Priority Based Forced Requeue to Reduce Worst-Case Latencies for Bursty Traffic

Mikael Millberg Axel Jantsch
ECS – ICT – KTH
Royal Institute of Technology, Sweden
{mick, axel}@kth.se

Abstract - **In this paper we introduce *Priority Based Forced Requeue* to decrease worst-case latencies in NoCs offering best effort services. *Forced Requeue* is to prematurely lift out low priority packets from the network and requeue them outside using priority queues. The first benefit of this approach, applicable to any NoC offering best effort services, is that packets that have *not yet* entered the network now compete with packets *inside* the network and hence tighter bounds on admission times can be given. The second benefit – which is more specific to deflective routing as in the *Nostrum* NoC – is that packet "reshuffling" dramatically reduces the latency inside the network for bursty traffic due to a lowered risk of collisions at the exit of the network. This paper studies the *Forced Requeuing* on a mesh with varying burst sizes and traffic scenarios. The experimental results show a 50% reduction in worst-case latency from a system perspective thanks to a reshaped latency distribution whilst keeping the average latency the same.**

I. Introduction

When offering best effort services naturally no performance guarantees can be given. However, it is desirable to offer these services with the best possible performance in terms of throughput, latency, and worst-case behaviours. In this paper, focus is set on improving the worst-case latencies for multi-packet messages. Multi-packet messages manifest as traffic bursts in the network and dramatically worsen the performance. Most real world traffic exhibits burstiness to some degree. This does not constitute any problems if the traffic is orchestrated in such way that traffic bursts in the system do not collide on their way to destination. However, traffic that *is* utilising the best effort services often does so because the traffic behaviour *is* hard to predict in detail and hence, disqualified from utilising a guaranteed service.

During a packet's lifetime it will go through three separate phases: *admission* to the network, *transport* through the network, and *exit* from the network. Our previous work mainly focused on the two latter phases [7, 9] whereas this paper approaches the problem of *admission*. The approach that is chosen within the NoC *Nostrum* [8] for offering *best effort* services at a low cost uses deflective routing to keep the size of the switches small since no explicit buffers are used [4]. Most NoCs today employ variants of wormhole routing in favour of deflective routing mostly due to the packet reordering issue of deflection routing. However, studies carried out by Tota et al. shows the deflective NoC competitive, and possibly advantageous, to wormhole routing in terms of area and power [10]. The performance is neither better nor worse than its competitor on realistic multiprocessing benchmarks. In parallel to the deflective best effort services *Nostrum* also offers quality guaranteed services using a TDMA based scheme relying on the concept of Temporally Disjoint Networks [8]. During start-up of *Nostrum* different traffic streams are assigned to the appropriate services.

The key problem that we address is that: *Regardless of routing policy the best effort services inherently have a problem giving statistical bounds on the admission time to the network, i.e. bounds on the down stream queuing time before a packet can enter the network.* The reason is that it is hard to predict the traffic in the switch connected to the resource where a packet is to be injected into the network. In our earlier work [7] a solution to the problem of a guaranteed throughput service utilising the concept of *looping containers* was presented. Here, a solution for the best effort case is proposed. The worst-case waiting time is kept down by prematurely lifting out low-priority packets from the network to be requeued. A successful concept similar to ours is the *Diverting Switch* of Lang et al. [6] where packets in a competitive situation are sent to an alternative destination in the network to be resent later.

The validity of the concept is demonstrated in simulations; one using a uniform random pattern and another focusing on traffic to centrally placed memories. Both scenarios explore varying degrees of bursty traffic. In order to explain why bursts are harmful to network traffic an estimate is derived on the extra cost of congestion at the exits of the network as a function of the emission probability and the burst size. To our knowledge, no work has been presented working on an estimate on the delay due to multi-packet admissions in deflection routing networks. However, a number of papers describing upper bounds on delivery times in a network exist, e.g. the work of Hajek [5] and Brassel [2].

The paper starts with an overview of the platform used to help the reader in relating results to other work in the field together with the technical contribution of the paper. Next, the implication on the network performance in the presence of bursts and an estimation on what performance that can be expected from "any" network at best is discussed. Then, a hardware architecture with an “acceptable” cost is suggested and justified by simulations comparing a *Forced Requeue* system with a system without. Finally, some discussions relate the approach to a general scenario to show where it is valid and useful.

