Effects of Injection Pressure on Network Throughput

C. Izu[1], J. Miguel-Alonso[2], J.A. Gregorio[3]

[1]*School of Computer Science, The University of Adelaide, Australia*
cruz@cs.adelaide.edu.au

[2]*Department of Computer Architecture and Technology, The University of the Basque Country, Spain*
j.miguel@ehu.es

[3]*Computer Architecture Research Group, Universidad de Cantabria, Spain*
joseangel.gregorio@unican.es

Abstract

Recent parallel systems use multiple injection ports and various injection policies, but little is known about their impact on network performance. This paper evaluates the influence that these injection interfaces have on maximum sustained throughput in adaptive cut-through torus networks by modeling the number of injection queues (1 or 4), and the allocation of new packets to those queues.

Network evaluations for medium to large size 2D tori show that designs with multiple injection ports do not improve performance under uniform traffic. On the contrary, they result in more pressure from the injection interface to acquire the scarce network resources of an already clogged system. Interestingly, for small networks, a single injection FIFO queue, with the HOLB it entails, indirectly provides the much needed injection control. For networks with thousands of nodes and multiple injection channels, as those being implemented in current massively parallel processors, this implicit form of congestion control is not enough. In such systems, restrictive injection policies are required to prevent routers from being flooded with new packets for loads beyond saturation.

1. Introduction

The interconnection network (IN) is a key element of a parallel computer system, providing a high-bandwidth and low-latency communication medium. Networks of this kind are now found in many systems to offer not only inter-processor communication or processor-memory interconnect, but also input-output and storage switches, or replacing dedicated wiring.

Networks throughput is often limited by message contention. Therefore, a large body of IN research has focused on reducing contention by increasing the number of requests made by incoming packets: adding virtual channels (*VCs*) [8], providing adaptive routing [6] or both [10]. As silicon area is less of a premium nowadays, the latest routers use virtual cut-through (*VCT*) flow control with large buffers to reduce contention at medium to high loads. For example, the Alpha 21364 router can store up to 316 packets [14].

Most implemented networks, from the Torus Routing Chip [9] to the Cray T3E Network [17] offered a single full-duplex connection between the router and its processing element, with a single injection port. Network performance is usually evaluated using register-level simulators that model the network interface as a single FIFO queue where new packets wait to be transferred into the router's injection port [5, 6, 15]. The impact on network performance of most router design parameters is reasonably well understood. For example, it is known that a small number of virtual channels, in the range 2 to 4, increases throughput by reducing head-of-line blocking (*HOLB*), regardless of the routing strategy applied.

As the standard processing elements of multiprocessors are replaced by multithreaded ones or by chip multiprocessors, both the number of injectors and the total offered load per node increases. For example, the Alpha 21364 router has 4 local ports [14] and the BlueGene/L torus network has 8 injection ports per router [4]. Having multiple ports is beneficial for local and multicast (or broadcast) communication patterns, as it allows the simultaneous injection of packets in each network direction, but little is known about its impact on point-to-point traffic. We have modeled a range of injection interfaces and explored the impact that some characteristics of these interfaces, the number of injection ports in particular, have on maximum sustained throughput. This study aims to close the gap between the interfaces of massively parallel systems currently being built (using torus networks), and the knowledge of adaptive *k*-ary *n*-cube networks available in the literature.

Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06)
1066-6192/06 $20.00 © 2006 IEEE

To the best of our knowledge, no other work has analyzed the impact of multiple injection ports on peak performance. Basak and Panda [1] analyzed the consumption bottleneck in wormhole tori, by increasing the number of consumption channels. That study covered a totally different design space: it considered small radix wormhole networks, in which throughput is limited by the node-to-router bandwidth, and it focused on its impact on network delay. Even though they mentioned that for symmetry they increased injection bandwidth, there is no description of how this bandwidth is used. In [11], Duato et al. briefly reviewed the effect of multiple injection ports in an adaptive wormhole router, for which adding more ports improved both latency and throughput. However, their plots did not include data for injected loads beyond saturation "*for the sake of clarity*", and the policy used to allocate packets to ports was not described. Recent works, such as [18], propose the use of multiple injection queues, one per destination, to obtain backpressure information and re-direct traffic over less congested areas. However, in the experimental setup authors assume that each node has an infinite source queue that models the network interface, thus modeling a single injection port per node; little is said about the mapping of those multiple queues into the injection port.

