_2 and b_6 satisfy reference consistency. We can select either of them as the reference buffer, say, b_2.

Step 2 and 3: We can start with either of the two VCs, say v_1. Referring to b_2, the slot set to consider is $\{0, 1, 2, 3, 4, 5, 6, 7\}$. Each of the elements in the set represents a LN with bandwidth supply $1/8$ packet/cycle. Initially all slots are available. Since $BW(v_1) = 3/8$, we allocate slots $\{0, 2, 4\}$ to v_1. The remaining slot set is $\{1, 3, 5, 6, 7\}$, providing a bandwidth of $5/8$. Apparently, this can satisfy the bandwidth requirement of v_2. We allocate slots $\{1, 3, 5, 7\}$ to v_2. The resulting slot allocation meets both VCs' bandwidth demand and packets on both VCs are contention free.

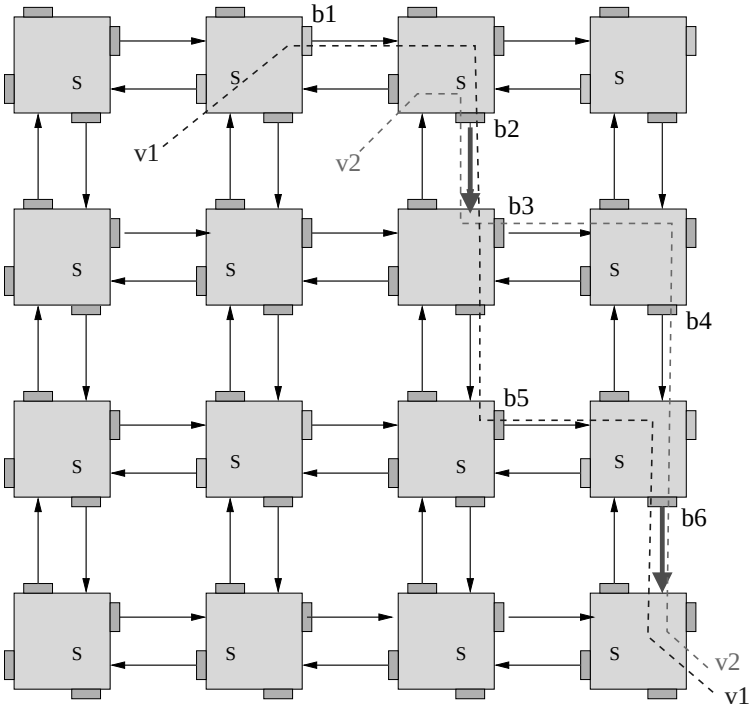


Figure 12: An example of LN-oriented slot allocation

The LN-based theory provides us a formal method to conduct slot allocation. The application of this theory to open-ended VCs is straightforward, as we have shown. Applying this theory to closed-loop VCs is also simple. In this case, the admission cycle for VCs is predetermined by the length of VC loops. While this approach opens a new angle for slot allocation avoiding ad hoc treatment of this problem, the complexity of optimal slot allocation still remains. In this regard, the key questions include how to make the right bandwidth granularity by scaling admission cycles without jeopardizing application requirements, and how to consume LNs in order to leave room for VCs that remain to be configured.

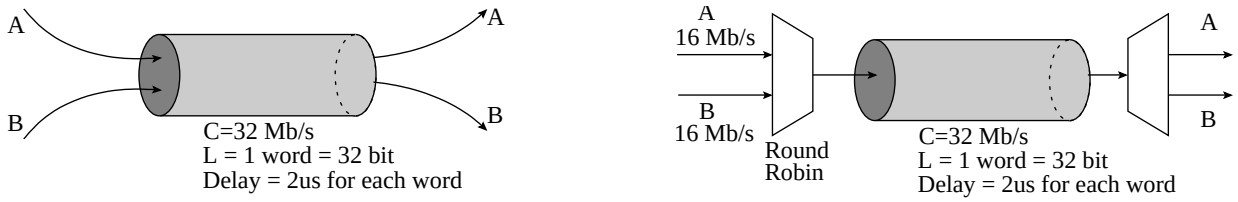
4 Aggregate Resource Allocation

In TDM and circuit switching approaches resources are exclusively allocated to connections either for short time slots (TDM) or during the entire lifetime of the connection (circuit switching). This results in precise knowledge of delay and throughput of a connection independent of other activities in the network. Once a connection is established, data communication in other connections cannot interfere or influence the performance. This exclusive allocation of resources results in strong isolation properties which is ideal for QoS guarantees. The disadvantage is the potential underutilization of resources. A resource (buffer, link) that is exclusively allocated to a connection, cannot be used by other connections even when it is idle. Dynamic and adaptive reallocation of resources for the benefit of overall performance is not possible. Thus, both TDM and circuit switching schemes are ideally suited for well known, regular, long-living traffic streams that require strong QoS guarantees. They are wasteful for dynamic, rapidly changing traffic pattern for which the occasional miss of a deadline is tolerable.

Resource planning with TDM and circuit switching schemes will optimize for the worst case situation and will provide strong upper bounds for delay and lower bounds for throughput. The average case delay and throughput is typically very close or even identical to the worst case. If a connection is setup based on circuit switching, the delay of every packet through the network is the same and worst case delay is the same as average case delay. For TDM connections the worst case delay occurs when a packet has just missed its time slot and has to wait one full period for the next time slot. On average this waiting time will be only half the worst case waiting time, but the delivery time through the network is identical for all packets. Hence, the average case delay will be lower but close to the worst case.

We can relax the assumption of exclusive resource ownership to allow for dynamic and more adaptive allocation of resources while still being able to provide QoS guarantees. *Aggregate resource allocation* assigns a resource to a group of users (connections or processing elements) and dynamically arbitrates the requests for resource usage. By constraining the volume of traffic injected by users and by employing proper arbitration policies we can guarantee worst case bounds on delay and throughput while maximizing resource utilization and, thus, average performance.

4.1 Aggregate Allocation of a channel



(a) One channel is allocated to two flows.

(b) The channel access is arbitrated with a Round Robin policy.

Figure 13: Shared channel.

Consider the situation in figure 13(a). Two flows share a single channel. We know the capacity of the channel (32 Mb/s) and the delay (2 μs for 32 bits which are transferred in parallel). However, in order to give bounds on delay and throughput for the two individual flows we need to know

1. Characteristics of the flows;
2. Arbitration policy for channel access.

The flows have to be characterized, e.g. in terms of their average traffic rate and their burstiness. The latter is important because a flow with low average rate and unlimited burst size can incur an unlimited delay on its own packets and, depending on the isolation properties of the arbiter, on the other flow as well. The arbitration policy has to be known because it determines how much the two flows influence each other. Figure 13(b) shows the situation where each flow has an average rate of 16 Mb/s and the channel access is controlled by a round robin arbiter. Assuming a fixed word length of L in both flows, round robin arbitration means that each flow gets at least a 50% of the channel bandwidth, which is 16 Mb/s. 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.