II. System Overview with Priority Based Forced Requeue

The topology of our network is an n×m mesh employing deflective routing with no explicit buffering, that is, no queues in the switches. Every switch is connected to a resource. A switch/resource pair is called a node. Packets are generated by the resources’ **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. The **Packet Source** generates λ packets, on average, every system clock cycle. Generated packets are pushed onto the resource’s **Downstream Packet Queue** waiting for permission to enter the network. To simplify the analysis in the current setup only one out of the four independent time-slots of the TDMA based network is utilised and analysed. Hence, the system delays are scaled according to the **Packet Source** process’s clock. At the destination node the packet is ejected from the network and pushed onto the **Upstream Packet Queue** of the destination resource. The **Packet Sink** process polls the queue and if a packet is found it is taken from the queue and can be considered delivered.

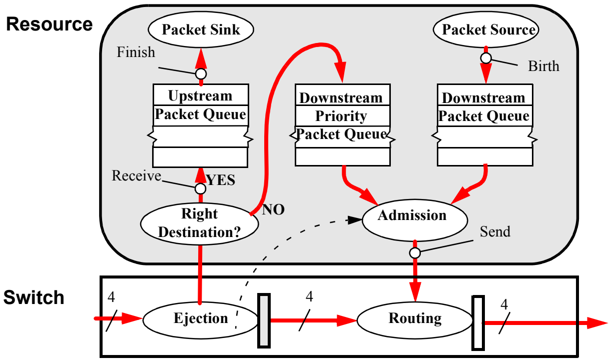


Fig. 1. The Switch and the Resource

Conceptually, inside the switches two separate stages exists – *Ejection* and *Routing*. In the *Ejection stage* incoming packets are examined to detect if they have reached their destination and are to be delivered to the resource. In case of competition the packet with the highest priority is delivered. Also, the Ejection stage informs the Resource’s **Admission process** whether there is room in the switch for a packet to enter the network at the next clock cycle. Since no explicit queues are used in the network, admission can only be granted if the switch holds fewer packets than its capacity of four packets.

In the Routing stage the incoming packets are dynamically assigned a priority, in our simulation the *Hop Count* (*HC*) is used. The HC is the time a packet has spent in the network – a high HC means a high priority. The priorities of all competing packets are used to select the best routing permutation.

A. Priority Based Forced Requeue (PBFR).

The contribution of this paper is the idea that low priority packets/worms can be taken out from the network before they actually reach their final destination. The packets that have been forcefully taken out are requeued to be admitted to the network later. By forcefully taking out packets, the worst-case latencies that are caused by being in the downstream packet queues, potentially indefinitely long, are significantly reduced. The cost is mainly in hardware since a *priority queue* needs to be implemented in the Network Interfaces. The priority queue, however, *does not have to hold all the packets* of the **Downstream Packet Queue.** The packets originating from this very resource are known to be sorted already and can be kept in a separate queue. From a performance point of view, the penalty for taking packets out of the network is an increased delay for the *individual* packets that are forcefully requeued. From a burst point of view this is a non-issue ¦rather the opposite thanks to the phenomena of Section III.

III. Understanding the Effects of Bursty Traffic

As can be seen in Section V., multi-packet bursts severely decrease the network performance; this despite that the traffic patterns, and the average packet injection rates, λ (in packets per cycle), are the same. Depicted in Fig. 2 are the increased individual packet latencies as a function of the burst sizes. As can be observed these latencies increase *at least* linearly with increasing burst sizes and hence make the packets of the bursts arrive at their respective destinations scattered in time. To develop an intuition why this degradation of the network performance occurs one must have knowledge about two related phenomena: How often packets in a network have the same destination and what is the cost incurred by this. We prioritise intuition over theoretical rigor and hence, this section should be considered a pointer in understanding the negative effect bursts could have on a network. Also, in our analysis only the effect of packet collisions at the destinations will be targeted, i.e. excluding the effects of bursts on the potential misroute of packets. Since the reasoning was developed with the deflective routing in mind the word *packet* is used – the reasoning in the

