Increasing NoC Performance and Utilisation using a Dual Packet Exit Strategy

Mikael Millberg & Axel Jantsch
ECS - KTH - Royal Institute of Technology, Sweden
{micke, axel}@imit.kth.se

When designing a network the use of buffers is inevitable. Buffers are used at the entry point, inside and at the exits of the network. The usage of these buffers significantly changes the performance of the system as a whole. In order to enhance the buffer utilisation the concept of letting more than one packet exit the network at every switch each clock cycle is introduced - *Dual Packet Exit (DPE)*. The approach is tried on a 4×4 and a 6×6 mesh. We demonstrate the buffers used in combination with different routing strategies for best effort performance. The result we present shows a 50% reduction in terms of worst case latency and a 30% reduction in terms of average latency as well as an increased throughput both from a system and network perspective. We define the term *Operational Efficiency* as a measure of the network efficiency and show that it increases by roughly 20% with the *DPE* technique.

1. Introduction

When offering services with best effort performance, naturally, no hard guarantees can be given due to the dynamic behaviour of any general purpose system. In order to make use of such services statistical performance measures are instead utilised. As a consequence the traffic has to be kept below a certain threshold for which the desired statistical properties can be given. These properties can be derived from a rigid reasoning based on the current implementation *or* from observed simulation results where the offered services can be given with certain properties within a safety margin.

If *real* guarantees are to be given the cost is often high since the capacity of the network has to be allocated in such way that the network is inherently bound to be over-dimensioned for any general scenario. If the traffic patterns are static and known prior to network setup the hard guarantees often offer a good alternative, but due to the dynamic behaviour of a general purpose system static traffic patters are rare.

Given that we are bound to offer services with statistical characteristics on performance, how do we do this at lowest possible cost? The cost in this context is the required buffers needed to guarantee a *no-packet-drop* policy together with a safety margin in terms of injection rate to "guarantee" a certain worst, and average, case latency.

The approach that we have chosen, within the mesh based NoC Nostrum [1], for giving the service of *Best Effort* at a low cost, is by utilising deflective routing in order to keep the size of the switches small [2]. The small size is a consequence of not employing explicit buffering. Through simulations with uniform traffic patterns, we can conclude that there seems to exist an upper bound on the performance of the network. On networks of the sizes 4×4 and 6×6 this upper bound is reached for an injection rate of 0.63, and 0.45, respectively, for the routing strategies tried out. Injection rate is defined as *packets per node and clock cycle*. Once this limit is reached packets start queuing up at the entry points of the network and the worst case system latency grows exponentially.

In the course of extensive simulation and performance analysis it became clear that the exit point from the network is a severe bottleneck that keeps packets unnecessarily long in the network. The obvious approach taken is to increase this bandwidth; we call this solution *Dual Packet Exit*. The benefit of the Dual Packet Exit is a higher throughput of the network and a significant lowering of the buffer requirements at the entry points to the network because the worst case scenario now happens at a higher injection rate.

The price for this solution is that packets now need to be buffered at the exits of the network, but the need for buffers at the entry of the network is reduced so that from an overall perspective the buffers required in total is kept constant (and even lowered) for a moderately to a heavily loaded network. For the 4×4 mesh the average system latency is reduced from 14 to 9 clock cycles and the observed worst case latency is reduced from 85 to 45 clock cycles at an injection rate of 0.63. This will give better

margins before the network saturates *or* a higher throughput with the previous margin kept. All this, of course, assumes that there exists a balanced load in the network in the sense that no single node or bisection cut of the network are exposed to a static over-utilisation.

The validity of the chosen approach is not restricted to uniformly random traffic patterns on meshes but also applicable to "any" topology where the traffic pattern involves potential network exit congestions due to multiple sources having the same destination or where multiple routing paths are possible.

The network exit strategy has not received as much attention by researchers as other parts of network design. Most work that in detail analyse cost and performance of a router and the network as a whole, e.g. [2, 3, 4] assume an ideal packet ejection model, which means that packets are absorbed by the receiving node as soon as they are delivered by the network. In [5] an ejection policy is studied that reduces the cost and complexity of the router while minimizing the impact on performance. However, to our knowledge no study about the trade-offs involved in increasing the network exit bandwidth has been reported, as we attempt in this paper.

The rest of the paper is organized as follows: We start with giving a general overview of the platform used, to help the reader to relate the results to other work in the field. After this, we discuss how packets are generated and buffered in the system together with two hard limitations on what performance we could expect from "any" network at best. Then we present the contribution of the paper together with simulation results comparing *Dual Packet Exit* with simple packet exit. Finally, some discussions relate the approach to a general scenario in order to show where it is valid and useful.