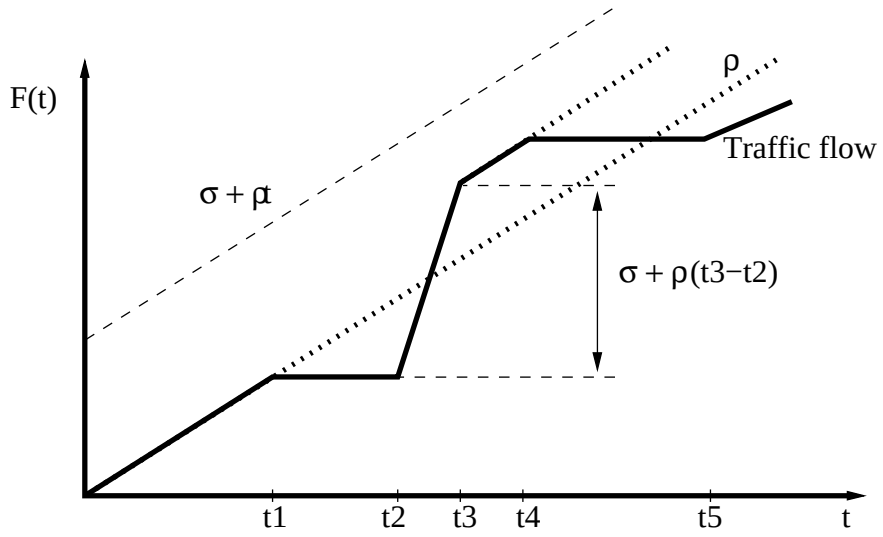


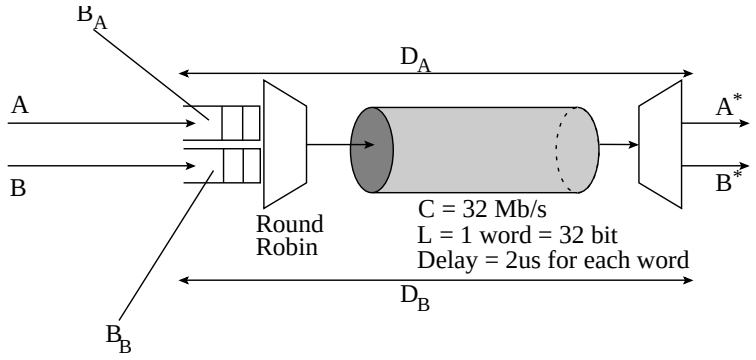
Figure 14: A (σ, ρ)-regulated flow.

In order to derive an upper bound on the delay for each flow, we have to know the maximum burst size. There are many ways to characterise burstiness of flows. We use a simple, yet powerful traffic volume based flow model from network calculus [5, 10]. In this model a traffic flow can be characterized by a pair of numbers (σ, ρ), where σ is the burstiness constraint and ρ is the average bit rate. Call $F(t)$ the total volume of traffic in bits on the flow in the period $[0, t]$. Then a flow is (σ, ρ)-regulated if

$$F(t_2) - F(t_1) \leq \sigma + \rho(t_2 - t_1)$$

for all time intervals $[t_1, t_2]$ with $0 \leq t_1 \leq t_2$. Hence, in any period the number of bits moving in the flow cannot exceed the average bit rate by more than σ. This concept is illustrated in figure 14 where the solid line shows a flow, which is constrained by the function $\sigma + \rho t$.

We use this notation in our shared channel example and we model a round robin arbiter as a *rate-latency server* [10] that serves each input flow with a minimum rate of $C/2$ after a maximum initial delay of L/C, assuming a constant word length of L in both flows. Then, based on the network calculus theory we can compute the maximum delay and backlog on flow A ($\bar{D}_A$, $\bar{B}_A$) and flow B ($\bar{D}_B$, $\bar{B}_B$), and the characteristics of the output flows A^* and B^*, as shown in figure 15. Due to the limited space we do not derive these formulas here (see [10] for detailed derivation and motivation), but we can make several observations. The delay in each flow consists of three components. The first two are due to arbitration and the last one, 2 μs, is the channel delay. The term L/C is the worst case time it takes for a word in one flow to get access to the channel if there are no other words in the same flow queued up before the arbiter. The second term, $2\sigma/C$, is the delay of a worst case burst. The formula for the maximum backlog also consists of two terms, one due to the worst case arbitration time ($\rho L/C$) and the other due to bursts (σ). The rates of the output flows are unchanged as it is to be expected but the burstiness increases due to the variable channel access delay in the arbiter. It can be seen in the formulas that delay and backlog



$$
\begin{aligned}
A &\sim (\sigma_A, \rho_A) \\
B &\sim (\sigma_B, \rho_B) \\
\bar{B}_A &= \sigma_A + \rho_A L/C \\
\bar{B}_B &= \sigma_B + \rho_B L/C \\
\bar{D}_A &= L/C + 2\sigma_A/C + 2\mu\text{sec} \\
\bar{D}_B &= L/C + 2\sigma_B/C + 2\mu\text{sec} \\
A^* &\sim (\sigma_A + \rho_A L/C, \rho_A) \\
B^* &\sim (\sigma_B + \rho_B L/C, \rho_B) \\
\\
\rho_A &\leq 0.5C \\
\rho_B &\leq 0.5C
\end{aligned}
$$

Figure 15: The shared channel serves two regulated flows A and B with round robin arbitration.

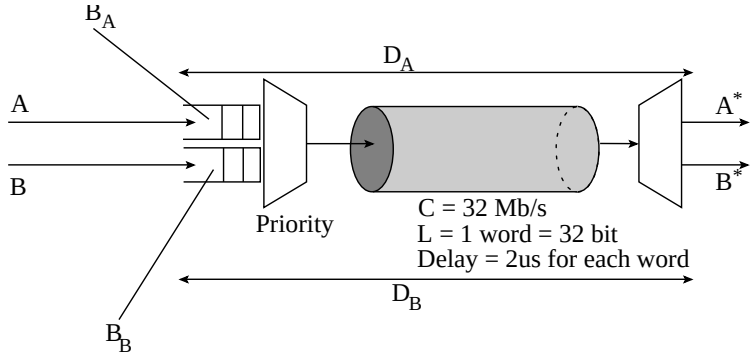
bounds and the output flow characteristics of each flow do not depend on the characteristics of the other flow. This demonstrates the strong isolation of the round robin arbiter that in the worst case always offers half the channel bandwidth to each flow. However, the average delay and backlog values of one flow do depend on the actual behavior of the other flow, because if one flow does not use its maximum share of the channel bandwidth ($0.5C$), the arbiter allows the other flow to use it. This dynamic reallocation of bandwidth will increase average performance and channel utilization. However, note that these formulas are only valid under the given assumptions, i.e. the average flows of both rates must be lower than 50% of the channel bandwidth. If one flow has a higher average rate, its worst case backlog and delay are unbounded.