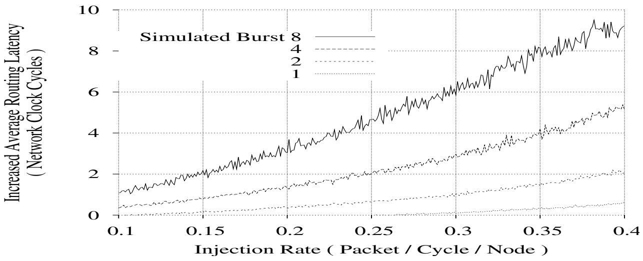


Fig. 2. Increased Routing Latency due to Bursts

upcoming Subsection III - *A. How Common are Common Destinations?* is, however, valid for a worm as well. The same holds for Subsection III - *B. What Does a Collision Cost?* where the burst size is changed into the length of the worm and a packet is a flit in the case of wormhole routing.

A. How Common are Common Destinations?

Assume a uniform distribution of the packet destinatios. How likely it is that packets will have the same destination? The answer is that it is, non-intuitively, very likely! To demonstrate, a very small network with only 4 nodes is chosen as a starting point. The nodes are all sending and receiving packets. If all permutations of destination patterns are transferred into a table that displays the relative frequencies of one or more packets sharing destination we get:

Competitors	Packets	Relative frequency
0	432	42 %
1	432	42 %
2	144	14 %
3	16	2 %

This means that only 42% of the packets will not experience *any* competition for destination whereas 58% will experience competition from *at least* one other packet! This problem is a variant of the *Birthday Paradox* [1] which is the, not intuitive, high chance of two (or more) people in a group having the same birthday. Intuitively one might object to that collision just *appears to be* this common due to two reasons. First, the small size of the network makes these numbers highly unrealistic. Second, uniform traffic does not coincide with any "real" scenario. To counter the first objection the competitions per packets for the 4x4 network are presented below.

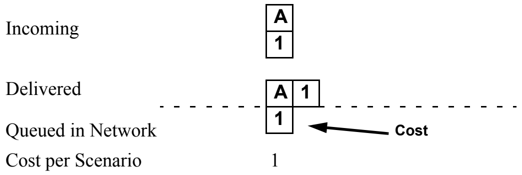
Competitors	Relative frequency
0	38 %
1	38 %
2	18 %

This means that, on average, 9 out of 16 packets will compete with other packets for any outcome. As it comes to the second objection it turns out that uniformly random traffic pattern actually gives the *least* number of coinciding destinations among the *random* traffic patterns. So, in general, packets/worms are most likely to share a destination under a random traffic pattern.

B. What Does a Collision Cost?

Given the conclusion about shared destinations – how likely is it that a packet will collide with other packets with the same destination and what does the collision cost? In order to develop an intuition some simplifications are made by only assuming the potential collision to take place at the destination nodes, i.e. ignoring effects of coinciding packet routes. Hence it is assumed that the burst will be delivered to the destination consecutively, i.e. it will not be split along its way. The validity of this assumption will be discussed in Subsection III - *D. The Moderating Effect on the Cost of Collisions.* Given these assumptions a *best case* scenario is derived as it comes to coinciding packet destinations.

To answer the question regarding the cost of collision burst size and the packet emission probability has to be known. In the case of not having bursts, i.e. the burst size is one; the potential cost will develop from one single scenario. Here, a packet denoted A competes with a packet denoted 1

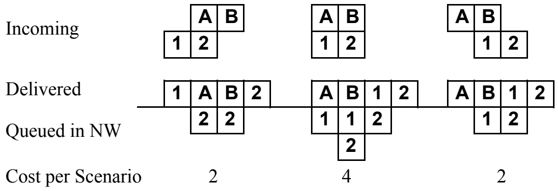


The cost per packet (cpp) in this single collision scenario is that one packet has to be queued per two incoming packets that collide. In the formula below the burst size is denoted b. To derive

$$\text{cpp} = \frac{1}{2 \cdot \text{b}} \cdot \sum \text{cps} = \frac{1}{2} \qquad \text{cps} = \text{CostPerScenario}$$

the average cost of having the same destination the emission probability, λ, is used to determine the interval length, i, in which a potential collision scenario will occur according to $\text{i} = \text{b}/\lambda$.

If the emission probability is 50% and the packets have the same destination, each packet will suffer an extra 0.25 clock cycle in delay due to competition for exit. If the burst size is increased, e.g. $\text{b} = 2$, the following three outcomes of packet collisions are possible.