2. System Overview

The topology that is chosen for the network is a n×n mesh which employs deflective routing with no explicit buffering (i.e. no queues) in the switches. Every switch is connected to a resource in a pair-wise fashion and a switch/resource pair is called a node. The total number of nodes in the system is $N = n \cdot n$. Packets are produced (generated) by the resource's **Packet Source** process, sent over the network, and later consumed by the **Packet Sink** process at the destination resource.The switches are individually connected to its four neighbouring switches in the direction of the compass. In addition we accept *no* packet loss. In our simulator a packet will receive a multitude of time tags for post simulation data analysis during its lifetime. The **Packet Source** generates λ packets, on average, every clock cycle. The packet is assigned a sequence number, tagged with a birth time, t_B, and thereafter pushed onto the resource's **Downstream Packet Queue** waiting for permission to enter the network. Once admitted to the network, the packet gets a send timing tag, t_S, and tries to reach its destination with a minimal number of hops according to the routing scheme described below. At the destination node the packet is ejected from the network and is pushed onto **Upstream Packet Queue** of the destination resource - this achievement renders the packet a reception time tag, t_R. The **Packet Sink** process polls the queue and if a packet is found it is popped from the queue and tagged with a finish time tag, t_F.

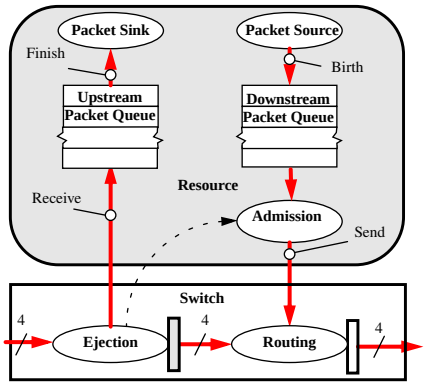

Fig. 2.1 The Switch and the Resource

Inside the switches there, conceptually, exist two separate stages - *Ejection* and *Routing*.

Ejection The ejection stage examines the incoming packets to detect if one (or more) have reached their destination and is to be delivered to the resource. In case of competition the packet with the highest priority is delivered. Also it informs the Resource's **Admission process** whether there is room in the switch for a packet to enter the network the next clock cycle. Since no explicit queues are employed in the network, admission can only be granted if the switch is currently holding fewer packets than its output buffer capacity, i.e. four packets.

Routing The deflective routing scheme is carried out in three phases: *Priority Assignment*, *Favoured Outport Selection & Permutation Routing*

Priority Assignment In this phase the incoming packets are dynamically assigned a priority, in our simulation the *Hop Count* (*HC*) is used. *Hop Count* is the time that the packets have spent in the network - a high *HC* means a high priority.

Favoured Outport Selection The packets now use their assigned priorities to select a desired outport. The priorities are utilised as credits which enable the packets to give different weights to favour a certain routing decisions in the coming *Permutation Routing* objective function. Here we try out two different strategies, *Uniform* and *Proportional*. *Uniform* implies that the packets use their priorities "uniformly" to select a favoured outport. That is, if a packet has a destination in a direction Northwest it will put a half of its priority to a routing decision where it gets routed to the West and half of its priority to a routing decision where it gets routed to the North. In the *Proportional*

strategy the packets favour a decision where the priorities are assigned proportionally to the direction of destination. An example: a packet has a destination two switches to the North and one to the West. The packet now chooses to use two thirds if its priority in favour of a decision where it will be routed North and the remaining third in favour of the West direction. The consequence of this strategy is that a packet will try to move in a direction where the degrees of freedom in routing is kept as long as possible to work against misroute closer to the destination.

Permutation Routing The weighted priorities of all the competing packets are summed to form the basis for selecting the best routing permutation. The number of permutations to select from in a four outport switch is 4! (= 24).

In order to slightly vary the routing strategy to make the analysis and claims about the importance of the buffer use stronger simulations are carried out with both the described variants of *Favoured Outport Selection*.

3. Packet Generation & Bounds on network Performance

Packets are generated with an average rate of λ. A λ of 0.3 means that there is a 30% chance that a packet will be generated during a clock cycle. The generation is "unaware" of the whereabouts of the network in the sense that it will perpetually generate packets regardless of the number of packets already in the **Downstream Packet Queues** or in the network.

Since each resource generates packets with an average rate of λ packets per clock cycle the total number of packets generated in the system every clock cycle is $N \cdot \lambda$.

The destinations of the generated packets are spread uniformly random over the network. This means that there, on average, will be a balanced load in the network, from the perspective of the source and destination nodes. The implications and validity of this approach will be further discussed in *8 - Discussions and future work*.

3.1. The Bisection Cut Bandwidth

The bisection bandwidth of the network is equal to the number of links crossing any bisection of the network [6].