Table 1: Maximum Delay, backlog and output flow characteristics for a round robin arbitration. Delays are in μs, rates are in Mb/s, and backlog and delay values are rounded up to full 32bit words.

A	B	A			B		
(σ_A, ρ_A)	(σ_B, ρ_B)	$\bar{B}_A$	$\bar{D}_A$	$(\sigma_{A^*}, \rho_{A^*})$	$\bar{B}_B$	$\bar{D}_B$	$(\sigma_{B^*}, \rho_{B^*})$
$(0, 16.00)$	$(0, 16.00)$	32	3	$(32, 16.00)$	32	3	$(32, 16.00)$
$(0, 12.80)$	$(0, 12.80)$	32	3	$(32, 12.80)$	32	3	$(32, 12.80)$
$(0, 9.60)$	$(0, 16.00)$	32	3	$(32, 9.60)$	32	3	$(32, 16.00)$
$(0, 6.40)$	$(0, 16.00)$	32	3	$(32, 6.40)$	32	3	$(32, 16.00)$
$(0, 3.20)$	$(0, 16.00)$	32	3	$(32, 3.20)$	32	3	$(32, 16.00)$
$(0, 16.00)$	$(0, 16.00)$	32	3	$(32, 16.00)$	32	3	$(32, 16.00)$
$(32, 16.00)$	$(0, 16.00)$	64	5	$(64, 16.00)$	32	3	$(32, 16.00)$
$(64, 16.00)$	$(0, 16.00)$	96	7	$(96, 16.00)$	32	3	$(32, 16.00)$
$(128, 16.00)$	$(0, 16.00)$	160	11	$(160, 16.00)$	32	3	$(32, 16.00)$
$(256, 16.00)$	$(0, 16.00)$	288	19	$(288, 16.00)$	32	3	$(32, 16.00)$

Table 1 shows how the delay and backlog bounds depend on input rates and burstiness. In the upper half of the table both flows have no burstiness but the rate of flow A is varying. It can be

seen that flow B is not influenced at all and for flow A only the output rate changes but delay and backlog bounds are not affected because, as long as the flow does not request more than 50% of the channel bandwidth (16Mb/s), both backlog and delay in the arbiter is only caused by the arbitration granularity of one word. In the lower part of the table the burstiness of flow A is steadily increased. This affects the backlog bound, the delay bound and the output flow characteristics of A. However, flow B is not affected at all which underscores the isolation property of round robin arbitration under the given constraints.



$$
\begin{aligned}
A &\sim (\sigma_A, \rho_A) \\
B &\sim (\sigma_B, \rho_B) \\
\bar{B}_A &= \sigma_A + \rho_A L/C \\
\bar{B}_B &= \sigma_B + \rho_B \sigma_A/(C - \rho_A) \\
\bar{D}_A &= (L + \sigma_A)/C + 2\mu\text{sec} \\
\bar{D}_B &= (\sigma_A + \sigma_B)/(C - \rho_A) \\
&\quad +2\mu\text{sec} \\
A^* &\sim (\sigma_A + \rho_A L/C, \rho_A) \\
B^* &\sim (\sigma_B + \rho_B \sigma_A/(C - \rho_A), \\
&\quad \rho_B)
\end{aligned}
$$

$$\rho_A + \rho_B \leq C$$

Figure 16: The shared channel serves two regulated flows A and B with a priority arbitration.

Table 2: Maximum Delay, backlog and output flow characteristics for a an arbitration giving A higher priority. Delays are in μs, rates are in Mb/s, and backlog and delay values are rounded up to full 32bit words.

A	B	A			B		
(σ_A, ρ_A)	(σ_B, ρ_B)	$\bar{B}_A$	$\bar{D}_A$	$(\sigma_{A^*}, \rho_{A^*})$	$\bar{B}_B$	$\bar{D}_B$	$(\sigma_{B^*}, \rho_{B^*})$
$(0, 16.00)$	$(0, 16.00)$	32	3	$(32, 16.00)$	32	4	$(32, 16.00)$
$(0, 12.80)$	$(0, 12.80)$	32	3	$(32, 12.80)$	32	4	$(32, 12.80)$
$(0, 9.60)$	$(0, 16.00)$	32	3	$(32, 9.60)$	32	3	$(32, 16.00)$
$(0, 6.40)$	$(0, 16.00)$	32	3	$(32, 6.40)$	32	3	$(32, 16.00)$
$(0, 3.20)$	$(0, 16.00)$	32	3	$(32, 3.20)$	32	3	$(32, 16.00)$
$(0, 16.00)$	$(0, 16.00)$	32	3	$(32, 16.00)$	32	4	$(32, 16.00)$
$(32, 16.00)$	$(0, 16.00)$	64	4	$(64, 16.00)$	32	4	$(32, 16.00)$
$(64, 16.00)$	$(0, 16.00)$	96	5	$(96, 16.00)$	64	6	$(64, 16.00)$
$(128, 16.00)$	$(0, 16.00)$	160	7	$(160, 16.00)$	128	10	$(128, 16.00)$
$(256, 16.00)$	$(0, 16.00)$	288	11	$(288, 16.00)$	256	18	$(256, 16.00)$

To illustrate the importance of the arbitration policy on QoS parameters and the isolation of flows, we present priority based arbitration as another example. Figure 16 shows the same situation but the arbiter gives higher priority to flow A, but a flow B word cannot be preempted. If a flow B word has obtained channel access, an arriving flow A word has to wait until the complete flow B word is emitted into the channel. As it can be seen from the formulas in figure 16, flow A is

served at full channel bandwidth C, but the service rate of flow B is dependent on flow A and it is $C - \rho_A$. The bounds and output characteristics of flow A are entirely independent of flow B. In fact, flow B is almost invisible to flow A because the full channel capacity is available to flow A if requested. The only delay incurred by flow B is L/C if a flow B word has been granted access and is allowed to complete transmission. On the other hand, the bounds and output characteristics of flow B depend heavily on σ_A and ρ_A. This impact is quantitatively shown in table 2.

To summarize this example, the round robin policy offers a fair access to the channel and provides very good isolation properties such that the performance of one flow can be analyzed independently of the behavior of the other flow. Priority based arbitration offers better QoS figures for one flow at the expense of the other. Its isolation properties are weaker because the performance of the low priority flow can only be analyzed if the characteristics of the high priority flow is known. An extensive analysis and comparison of different service disciplines has been presented by Hui Zhang [20]. An analysis of some arbitration policies in the network calculus framework is elaborated by LeBoudec [10].