In the general case, the total cost from all possible scenarios grows as b^3, which means that the average **c**ost of a pair-wise **c**ollision **p**er **p**acket (ccpp) can be reduced to:

$$\text{ccpp} = \frac{1}{2 \cdot \text{b} \cdot \text{i}} \cdot \sum \text{cps} = \frac{1}{2 \cdot \text{b} \cdot (\text{b} \div \lambda)} \cdot \text{b}^3 = \frac{\text{b} \cdot \lambda}{2}$$

In short – the cost for a collision increases linearly with the burst size! One important note to make here is that the average number of packets queued/in transit in the network per packet sent actually corresponds to an increased routing latency with the same amount of cycles.

C. The Combined Cost of Bursts

If the information about how often packets collide is combined with the relative cost of collisions a bound on the increase in latency due to the increased burst size is derived. Since it is only claimed to be a lower bound it is reasonable to

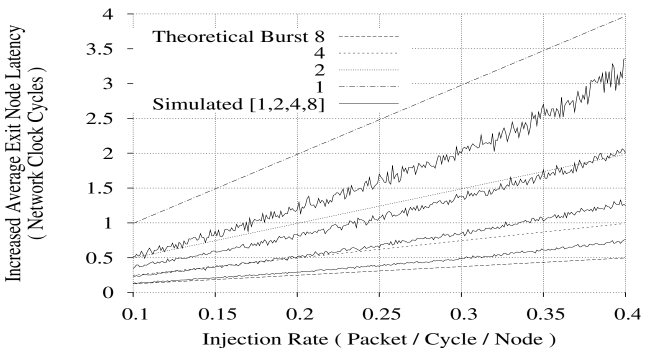


Fig. 3. "Theoretical Bound" on The Cost of Bursts

only consider all collisions to be pair-wise to simplify the analysis. For the 4x4 network the probability for a packet having a coinciding destination with another packet was 62%. For the Nostrum network, where switches are double buffered and deflection routing is employed, a deflection at destination means that the packet has to follow a minimal path of four hops before it can make a new attempt. This detour means that the penalty is quadrupled. If the penalty together with the relative number of packets competing is taken into the formula *cost of collision per packets* of Subsection III - B. it becomes:

$$\mathrm{ccpp}_{16} = 4 \cdot 0.62 \cdot b \cdot \lambda \div 2 = 1.24 \cdot b \cdot \lambda$$

The graph of Figure 3 shows a family of plots over the increased latency that originates from pair-wise deflections at the destination vs. the emission probability. Depicted in Figure 3 are also the simulated corresponding "real" increased latencies. These simulated incremental latencies are *only* the latencies observed at the exits of the network due to bursty traffic. Additional latencies due to the bursty traffic in other parts of the network are *not* included; hence the situation is actually worse! On the other hand, what is striking when examining the graph closer is the rather distressing fact that the simulation result actually is *lower* than the theoretical lower bound! The main reason for this anomaly is that the effect of the increased burst size at the destination nodes are moderated by an increased degree of deflection for the packets on their way to the destination, i.e. the preconditions of the analysis do not hold – more on this in next section. In the case of wormhole routing the cost of collision would be different since flits are not deflected but simply buffered.

D. *The Moderating Effect on the Cost of Collisions*

With an increased burst size a moderating effect on the cost of collision will manifest itself. The cause of this moderation has two components. The first component is that the burst is split up due to misrouting along a packet's way from source to destination. The second is caused by the senders' inability to get packets that belong to the same bursts, out on the network consecutively. The downside of the analysis of Fig. 2 is that our assumption about having the *continuous* bursts only colliding at the destination nodes does not hold since the burst obviously have been *spread out over time*! The positive thing is that this moderating effect of spreading the burst to could be further be exploited as *a positive side effect* to the Forced Requeue approach!