The reasoning is the following: If all nodes emit packets with a uniformly random destination distribution half of the packets will with 50% probability cross the bisection in one direction.

$$\lambda \cdot N/4 \leq n$$

$$\lambda \leq 4n/N = 4n/(n \cdot n) = 4/n$$

If $n=4 \Rightarrow \lambda \leq 1$ and $n=6 \Rightarrow \lambda \leq 0.67$ this is the first upper bound of our network and it gives us a limit on how many packets that can, under uniform load, be transferred over the network per node.

4. Buffering

Buffers exist at three places: in the **Downstream Packet Queue**, in the network, and in the **Upstream Packet Queue**.

When talking about buffering we will discuss utilisation as well as minimum required capacity. Buffer utilisation is the average number of buffers utilised during the simulation, the utilisation is coupled to the dynamic energy consumption since we assume that energy consumption, in the buffers, is mainly dependent upon whether the buffers currently hold a packet or not. Regarding the minimum capacity it is the number of buffers needed to fulfil the requirement of no packet drop. For the **Up-** and **Downstream Packet Queues**, (which are implemented as FIFOs) the assumption of linear dependency of power consumption and buffer utilisation is unrealistic since they could be implemented in a memory structure as circular buffers. The buffer capacities of the FIFOs are derived from a worst case observed in the simulations presented in Section *7*. The buffer capacity of the network, is in general, equal to the number of switches times the number of buffers they contain.

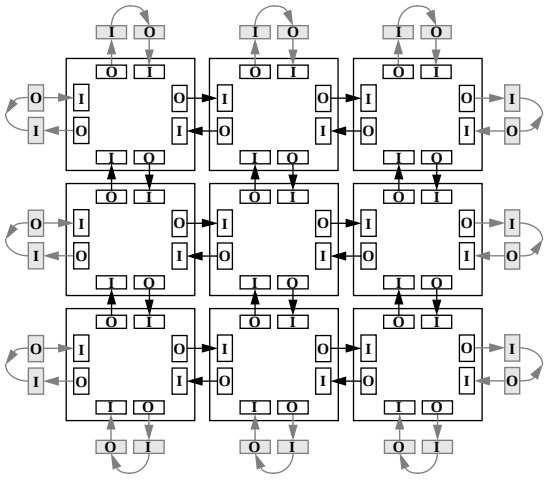

Fig. 4.1 The Buffers in the Network

Since the switch uses two stages: *Ejection* and *Routing* we have both in and output buffering (b=2). The number of buffers for the network is

$$BC = switchBuffers + edgeBuffers = 4 \cdot b \cdot n \cdot m + 2 \cdot b \cdot (n+m)$$

In the case $n=m$ we get

$$BC = 4 \cdot b \cdot n \cdot (n+1)$$

In our case with n=4 and n=6 we have 160 and 336 buffers available, respectively.

The geometrical distance between any two switches is denoted, $d_{geom.}$ The distance in clock cycles is the geometrical distance times the number of buffer stages, b, used in the switches. In addition to this, clock cycles for the buffer stages in the sender and receiver switches need to be taken into account to make a complete analysis, these are not, however, included in the formula below

$$d = b \cdot d_{geom}$$

If all nodes are emitting packets with a uniform random

destination, where the destinations are all other nodes excluding the sending node, the average geometrical distance, in a $n \times n$ mesh, is according to [6]:

$$d_{geom} = 2 \cdot n/3$$

The average distance in clock cycles is hence

$$d = b \cdot d_{geom} = b \cdot 2 \cdot n/3$$

On our case, with double buffering in the switches we get an average distance of $d \approx 5.3$ (n=4) and $d = 8$ (n=6). During the routing of a packet it will traverse a number of buffers along its way. The longer distance a packet has to travel the more buffers/network buffer capacity it will utilise. This was first described by Little [7] and in our example the average number of buffers utilised under "minimal" routing is:

$$B_{used} = P_{tot} \cdot d = \lambda \cdot N \cdot b \cdot 2 \cdot n/3 = 2\lambda \cdot b \cdot n^3/3$$

4.1. The Network Buffer Capacity

In order to satisfy the demand of Little's formula the available number of buffers in the network has to be higher than the buffers required by the traffic. In the expression below the buffers at the edges are removed since they are not part of a minimal path.

$$\begin{aligned} B_{used} &< n \cdot n \cdot b \cdot 4 - (n+n) \cdot 2 \cdot b \\ 2 \cdot \lambda \cdot b \cdot n^3/3 &< 4 \cdot b \cdot n \cdot (n-1) \\ \lambda &< 6 \cdot (n-1)/n^2 \end{aligned}$$

This is the second upper bound of our network and with n= 4 we get $\lambda < 9/8$ and n=6 gives $\lambda < 5/6$

5. The Contribution - Dual Packet Exit

The contribution of this paper is the observation that the routing time for each packet inside the network has three different components: Minimum routing distance, Deflection *prior* to reaching destination for the first time, Deflection *after* reaching destination.