4.2 Aggregate Allocation of a Network

In a network each connection needs a chain of resources. To perform aggregate resource allocation for entire connections, we can do it for each individual resource and then compute performance properties for the entire connection. Network calculus is a suitable framework for this approach because it allows to derive tighter bounds for sequences of resources than what is possible by simply adding up the worst cases of individual resources. This feature is known as *pay bursts only once*. While this is a feasible and promising approach, we illustrate here an alternative technique that views the entire network as a resource to derive QoS properties. It has been elaborated in the context of the Nostrum NoC [9], which we take as an example in the following.

Each processing element in the network is assigned a traffic budget for both incoming and outgoing traffic. The amount of traffic in the network is bounded by these node budgets. As a consequence, the network exhibits predictable performance which can be used to compute QoS characteristics for each connection.

The Nostrum NoC has mesh topology and a deflection routing algorithm, which means that packets that loose competition for a resource are not buffered but deflected to a non-ideal direction. Hence, packets that are deflected take non-minimal paths through the network. A connection h between a sender **A** and a receiver **B** loads the network with

$$E_h = n_h d_h \delta \tag{1}$$

where n_h is the number of packets **A** injects into the network during a given window W, d_h is the shortest distance between **A** and **B**, and δ is the *average deflection factor*. It expresses the average amount of deflections a packet experiences and is defined as

$$\delta = \frac{\text{sum of traveling time of all packets in cycles}}{\text{sum of shortest path of all packets in cycles}}.$$

δ is load dependent and, as we will see below, the network load has to be limited in order to bound δ. Call H_r^o and H_r^i the sets of all outgoing and incoming connections of node r, respectively. We assign traffic budgets for each node as follows.

$$\sum_{h \in H_r^o} E_h \leq B_r^o \tag{2}$$

$$\sum_{h \in H_r^i} E_h \leq B_r^i \tag{3}$$

$$\sum_r B_r^o = \sum_r B_r^i \leq \kappa C_{\text{Net}} \tag{4}$$

B_r^o and B_r^i constitute the traffic budgets for each node r and C_{Net} is the total communication capacity of the network during the time window $W.\kappa$, with $0 \leq \kappa \leq 1$, is called the *traffic ceiling*. It is an empirical constant that has to be set properly to bound the deflection factor δ. A node is allowed to set up a new connection as long as the constraints (2) and (3) are met. In return every connection is characterized by the following bandwidth, average delay, maximum delay bounds as derived in [9],

$$\text{BW}_h = \frac{n_h}{W} \tag{5}$$

$$\text{maxLat}_h = 5DN \tag{6}$$

$$\text{avgLat}_h = d_h \delta \tag{7}$$

where D is the diameter of the network and N the number of nodes.

Thus, to summarize, by constraining the traffic for the whole network (by κ), for each resource node (B_r^i, B_r^o) and for each connection ($\frac{n_h}{W}$), QoS characteristics ((5), (6), (7)) are obtained. But note the dependency of the deflection factor δ on the traffic ceiling κ, for which a closed analytic formula is not known for a deflective routing network with its complex, adaptive behavior. In [9] D_1 is suggested as upper bound for δ. D_1 is the delay bound for 90% of the packets under uniformly distributed traffic. It can be determined empirically and has been found to be fairly tight bound when the network is only lightly loaded. When the network is operated close to its saturation point, the bound is much looser. However, the important point is, that D_1 has been found to be an upper bound for δ even for a large number of different traffic patterns. Hence, it can serve as an empirical upper bound under a wide range of conditions. Table 3 shows a given traffic ceiling κ and the measured corresponding D_1 for a range of network sizes. The *sat.* entries mean that the network is saturated and delays are unbounded.

This approach of using the entire network as an aggregate resource that is managed by controlling the incoming traffic gives very loose worst case bounds. The worst case bounds on maximum delay and minimum bandwidth are conservative and are always honored. In contrast, the given average delay may be violated. It is an upper bound in the sense that in most cases the observed average delay is below, but it is not guaranteed because it is possible to construct traffic patterns that violate this bound.

Table 3: (κ, D_1) pairs for various network sizes N and emission budgets per cycle B_r^o/W.

B_r^o/W	16	30	50	70	100
	(κ, D_1)	(κ, D_1)	(κ, D_1)	(κ, D_1)	(κ, D_1)
0.05	$(0.04, 1.12)$	$(0.06, 1.12)$	$(0.07, 1.15)$	$(0.08, 1.16)$	$(0.09, 1.11)$
0.10	$(0.09, 1.12)$	$(0.11, 1.15)$	$(0.14, 1.23)$	$(0.16, 1.23)$	$(0.19, 1.23)$
0.15	$(0.13, 1.12)$	$(0.17, 1.30)$	$(0.21, 1.41)$	$(0.24, 1.35)$	$(0.28, 1.35)$
0.20	$(0.18, 1.36)$	$(0.22, 1.40)$	$(0.27, 1.46)$	$(0.32, 1.46)$	$(0.37, 1.55)$
0.25	$(0.22, 1.44)$	$(0.28, 1.45)$	$(0.34, 1.64)$	$(0.40, 1.80)$	$(0.46, sat.)$
0.30	$(0.27, 1.44)$	$(0.34, 1.61)$	$(0.41, 4.65)$	$(0.48, sat.)$	$(0.56, sat.)$
0.35	$(0.31, 1.60)$	$(0.39, 1.72)$	$(0.48, sat.)$	$(0.55, sat.)$	$(0.65, sat.)$
0.40	$(0.36, 1.60)$	$(0.45, 6.10)$	$(0.55, sat.)$	$(0.63, sat.)$	$(0.74, sat.)$
0.45	$(0.40, 1.80)$	$(0.50, sat.)$	$(0.62, sat.)$	$(0.71, sat.)$	$(0.83, sat.)$
0.50	$(0.44, 6.17)$	$(0.56, sat.)$	$(0.69, sat.)$	$(0.79, sat.)$	$(0.93, sat.)$

This approach, while giving loose worst case bounds, optimizes the average performance, because all network resources are adaptively allocated to the traffic that needs them. It is also cost effective because in the network there is overhead for reserving resources and no sophisticated scheduling algorithm for setting up connections is required. The budget regulation at the network entry can be implemented cost efficiently and the decision for setting up a new connection can be taken quickly based on locally available information. However, the check if the *receiving node* has sufficient incoming traffic capacity is more time consuming because it requires a communication and acknowledgment across the network.

5 Dynamic Connection Setup

The reservation of resources to provide performance guarantees poses a dilemma. Once all resources are allocated it is straight forward to calculate the delay of a packet from a sender to a receiver. However, the setup time to establish a new connection may well be subject to arbitrary delay and unbounded. Hence, the emission time of the first packet in a connection cannot be part of the QoS guarantees. This feature of pushing the uncertainty of delays from the communication of an individual packet to the setup of a connection is in common to all three resource allocation classes discussed above. TDM, circuit switching and aggregate resource allocation schemes all have to setup a connection first, to be able to provide QoS guarantees for established connections. We have three possibilities to deal with this problem.