IV. Hardware Implementation

The biggest objection to the use of the Priority Based Forced Requeue is the cost of additional hardware. Any solution that is going to operate at realistic speed will involve shift registers. In our search for an acceptable solution there are two lucky circumstances. The first one is the fact that the shift registers do not have to store complete packets but the packet's individual priorities together with a reference to a memory position where that actual packet is stored. The second is that only requeued packets need to be sorted. The packets originating from the Resources hosting the queue is already sorted and hence, only the heads of the individual queues need to be compared. Several proposals have been suggested for implementing priority queues – one appealing to our needs is the *Sequencer Chip* originally developed for the *ATM traffic shaper* of Chao and Uzun [3]. The basic idea is to keep sorting keys in registers and in parallel compare any incoming element to all the keys to determine which packets that needs to be shifted. From simulation data it could be observed that the priority queue of the individual network interfaces never exceeded 60 elements despite quite substantial traffic loads. Hence, a reasonable assumption is that 8 bits are needed for the sorting queues and 6 bits for memory references. This gives us the possibility to administer 64 packets with a maximum latency of 256 cycles. Each module will contain 14 bit registers and some combinatorial hardware. The combinatorial hardware needed is approximately 20k gates for the full sequencer. The memories required for the requeued packets will not add any extra cost since it can be observed from the simulations in Section V. that regardless of whether the priority queues of the *Forced Requeue* is used or not, roughly, the same number of packets will compete for

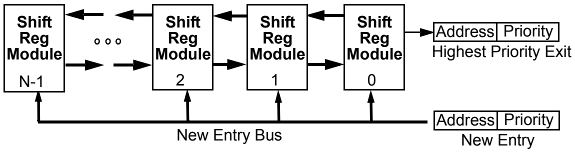


Fig. 4. Sequencer – Overview

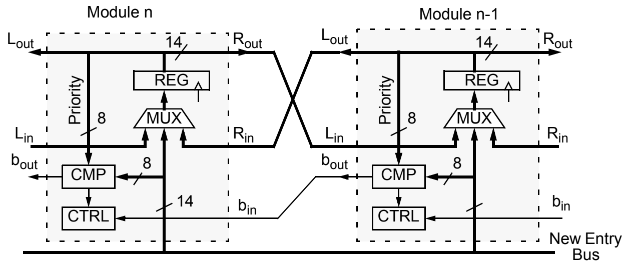


Fig. 5. Sequencer Module(s)

admission to the network. For a network using wormhole routing the cost is significantly less since only one time-tag per worm needs to be stored.

V. Simulation Results

Due to a limited space the result presented will focus on showing the decrease in observed worst-case latency. Regarding other aspects, such as average latency, throughput, etc. they are unaffected. In summary the network performance is claimed to be by no means worsened in any aspect due to the PBFR. The only effect observed is a reshaped latency curve for the delivered packets where the heavy tail is shortened and packets with low latency have a slightly increased latency but the *average* latency is kept constant.

A. Simulation Setup

Our cycle accurate simulator used is entirely written in SystemC. As mentioned before, all resources generate packets with a rate of λ packets per cycle. The experiments cover two different access patterns: (1) The Random Uniform Pattern (RUP) where all Resource nodes (R) are communicating with other nodes in the system with equal probability. (2) The Central Memory Pattern (CMP) implements an access pattern where the R is communicating with the centrally placed "memory" nodes (M) located in the centre of the chip. For both scenarios all nodes generate 2048 packets each, since it is enough to make the effect of start-up and empty phases insignificant. The parameter that is changed from one simulation to the next is the injection rate. For the RUP the λ increases from 0.100 - 0.400 in steps of 0.001. This gives 301 measurement points from a simulation run. For the CMP, the interval was 0.050 - 0.220 in steps of 0.001 giving 171 points. The lower range of CMP is due to the fact that the memory nodes are more heavily loaded. In some graphs not the full range of measurement points is presented to enhance the readability.

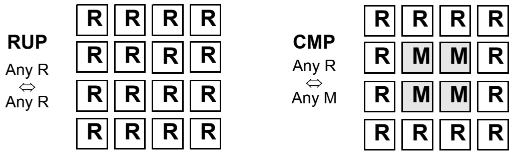


Fig. 6. Communication Patterns

B. Required Downstream Buffer Capacity

The individual sizes of all 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 time held 10 packets all packet queues of the network are given that size. In Fig. 7 it is seen that the total required buffer capacity is independent of whether PBFR is used or not. Also, the relative amount of traffic using the priority queues is depicted.