The first component - the minimum routing distance is not something that we can do much about, it is an lower bound that stems from the mapping and the traffic pattern.

The second component is the deflection prior to reaching destination. By this we mean the deflection that occurs before it reaches its destination for the first time. Once the packet has reached its destination for the first time and potentially is deflected due to an exit congestion it is considered to be in the third category.

The second component is however, a consequence of the competition for resources in the network. This in turn is dependent upon the overall load of the network and routing strategy. The load and routing strategy is tightly coupled in a looped fashion in the sense that a good routing strategy gives a lower load which in turn improves the possibilities for better routing. This component can easily be reduced by lowering the load of the network or by choosing a better routing strategy which is considerably more difficult. The difficulty lies within the problem of choosing a strategy that gives a good performance to *any* traffic pattern.

The third component is the deflection *after* reaching destination, which is the component that we here aim to lower.

5.1. Exit Congestion Limit

If two (or more) packets will reach the same destination node at the same time at least one of them will be deflected which contributes to the routing distance used in Buffer capacity bandwidth. The contribution will be four extra clock cycles for that particular packet since we are employing both in and output buffering of the switches.

The chance of two packets having the same destination is 1/N, if their individual destinations are randomly chosen in the range 1..N. If three random packets are chosen, {A, B, C} the cases of packet A and B, packet A and C and packet B and C having the same destination has to be taken into the formula. As well as the possibility of all three packets having the same destination has to be taken into account. The last scenario with three competing packets must be weighted with the penalty of *two* deflections. For an increasing number of competing packets the scenarios quickly becomes significantly more complex and a closed expression that captures the penalty for multiple packets having the same destination is hard to find. This problem has strong resemblance with what's referred to as the *Birthday Paradox*, which is the, not intuitive, high chance of two, or more, people having the same birthday in a group. If n is the number of people the chance of two or more people having the same birthday is given by the equation

$$P(n) = 1 - \frac{365!}{365^n (365 - n)!}$$

For the Birthday Paradox, a closed expression for the probability of coinciding birthdays apparently exist but does not, naturally, incorporate expressions for penalties varying with number of coinciding birthdays. In [12] this is well described and also it is shown that the uniform distribution gives rise to the smallest number of coincidences. For our problem this means that any non-uniform packet distribution worsens the problem of congestion at the exits of the network. However, this procedure of calculating the

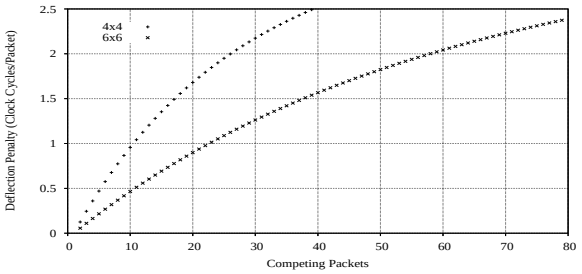

Fig. 5.1 Deflection Penalty

deflection penalties is carried out with an increasing number of packets competing and the number of deflections is depicted in Fig. 5.1. The horizontal axis shows the number of competing packets and the vertical shows the

average deflection penalty for each packet in a 4×4 and a 6×6 network. In short - this is the *average* penalty, in hops, that *every* packet gets its routing distance increased due to the exit node congestion!

If we compare these data with the circumstance that the average minimum routing distance in a 4×4 network is ≈5.33 the penalty of 1-2 clock cycles becomes significant due to the coupled load and average routing distance which will even further worsen the penalty in clock cycles due to the increased load, which will even further increase the load, and so on.

The obvious remedy for the exit packet congestion problem is to let more than a single packet exit the network per clock cycle and node. The cost connected to this is the dual wiring needed from the switch to the Resource together with the **Upstream Packet Queues** and corresponding logic. However this cost is relatively low compared to the gains that can be made in the network and **Downstream Packet Queues** as illustrated in Section *7*.

6. Simulation setup

The simulator used is fully written in SystemC [11] on a cycle accurate basis. It implements the network as well as the packet generators and the queues in the resources.