1. Setup of static connections at design time.
2. Limit the duration of connections to bound the setup time for a staticly defined set of connections.
3. Accept unbounded setup time.

Alternative (1), to allow only staticly defined connections is acceptable only for a certain class of applications which exhibit a well known and static traffic pattern. For more dynamic applications this option is either too limiting or too wasteful.

Alternative (2) is a compromise between (1) and (3). It defines statically at design time a set of connections. Each connection is characterized and assigned a maximum traffic volume and lifetime. Since the maximum lifetime of all connections is known, the setup time for a new connection is bounded and becomes part of the QoS characteristics of a connection. Only a small subset of all connections is active concurrently at any time and each connection competes for getting access to the network. For this process we can use many of the same techniques such as scheduling, arbitration, priority schemes, preemption, etc. as we use for individual packet transmission. Thus the QoS parameters would describe a two level hierarchy: (a) the worst case setup time, the minimum frequency of network access and minimum lifetime of a connection; (b) the worst case delay and minimum bandwidth for packets belonging to established connections.

Alternative (3) is acceptable for many applications. For instance a multimedia supporting user device or a telecom switch may simply refuse new requests if the system is already overloaded. Every finite resource system can only handle a finite number of applications and tasks, thus it is only natural and unavoidable to sometimes reject new requests.

In the following we briefly discuss connection setup in the context of circuit switching. TDM based connections are established in essentially the same way. In aggregate resource allocation schemes the setup may work similar as well but, depending on the schemes, some problems may not appear or be posed differently.

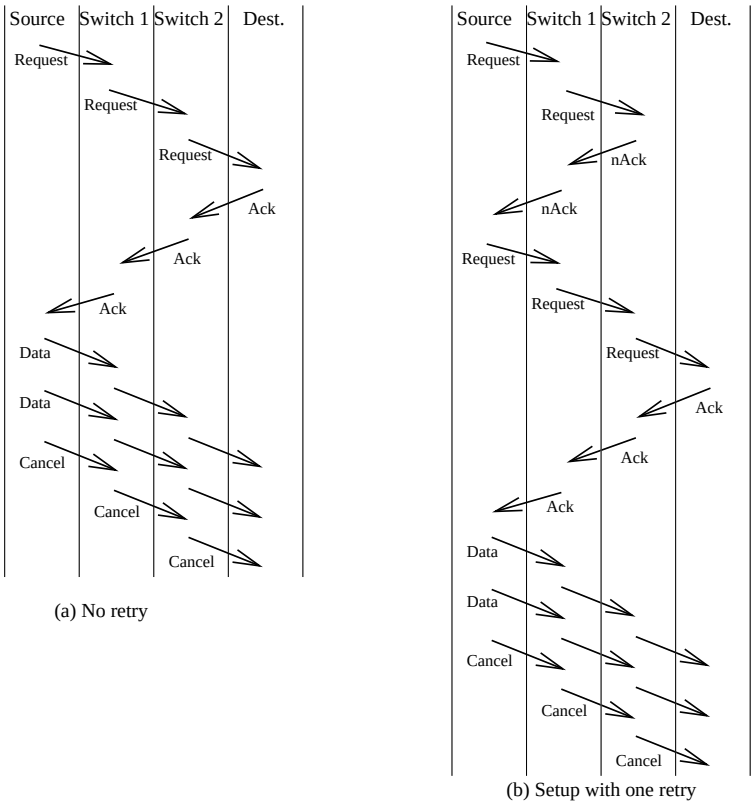


Figure 17: The three phases of circuit switched communication in SoCBUS [19].

If circuit switched connections are configured dynamically, as in SoCBUS [19] and Crossroad [4], the connection is setup by transmitting a special request signal or packet from the source node to the destination node along the intended route of the connection. This is illustrated in

figure 17 for a connection extending over two intermediate switches. If the connection is built successfully, the destination node responds by returning an acknowledgment packet (Ack) to the source. Once the source node has received the acknowledgment, it sends data packets. Since the delivery of data packets along the active connection is guaranteed, no acknowledgment is required and data transmission can proceed at very high speed. When the last data packet is emitted by the source node, it tears-down the connection by sending a cancel packet which releases all reserved resources of the connection.

Figure 17(b) illustrates the case when a requested resource is not available because it is in use by another connection, e.g. the link between switch 2 and the destination node. In this situation there are two main possibilities. First, the request packet waits in switch 2 until the requested resource (the link) is free. When the link becomes available, the request packet proceeds further with building the connection until it arrives at the destination node. The second alternative, which is shown in the figure, is to tear-down the partially built-up connection with a negative acknowledgment (nAck) packet. The main disadvantage of the first approach is the occurrence of deadlocks. Two or more partially built-up connections could end up in a cyclic dependency, each of them waiting indefinitely for a resource that one of the others blocks. A second disadvantage of the waiting approach is that a partially built-up connection may potentially block resources for a very long time without using them and prohibiting other connections to use them as well. Consequently, both SoCBUS and Crossroad tear-down partially built-up connections when a requested resource is not available. After some delay the source node makes a new attempt.

In principal the connection setup can use any routing policy, deterministic or adaptive, minimal or non-minimal path routing. For instance, the SoCBUS designers have implemented two different routing algorithms, source based routing and minimal adaptive routing [18]. In source based routing the path is determined by the source node and included in the request packet. The source node is free to adopt any routing policy; it could choose different routes when a setup attempt has failed. In the second routing algorithm of SoCBUS local routing decisions are taken by the switches. If a preferred output port of the switch is blocked, the switch will select another output port, which still has to be on the shortest path to the destination. The candidate output ports lying on a minimal path are tried in a round robin fashion. Source based routing is deterministic and reduces the complexity of the router at the cost of the network interface. The minimal adaptive routing algorithm increases the complexity of the router but is able to use local load information for the routing decision. It results in a higher delay in the router but it may find a path in some situations where the deterministic source based routing fails.

6 Priority and Fairness

By focusing on a resource allocation perspective, we have not illuminated several other issues related to QoS. For instance, *priority schemes* are often used to control QoS levels. QNoC, proposed by Bolotin et al. [2], groups all traffic in four categories and assigns different priorities. The four traffic classes are *Signaling*, *Real-Time*, *Read/Write* and *Block-Transfer* with *Signaling* having highest priority and *Block-Transfer* lowest. It can be observed that the *Signaling* traffic, which is characterized by low bandwidth and delay requirements, enjoys a very good QoS level without having a strong, adverse impact on other traffic. Since *Signaling* traffic is rare, its preferential treatment does not diminish the average delay of other, high throughput traffic too much.