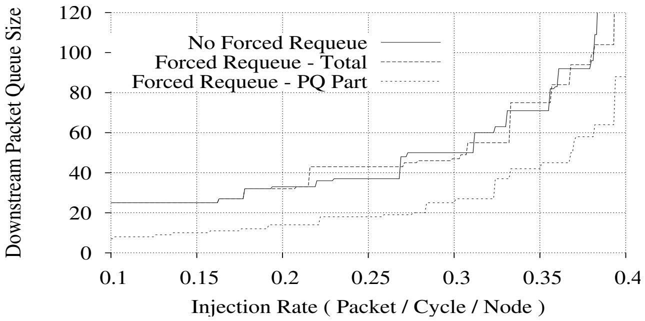


Fig. 7. Downstream Buffer Requirements

C. The Latency Distribution Shift

In order to understand what is happening with the latencies within one single simulation Fig. 8 is provided which is a histogram that depicts the latency. As can be seen the latency is shifted to the right when utilising Forced Requeue but with a shortened heavy tail but with average kept. To enhance the readability the graphs has been smoothed.

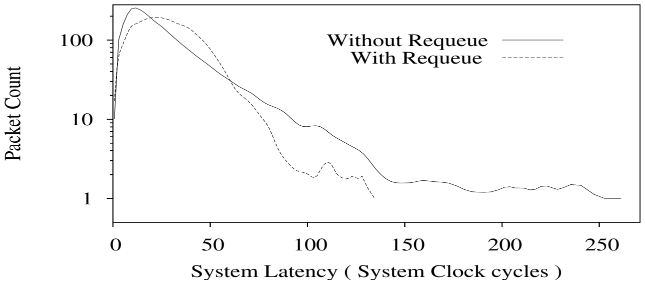


Fig. 8. Latency Distribution for λ = 0.400, burst = 8

D. Performance – Worst-Case Latency.

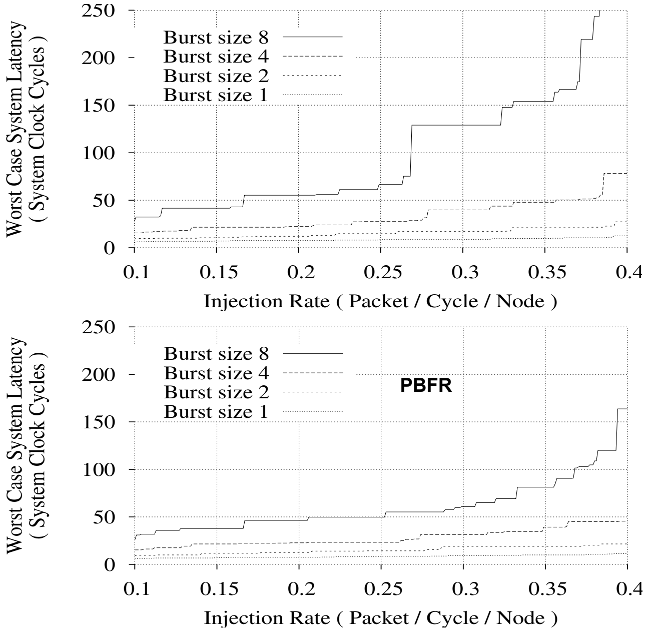


Fig. 9. RUP – Worst-Case Latency

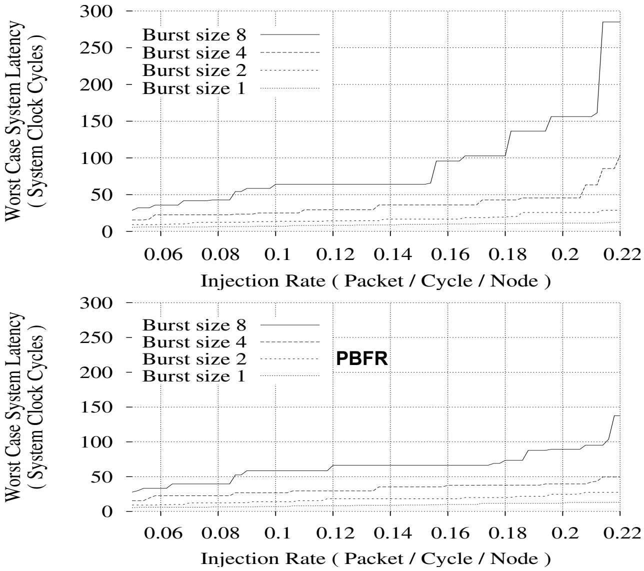


Fig. 10. CMP – Worst-Case Latency

In order to enhance readability the worst-cases in terms of latencies the graphs are made monotonously increasing, this is also done for the Fig. 7. As it can be seen in Fig. 9 (RUP) and in Fig. 10 (CMP), respectively the latency is roughly reduced by 50% if the PBFR approach is utilised.