As mentioned before all resources generate packets with a rate of λ packets per cycle, with a random uniform destination pattern. All resources will generate 16000 packets each, in their respective, **Source Processes**. In total $P_{system} = N*P_{node}$ ($N=16 \Rightarrow P=256000$, $N=36 \Rightarrow P=576000$) packets will be generated during one simulation. The simulation is stopped when all packets are delivered. The reason for sending 16000 packets per node is that this is enough for making the effect of start-up and empty phases insignificant to the total result. The parameter that is changed from one simulation to the next, within one simulation run, is the injection rate, that is swept from 0.3 - 0.7 in steps of 0.002 for the 4×4 mesh. This means that we get 1+(0.7-0.3)/0.002 (201) measurement points from every simulation run. For the 6×6 network the range is 0.1 - 0.5 in steps of 0.004 which give 100 measurement points. In some graphs presented not the full range of measurement points are present in order to enhance the readability of the interesting portions of the graph.

We describe the measurements that we have chosen to present from two main perspectives: *End User & System Designer.* The *End User,* is interested in *Performance* like *Throughput & Latency* whereas the *System Architect* which may be more interested in implementation costs, like *Required Buffer Capacity*, and effectiveness measurements. Of course both of these are closely related and the *Average Latency* is an obvious shared concern since the *End User* sees packet latency but the *System Architect* may consider this as an increased cost in terms of energy due to buffering and switching.

6.1. Performance - Worst Case Latency

The *Worst Case Latency* is the biggest difference in t_B and t_F that we can find for any packet during the simulation. In order to enhance readability the worst cases in terms of latencies the graphs are made monotonously increasing, this is also done for the *Required Buffer Capacity.*

6.2. Average Latency

The average latency is the average of all packets' individual latencies. The *System Latency* and *Network Latency* of a packet is derived from (t_F-t_B) and (t_R-t_S), respectively.

6.3. Required Buffer Capacity

The Required Buffer Capacity has three components: The sum of all **Up-** and **Downstream Packet Queue** sizes and the buffer capacity of the network. The individual sizes of all the downstream packet queues in the resources are dimensioned from the observed worst case load of *any* downstream packet queue during the simulation. E.g. if one packet queue at any point in time held 10 packets all the packet queues in the network are given that size. The same is done for the up stream packet queues. In short:

$$N \cdot (DSLoad_{WorstCase} + USLoad_{WorstCase}) + BC$$

6.4. Operational Efficiency - Throughput per Buffers Used

The throughput (accept bandwidth) of the network is the average number of packets the system can deliver per clock cycle. When increasing the injection rate the throughput increases accordingly until the network is saturated. Given this fact it is easy to jump to the conclusion that the best performance is achieved by saturating the network! However, loading the network to this extent must come with a cost. The cost is the increase in average buffers needed to transfer a packet through the system, together with a, potentially, higher worst case latency. Ignoring the worst case latency, we propose the odd measure of *Operational Efficiency* to capture the *Throughput per Buffers Used* in the network. The idea is that, both, an underutilised as well as an overutilised network will give a bad ratio between the throughput delivered and number of buffers that is used. From a system perspective this is expressed as

$$OpEfficiency = \frac{PacketsTransmitted}{TotalTransmTime \times BuffersUsedPerPacket}$$

Since we do not have a realistic power model for the current implementation of the network it is hard to give any concrete numbers of how much energy a network will require in order to give a certain throughput. Anyway this measure will give a picture of the potential load of the network that gives acceptable performance.

7. Simulation Results - Comparisons and Discussion

7.1. Required Buffer Capacity

Both for the 4×4 and the 6×6 network it can be seen that the *Dual Packet Exit (DPE)* based approach has a higher buffer requirement at lower load, this since we get an early contribution from the **Up-stream buffers**. When the injection rate increases the 4×4 DPE network breaks down later than the *non-DPE*. This means that we, *either* can, drive the network harder *or* give performance guarantees with better margins! For the 6×6 network we get basically the same behaviour both with and without the DPE. The reason for not getting the improvements of the 4¥4 network is that since the network is bigger the packets spend a proportionally smaller time competing for ejection than on routing.