In general, a priority scheme allows to control the access to a resource, which is not exclu-

sively reserved. Hence, it is an arbitration technique in aggregate allocation schemes. Its effect is to decrease the delay of one traffic class at the expense of other traffic, and it makes all low priority traffic invisible to high priority traffic. To compute the delay and throughput bounds of a traffic class we only have to consider traffic of the same and higher priority.[3] We have seen this phenomenon in section 4, figure 16, where the high priority flow could command the entire channel capacity. However, if we know that the high priority flow A uses only a small fraction of the channel bandwidth, say $\rho_A \leq 0.05C$, even flow B will be served very well. This knowledge of application traffic characteristics is utilized in QNoC and most other priority schemes leading to cost efficient implementations with good QoS levels.

In summary we note, that hard bounds on delay and bandwidth can only be given if the rate and burstiness of all higher priority traffic is constrained and known. Priority schemes work best with a relatively small number of priority levels (2-8) and if well characterized, low throughput traffic is assigned to high priority levels.

All arbitration policies should feature a certain *fairness* of access to a shared resource. Which notion of fairness to apply is however less obvious. Local versus global fairness is a case in point, illustrated in figure 18. Packets of connection A are subject to three arbitration points. At

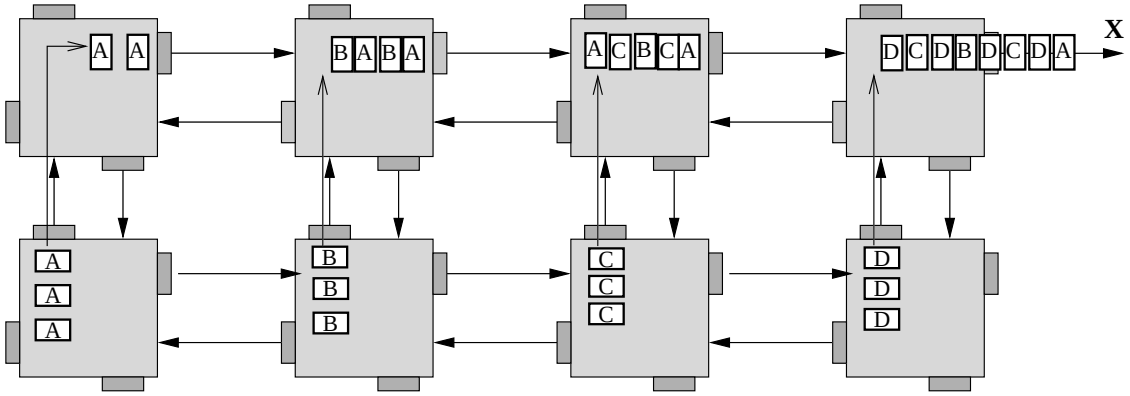


Figure 18: Local fairness may be very unfair globally.

each point a round robin arbiter is fair to both connections. However, at channel **X** connection D occupies four times the bandwidth and experiences 1/4 of the delay as connection A. This example shows that if only local fairness is considered, the number of arbitration points that a connection meets have a big impact on its performance since its assigned bandwidth drops by a factor two at each arbitration point. Consequently, in multi stage networks often age based fairness or priority schemes are used. This can be implemented with a counter in the packet header that is set to zero when the packet enters the network and is incremented in every cycle. For instance, Nostrum uses an age based arbitration scheme to guarantee the maximum delay bound given sin section 4, equation (6).

Another potential negative effect of ill conceived fairness is shown in figure 19. Assume we have two messages A and B, each consisting of 10 packets each. Assume further that the delay of the message is determined by the delay of the last packet. Packets of messages A and B compete for channel **X**. If they are arbitrated fairly in a round robin fashion, they occupy the channel alternatively. Assume it takes one cycle to cross channel **X**. If a packet A gets first access, the last A

[3]For the sake of simplicity we ignore priority inversion. *Priority inversion* is a time period when a high priority packet waits for a low priority packet. This period is typically limited and known such as L/C in figure 15.

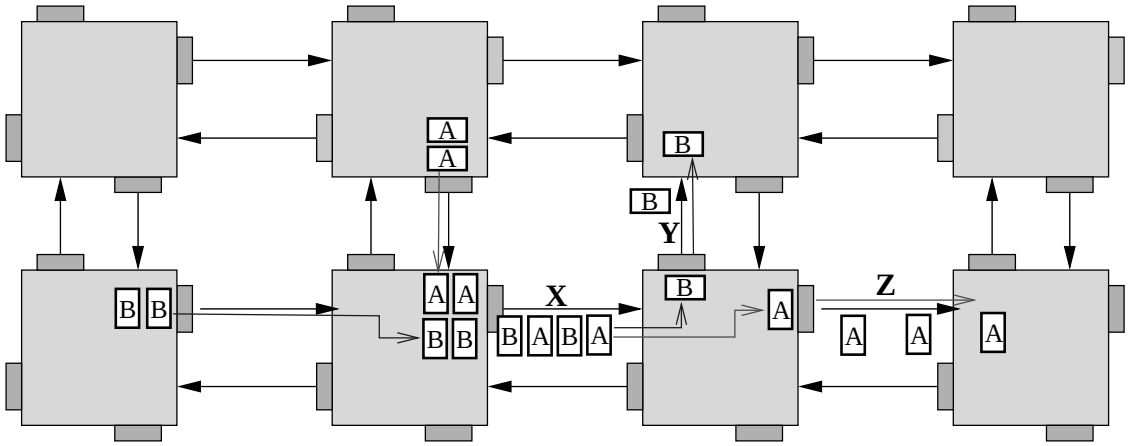


Figure 19: Local fairness may lead to lower performance.

packet will have crossed the channel after 19 cycles and the last B packet after 20 cycles. If we opt for an alternative strategy and assign the channel **X** exclusively to message A, all A packets will have crossed the channel after 10 cycles while all B packets will still need 20 cycles. Thus, a *winner-takes-it-all* arbitration policy would decrease the delay of message A by half without adversely affecting the delay of message B. Moreover, if the buffers are exclusively reserved for a message, both messages will block their buffers for a shorter time period compared to the fair round robin arbitration.

These examples illustrate that fairness issues require attention and the effects of arbitration policies on global fairness and performance are not always obvious. For a complete trade-off analysis the cost of implementation also has to be taken into account. For a discussion of implementation circuits, their size and delay, that realize different arbitration policies see [6, chapter 18].

7 QoS in A Telecom Application

To illustrate the usage of QoS communication, we present a case study on applying TDM VCs to an industrial application provided by Ericsson Radio Systems [14].