VI. Discussion

For any NoC that implements *best effort* and is exposed to a random traffic pattern are bound to get packets competing for resources due to the phenomena described in Section III. The effects for routing techniques such as wormhole routing are basically the same. A worm denied exit from the network locks up resources in the network. These resources could in turn, potentially, lock up other resources and so forth. Hence, due to that the single biggest source of delay is the admission queuing time the Forced Requeue for routing techniques such as wormhole routing would be beneficial since tighter bounds on worst-case delay can be given. This has not yet been studied nor has the PBFR been implemented for wormhole routing.

A. *Nostrum - Full System Performances*

To keep the analysis clean from effects that comes from other improvements of Nostrum they are left out. If PBFR is combined with our Dual Packet Exit strategy [7] and a mild traffic regulation is used the worst-case latencies of Fig. 11 is presented and can be compared with Fig. 9.

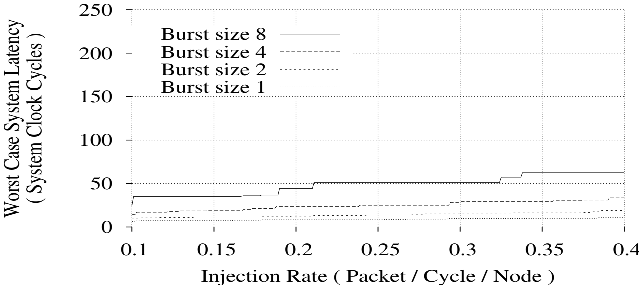


Fig. 11. Worst-Case latency with “All included”

VII. Conclusions

The concept of *Priority Based Forced Requeue* is presented due to that best effort services inherently have a problem giving statistical bounds on admission time to the network, i.e. bounds on the time a packet has to spend queuing before it can enter the network. The PBFR both reduces worst-case latencies as well as the harmful effects due to bursty traffic in the network. The contribution is the idea that low priority packets/ worms can be taken out from the network before they actually reach their final destination to be resent later.

In order to give the reader an intuition about the harmful effect of bursts a model for giving an estimate on the effect of bursts is developed. The model shows that the cost in terms of extra delay is linearly dependent of the burst size.

From simulation data we claim that there are no performance degrading penalties related to the use of PBFR. However, there will be an extra cost in hardware and an implementation based on shift registers is proposed. As can be observed in simulation; using the PBFR approach will reduce the worst-case latencies by 50% while still using the same number of buffers! This is demonstrated both for a uniform traffic pattern as well as for a traffic pattern constructed to create hot-spots in the centre of the NoC. The generality of the approach is discussed and it is claimed that the performance improvements will exist in other networks than Nostrum as well.

References

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

[2] J. T. Brassel, “Deflection Routing in Certain Regular Networks”, Ph.D. Dissertation, University of California (1991)

[3] H.J. Chao et al., A VLSI sequencer chip for ATM traffic shaper and queue manager, *IEEE J. Solid-State Circuits* 1992

[4] 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.

[5] H. B. Hajek, “Bounds on Evacuation Time for Deflection Routing,” *Proc. CISS* (March 1989).

[6] T. Lang, L. Kurisaki: Nonuniform Traffic Spots (NUTS) in Multistage Interconnection Networks. *IEEE Journal of Parallel and Distributed Computing, 10(1): 55-67* (1990)

[7] M. Millberg, E. Nilsson, R Thid, A. Jantsch. “Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum NoC.” *DATE* (2004)

[8] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch. “The Nostrum backbone – a communication protocol stack for networks on chip” *Proc. VLSI Design Conf., India* (2004)

[9] E. Nilsson et al. “Load distribution with proximity congestion awareness in NoC”. *DATE* (2003)

[10] V. Tota, M. R. Casu, L. Macchiarulo, “Implementation Analysis of NoC: A MPSoC Trace-Driven Approach”, pp. 204-209, in *Proc. of ACM Great Lakes Symposium on VLSI* (2006)