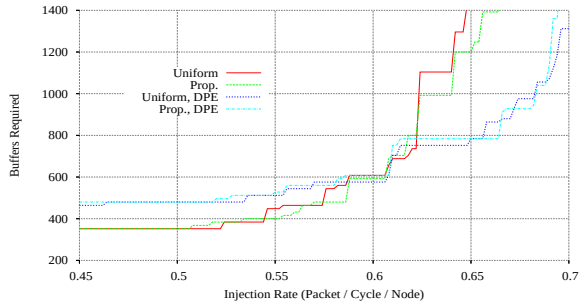

Fig. 7.1 Required Buffer Capacity with and without DPE - 4×4

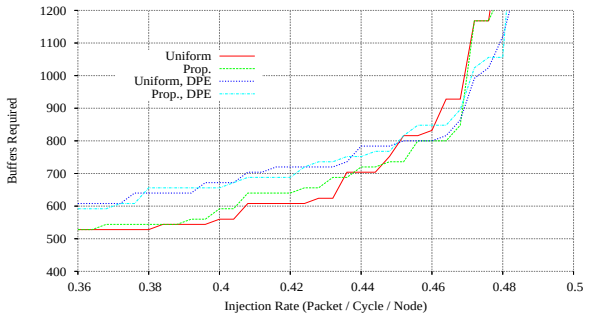

Fig. 7.2 Required Buffer Capacity with and without DPE - 6×6

7.2. Average Total Latency

If we look at the average total latency in the system we can see that all strategies have basically the same latency for low loads but for higher loads the DPE outperforms the other by far. If we set for an average latency of 10 cycles for the 4×4 mesh to the strategy with no *DPE* and the *DPE* gives roughly 25% higher injection rate for the same latency. The main contributing factor is that packets, to higher extent, are given access to the network. The contribution of the *Exit Congestion Limit*, described in section *5 - The Contribution - Dual Packet Exit*, can be seen if we only look at the latency *within* the network, as depicted in Fig. 7.5. The *DPE* clearly reduces this effect and hence give a lowered average latency.

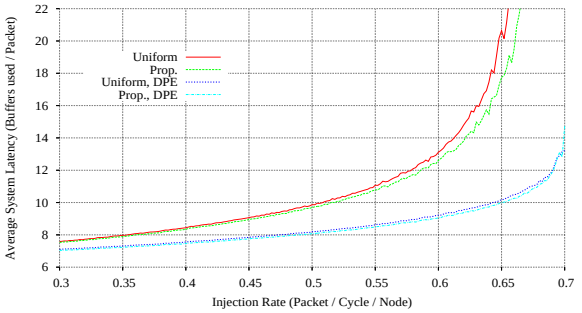

Fig. 7.3 Average System Latency - 4×4

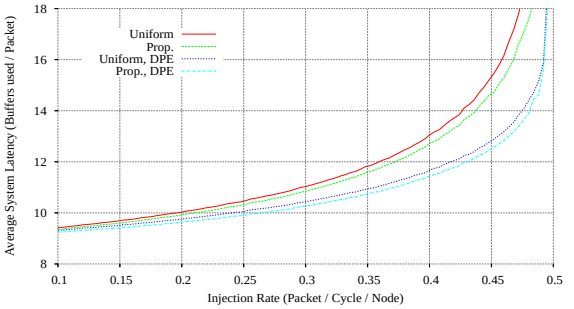

Fig. 7.4 Average System Latency - 6×6

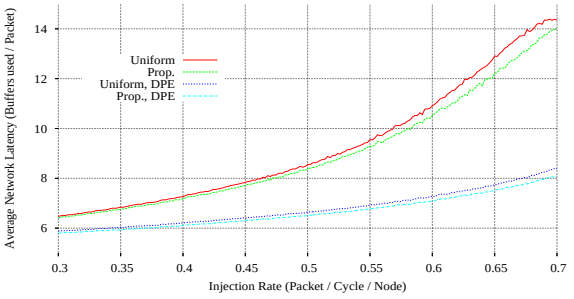

Fig. 7.5 Average Network Latency- 4×4

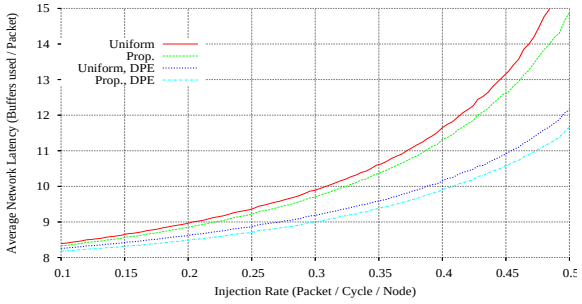

Fig. 7.6 Average Network Latency - 6×6

7.3. Performance - Worst Case Latency

As seen in Fig. 7.7 the worst case latency is reduced for the full spectra of injection rates for both sizes of the network. This result we consider to be the strongest benefit of the DPE approach since the worst case latencies now is within the same magnitude as the average case latencies. Also the worst case latency of the “break down” injection rate can be slightly shifted by the use of the *DPE* which