7.1 An Industrial Application

As mapped onto a 4×4 mesh NoC in Figure 20, this application is a radio system consisting of 16 IPs. Specifically, n_2, n_3, n_6, n_9, n_{10} and n_{11} are ASICs; n_4, 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 start-up, sets up and controls resources in normal operation. Traffic to/from n_1 is for system initial configuration and no longer used afterwards. The mesh has 48 duplex links with uniform link capacity. There are 26 node-to-node traffic flows that are categorized into 11 types of traffic flows {**a**, **b**, **c**, **d**, **e**, **f**, **g**, **h**, **i**, **j**, **k**}, as marked in the figure. The traffic flows are associated with a bandwidth requirement. In the example, **a** and **h** are multi-cast traffic, and others are unicast

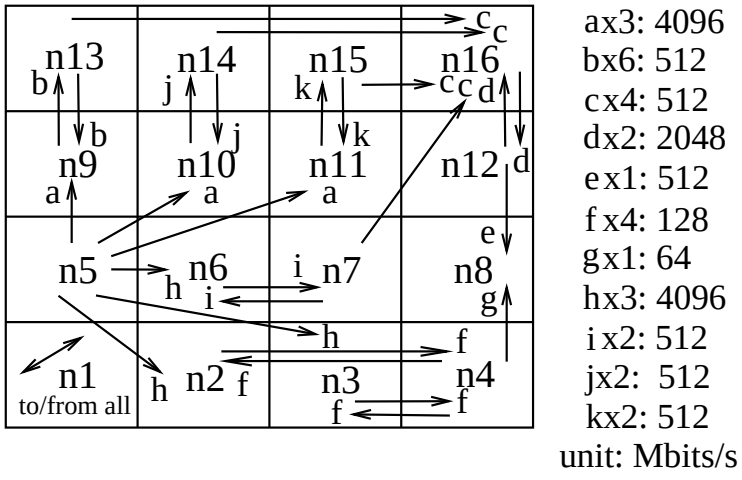


Figure 20: Node-to-node traffic flows for a radio system

traffic. As the application requires strict bandwidth guarantees for processing traffic streams, we use TDM VCs to serve the traffic flows. In this case study, we use *closed-loop VCs*.

The case study comprises two phases: *VC specification* and *VC configuration*. The VC specification phase defines a set of source and destination (sink) nodes and normalized bandwidth demand for each VC. The VC configuration phase constructs VC implementations satisfying VCs' specification requirement, one VC implementation for one VC specification. In this case study, a VC implementation is a looped TDM VC. Note that, a VC specification only consists of source and destination nodes, while its corresponding VC implementation consists of the source and destination nodes plus intermediate visiting nodes.

7.2 VC Specification

VC spec.	Traffic	BW (Mbits/s)	Num. of node-to-node flows	Source and sink nodes	BW demand
1	a	4096	3	n5, n9, n10, n11	1
2	b	512	2	n9, n13	1/8
3	c	512	4	n7, n13, n14, n15, n16	1/2
4	d	2048	2	n12, n16	1/2
5	e	512	1	n8, n12	1/8
6	f	128	4	n2, n3, n4	1/16
7	g	64	1	n4, n8, n4	1/64
8	h	4096	3	n5, n6, n2, n3	1
9	i	512	2	n6, n7	1/8
10	j	512	2	n10, n14	1/8
11	k	512	2	n11, n15	1/8

Table 4: VC specification for traffic flows

The VC specification phase consists of three steps: *determining link capacity*, *merging traffic flows*, and *normalizing VC bandwidth demand*.

We first determine the minimum required link capacity by identifying a critical (heaviest loaded) link. The most heavily loaded link may be the link directing from n_5 to n_9. The **a**-type traffic passes it and $bw_a = 4096$ Mbits/s. To support bw_a, link bandwidth bw_{link} must be not less than 4096 Mbits/s. We choose the minimum 4096 Mbits/s for bw_{link}. This is an initial estimation and subject to adjustment and optimization, if necessary.

Since the VC path search space increases exponentially with the number of VCs, reducing the number of VCs when building a VC specification set is crucial. In our case, we intend to define 11 VCs for the 11 types of traffic. To this end, we *merge traffic flows* by taking advantage of the fact that the VC loop allows multiple source and destination nodes (multi-node VCs) on it, functioning as a virtual bus supporting arbitrary communication patterns [14]. Specifically, this merging can be done for *multicast*, *multiple-flow low-bandwidth*, and *round-trip (bi-directional)* traffic. In the example, for the two multi-cast traffic **a** and **h**, we specify two multi-node VCs for them as $\bar{v}_a(n_5, n_9, n_{10}, n_{11})$ and $\bar{v}_h(n_5, n_6, n_2, n_3)$. For multiple-flow low-bandwidth type of traffic, we can specify a VC to include as many nodes as a type of traffic spreads. For instance, traffic **c** and **f** include 4 node-to-node flows each, and their node-to-node flows require lower bandwidth, 512 Mbits for traffic type **c** and 128 Mbits for traffic type **f**. For **c**, we specify a a five-node VC $\bar{v}_c(n_{13}, n_{14}, n_{15}, n_{16}, n_7)$; for **f**, a three-node VC $\bar{v}_f(n_2, n_3, n_4)$. Furthermore, as we use a closed-loop VC, two simplex traffic flows can be merged into one duplex flow. For instance, for two **i** flows, we specify only one VC $\bar{v}_i(n_6, n_7)$. This also applies to traffic **b**, **d**, **j** and **k**.

Based on results from the last two steps, we then compute normalized bandwidth demand for each VC specification. Suppose link capacity $bw_{link} = 4096$ Mbits/s, 512 Mbits/s is equivalent to $\frac{1}{8}\ bw_{link}$. While calculating this, we need to be careful for duplex traffic. Since the VC implementation is a loop, a container on it offers equal bandwidth in a round trip. Therefore, duplex traffic can exploit this by utilizing bandwidth in either direction. For example, traffic **d** has two flows, one from n_{16} to n_{12}, the other from n_{12} to n_{16}, requiring 1/2 bandwidth in each direction. By using a looped VC, the actual bandwidth demand on the VC is still 1/2 (not $2 \times 1/2$). Because of this, the bandwidth requirements on VCs for traffic **b**, **d**, **f**, **i**, **j** and **k** are 1/8, 1/2, 1/16, 1/8, 1/8, and 1/8, respectively.

With the above steps, we obtain a set of VC specification as listed in Table 4.

7.3 Looped VC Implementation

In the VC implementation phase, we find a route for each VC and then compute the number n_c of required containers in order to support the required bandwidth. This is calculated by $n_c \geq \bar{bw} \cdot |v|$, where $\bar{bw}$ is the normalized bandwidth demand of $\bar{v}$ and $|v|$ is the loop length of the VC implementation v. After configuration, we obtain TDM VC implementations for all traffic flows, one for each VC specification. One feasible solution when link capacity is set to be 4096 Mbits/s is shown in Figure 21.

The VC implementation details are listed in Table 5. In total, there are 25 containers launched on 11 VCs. The network has a utilization of 52%.