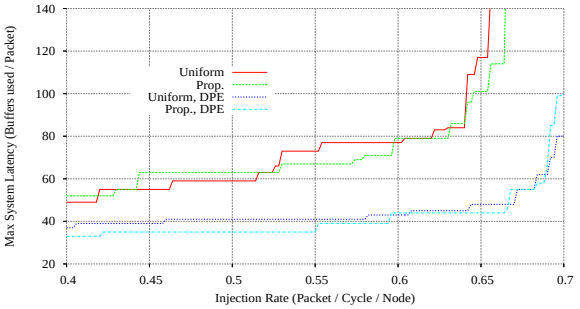

Fig. 7.7 Worst Case Latency - 4×4

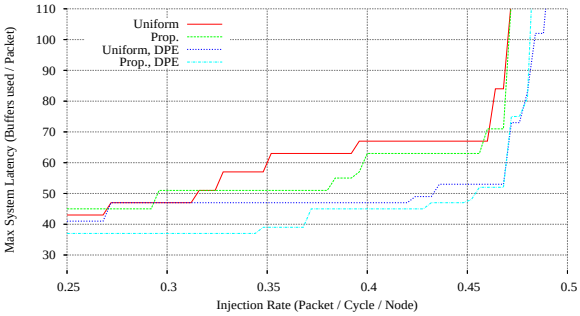

Fig. 7.8 Worst Case Latency - 6×6

can potentially give better margins before packet drop occur due to buffer overflow.

7.4. Operational Efficiency - Throughput per Buffers Used

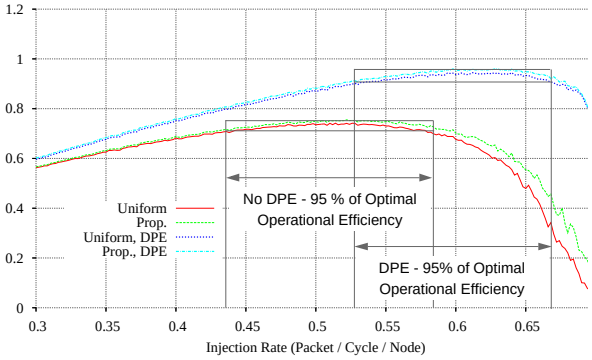

Fig. 7.9 Throughput Per Buffers Used - 4×4

Since the *Operational Efficiency* is derived from

$$\text{OperationalEfficency} = \frac{\text{SystemThroughput}}{\text{BuffersUsedPerPacket}}$$

the immediate observation that can be made in Fig. 7.9 is that with the *DPE* approach the throughput of the network can be increased while still using the same number of buffers. Also it can be seen the network utilising DPE is more effective. If we define a region where the network operates within 95% of its optimum that region is considerably moved in a higher throughput area ([0.44..0.59] ⇒ [0.53..0.67]) and the drop-off of the effectiveness also happens much closer to the saturation point of the DPE network. For the 6×6 network the situation is similar even

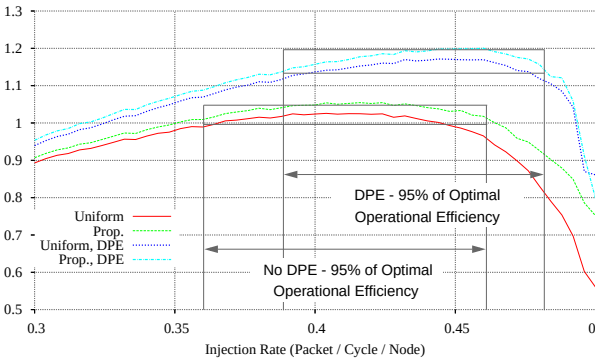

Fig. 7.10 Throughput Per Buffers Used - 6×6

though the *relative* effect is slightly reduced. Once again the relative reduction stems from the fact the packets spend relatively less time waiting for admittance in a larger network.

8. Discussions and future work

In this section we reason about the validity and generality of out approach.

8.1. Input/output balance

In our simulation we have assumed the network load to be statistically balanced. By this we mean that on average there exists a balance between the number of packets destined for a particular destination node in such way that if, on average, λ packets are generated by each of the N participating nodes at every clock cycle, λ packets will have a particular node as destination node. As shown in *5 - The Contribution - Dual Packet Exit* there exists a variance in the uniformity of the packet destination and due to this a potential congestion at the exits of the network. This is the reason for implementing the Dual Packet Exit. One might argue that if we have a "true" balance in the sense that we generated packets every λ clock cycle one "unique" packet for every destination, i.e., the packets generation process every λ is a permutation between the sets $\mathbf{N}_{16} \Rightarrow \mathbf{N}_{16}$ will the approach still hold? We claim that the answer is yes, and reason as follows:

The time the packets spend in the **Downstream Packet Queues** and in the network could for good reason be considered as random. For instance assume the latency for any packet varies by 10 clock cycles. The start permutation $\mathbf{N}_{16} \Rightarrow \mathbf{N}_{16}$ would become $\mathbf{N}_{160} \Rightarrow \mathbf{N}_{160}$, with 16 unique symbols, and the "permutation balance" would be more or less gone. This is not validated quantitatively and needs to be investigated further.

8.2. Applicability to other routing strategies and topologies

For any network that implements Best Effort and has a jitter in latencies that stems from random admission queuing times and/or there exist non-determinism in the routing

the problem of congestion at the output of the network will occur. And hence would benefit from an increased "exit-from-the-network" bandwidth.

We suspect that the effects for deterministic routing techniques such as wormhole routing are basically the same. Any worm denied exit from the network would be locking up resources in the network. These resources (buffers) would in turn, potentially, lock up other resources and so on. It will be interesting to investigate this presumption in detail.

8.3. Size of the Network

The results presented could potentially be claimed to be unrealistic due to the relatively small size of the network but we have a strong conviction that the network in question is quite realistic from the perspective of the uniform load in the sense that the 4×4 network could be seen as a subset of a bigger network where some nodes are employing an all-to-all communication pattern.

8.4. Uniform Load

The choice of uniform destination selection is most questionable from a real world application perspective. The obvious reason for choosing this pattern is simplicity and the possibility of making general statements from a relatively easily to describe approach. Up till now we are not aware of any "official", (by the NoC community approved) benchmarks for measuring performance. Hence any choice of traffic pattern that has some special characteristics has to be rigorously motivated before any claims can be made.

Except for the above mentioned difficulties we believe that the uniform traffic pattern actually is some kind of "best-case" pattern, because if we added burstiness to our approach the effect of limited exit bandwidth would actually be worsened since we know, if a small portion of packets get misrouted, will have a guaranteed exit congestion.

The same is valid for any stream between any two nodes since keeping the network under-utilised would be the only guarantee against the effect of a misroute disturbing the balance. Moreover, all experiments with specific application traffic patterns will not allow to draw general conclusions because any result may not hold for other traffic patterns. Even tiny differences of the traffic patterns may have a profound impact on cost and performance of the network. Thus, in summary the uniform distribution is a crude but robust assumption preferable to any other more "realistic" but arbitrary traffic.

8.5. Hardware Cost

The hardware cost that has been discussed in this paper is the relative cost of buffers. The cost due to extra wiring and logic between the switch and the resource is not included but our intention is to investigate this in future work.

9. Conclusions

We present the concept of *Dual Packet Exit* in order to increase the outgoing bandwidth between the network and resources since it is identified as a bottleneck. The effect of this increase in bandwidth gives better throughput and a lowered buffer requirement. This, *either* can be utilised for a higher packet injection rate offering better throughput with the same margins to network breakdown; *or* it can be utilised for a achieving a lowered load in the network for a fixed packet injection with better margins when offering QoS for the same throughput as result.

The buffers needed in order to buffer incoming traffic to the network are reduced but instead buffering of outgoing traffic is introduced. The net requirement of buffers is however reduced for a fixed injection rate. Using the DPE approach not only increases the throughput of the network while still using the same number of buffers, also it can be seen a system utilising DPE is more effective.

The generality of the approach is discussed Section *8* and we suppose that this bottleneck may exist in other networks than Nostrum as well. In addition to the future work suggested we will also investigate effects of further increase in the outgoing bandwidth.

10. References

[1] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch. The Nostrum backbone - a communication protocol stack for networks on chip. In Proc. of the VLSI Design Conference, Mumbai, India, January 2004.

[2] A. A. Chien. A cost and speed model for k-ary n-cube wormhole routers. IEEE Transactions on Parallel and Distributed Systems, 9(2):150-162, Feb. 1998.

[3] L. S. Peh and W. J. Dally. A delay model for router micro-architectures. IEEE Micro, pages 26-34, Jan.-Feb. 2001.

[4] E. Rijpkema, et al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In DATE, 2003.

[5] Z. Lu and A. Jantsch. Flit ejection in on-chip wormhole-switched networks with virtual channels. Proc. of the IEEE NorChip Conf., 2004.

[6] W. J. Dally and B. Towles. Principles and Practices of Interconnection Networks. Morgan Kaufmann, 2004

[7] J. D.C. Little. A proof of the queueing formulae L=λW. Operations Research, 9(3):383-387 May 1961

[8] E. Nilsson, M. Millberg, J. Öberg, A. Jantsch. Load distribution with the proximity congestion awareness in a network on chip. DATE 2003.

[9] M. Millberg, E. Nilsson, R Thid, A. Jantsch. Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip. In DATE, 2004

[10] U. Feige, P. Raghavan. Exact analysis of hot-potato routing. In Proc. of Foundations of Computer Science, p. 553 -562, 1992

[11] System C, www.systemc.org

[12] D. Bloom. A birthday problem, American Mathematical Monthly 80 (1973), pages 1141-1142.