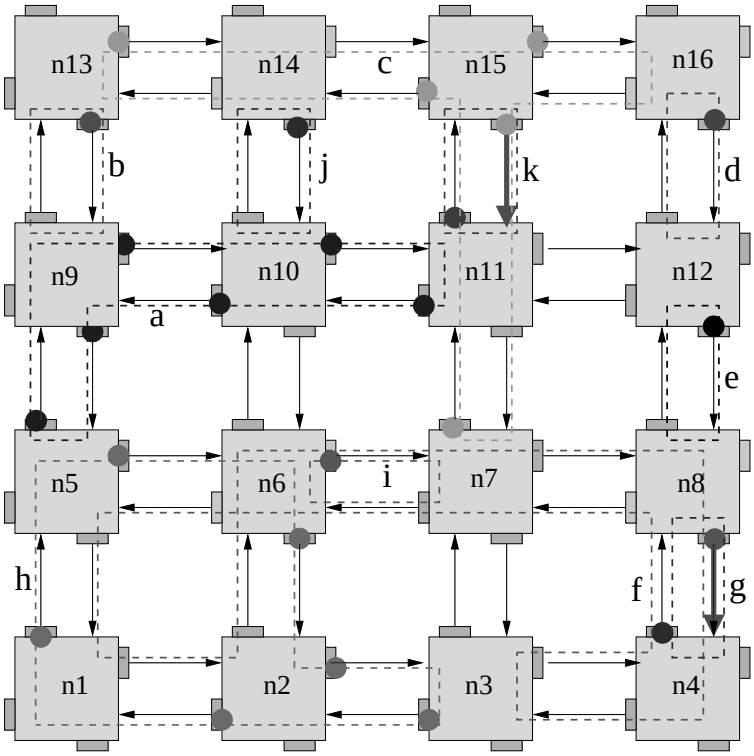


Figure 21: One solution of looped VC implementations with a snapshot of containers on VCs.

VC impl.	Traffic	Visiting nodes	Loop length	Containers	BW supply
1	a	n5, n9, n10, n11, n10, n9, n5	6	6	1
2	b	n9, n13, n9	2	1	1/2
3	c	n7, n11, n15, n14, n13, n14, n15, n16, n15, n11, n7	10	5	1/2
4	d	n12, n16, n12	2	1	1/2
5	e	n8, n12, n8	2	1	1/2
6	f	n3, n4, n8, n7, n6, n5, n1, n2, n6, n7, n8, n4, n3	12	1	1/12
7	g	n4, n8, n4	2	1	1/2
8	h	n1, n5, n6, n2, n3, n2, n1	6	6	1
9	i	n6, n7, n6	2	1	1/2
10	j	n10, n14, n10	2	1	1/2
11	k	n11, n15, n11	2	1	1/2

Table 5: Looped TDM VC implementations for traffic flows

8 Summary

We have addressed the provision of QoS for communication performance from the perspective of resource allocation. We have seen that we can reserve communication resources exclusively during the lifetime of a connection (circuit switching) or during individual time slots (TDM). We have discussed non-exclusive usage of resources under the term aggregate resource allocation and noticed that QoS guarantees can be provided by analyzing the worst case interaction of all involved connections. We have observed a general trade-off between the utilization of resources and the

tightness of bounds. If we exclusively allocate resources to a single connection, their utilization may be very low because no other connection can use them. But the delay of packets is accurately known and the worst case is the same as the average and the best case. In the other extreme we have aggregate allocation of the entire network to a set of connections. The utilization of resources is potentially very high because they are adaptively assigned to packets in need. However, the worst case delay can be several times the average case delay because many connections may compete for the same resource simultaneously. Which solution to select depends on the application's traffic patterns, on the real-time requirements and on what constitutes an acceptable cost.

In practice all the presented techniques of resource allocation and arbitration can be mixed. By using different techniques for managing the various resources such as links, buffers, crossbars, NIs, etc. a network can be optimized for a given set of objectives while exploiting knowledge of application features and requirements.

References

[1] Tobias Bjerregaard and Jens Sparso. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 2*, pages 1226 – 1231, March 2005.

[2] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 50(2-3):105–128, February 2004.

[3] Allan Borodin, Yuval Rabani, and Baruch Schieber. Deterministic many-to-many hot potato routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):587–596, 1997.

[4] Kuei-Chung Chang, Jih-Sheng Shen, and Tien-Fu Chen. Evaluation and design trade-offs between circuit-switched and packet-switched NOCs for application-specific SOCs. In *Proceedings of the 43rd Annual Conference on Design Automation*, pages 143–148, 2006.

[5] Rene L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[6] William James Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufman Publishers, 2004.

[7] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):21–31, Sept-Oct 2005.

[8] C. Hilton and B. Nelson. A flexible circuit switched NOC for FPGA based systems. In *Proceedings of the Conference on Field Programmable Logic (FPL)*, pages 24–26, August 2005.

[9] Axel Jantsch. Models of computation for networks on chip. In *Proceedings of the Sixth International Conference on Application of Concurrency to System Design*, June 2006. invited paper.

[10] Jean-Yves LeBoudec. *Network Calculus*. Number 2050 in LNCS. Springer Verlag, 2001.

[11] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, and D. Verkest. Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs. In *Proceedings of the Internationial Conference on Hardware/Software Codesign and System Synthesis*, pages 81–86, September 2005.

[12] A. Lines. Asynchronous interconnect for synchronous SoC design. *IEEE Micro*, 24(1):32–41, Jan-Feb 2004.

[13] Zhonghai Lu and Axel Jantsch. Slot allocation using logical networks for TDM virtual-circuit configuration for network-on-chip. In *International Conference on Computer Aided Design (ICCAD)*, November 2007.

[14] Zhonghai Lu and Axel Jantsch. TDM virtual-circuit configuration for network-on-chip. *IEEE Transactions on Very Large Scale Integration Systems*, 16(8), August 2008.

[15] Mickael Millberg and Axel Jantsch. Increasing NoC performance and utilisation using a dualpacket exit strategy. In *10th Euromicro Conference on Digital System Design*, Lubeck, Germany, August 2007.

[16] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the Design Automation and Test in Europe Conference*, Paris, France, February 2004.

[17] Erland Nilsson and Johnny Öberg. Reducing peak power and latency in 2-D mesh NoCs using globally pseudochronous locally synchronous clocking. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, September 2004.

[18] Daniel Wiklund. *Development and Performance Evaluation of Networks on Chip*. PhD thesis, Department of Electrical Engineering, Linkping University, SE-581 83 Linkping, Sweden, 2005. Linkping Studies in Science and Technology, Dissertation No. 932.

[19] Daniel Wiklund and Dake Liu. SoCBUS: Switched network on chip for real time embedded systems. In *Proceedings of the Parallel and Distributed Processing Symposium*, April 2003.

[20] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83:1374–1396, 1995.