Our work will show that a single queue, with the HOLB it entails, does not limit network throughput under uniform traffic. On the contrary, it prevents network performance from further degrading under heavy loads by providing partial congestion control. Besides, this study highlights the need to throttle injection in order to sustain peak performance in large radix networks.

The paper is organized as follows: Section 2 describes different injection interface designs. Section 3 describes the evaluation methodology. Section 4 presents simulation results for two adaptive VCT routers with a range of interface designs. Finally, Section 5 summarizes the findings of this work.

2. Injection interface design

The network interface provides queuing between the router and the computing node(s). Figure 1 (top) shows a generic router for 2D networks, with a single injection port. The packet at the head of the queue will advance to the router's injection port and request one or more output links. Normally, new and in-transit packets are governed by the same rules: they can advance provided that the flow control allows it and a connection can be established to the selected output port.

We can see in this figure that, while the –X output is free, the second packet in the injection queue, with header $(\Delta x,\Delta y) = (-3,-1)$, is blocked due to the contention in the +X direction, which prevents the first packet (with header (+4,–2)) from being injected.

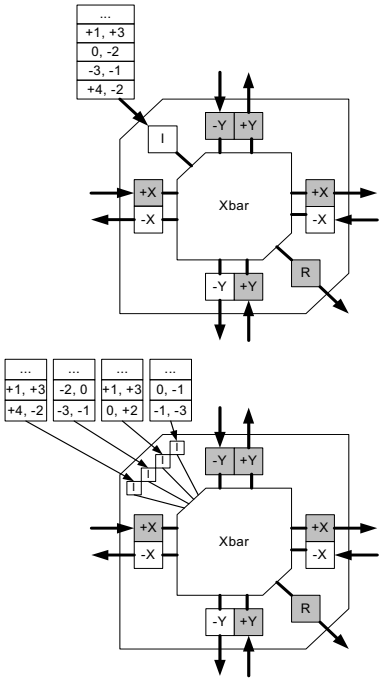

Figure 1. Two organizations for the injection interface: single injector (top) vs. four injectors (bottom). Packets at injection queues are represented by their routing records. Grey shade indicates ports are busy.

In fact, for large *k*-ary *n*-cube networks, this HOLB has been identified as the reason why some network resources were under-utilized [13]. Under uniform loads, such networks exhibit an unbalanced use of their +X and –X buffers. This is explained by the fact that the first set of channels to saturate, lets assume it is +X, fills its buffers and stops accepting new packets, while the other direction (–X), is prevented from receiving new packets until the blocked packet at the head of the queue has been injected into +X.

Figure 1 (bottom) shows an alternative design to eliminate HOLB: we need 4 queues and their associated ports (or *2d* queues for a *d*-dimensional router) to eliminate *all* possible cases of HOLB at the injection interface.

We need a policy to select, for each packet, in which injection queue it will be stored. As this is a pre-routing decision, it can be either static or dynamic. We have considered the following ones:

- *Shortest (**sh**)*. A simple dynamic policy: a new packet is allocated to the less-populated queue.

- *Shortest with pre-routing (**shp**)*. Each injection queue is assigned to a different output direction, as shown in Figure 1 (bottom). Each packet can go to any queue in the direction it travels. For example, the packet with header (–3,+1) can go to either the –X queue or the +Y queue. From this set, the less populated one is selected.
- *Longest path first (**lpath**)*: each packet is allocated to the queue associated to the direction in which it will traverse a longer path. A packet with header (–3,+1) will be placed on the –X queue; this is a static decision.

If the selected injection queue is full, the generation of packets stalls until the packet is successfully queued. This means that, above saturation, the load actually generated may be smaller that the target load.

Note that, in Figure 1, a physical channel connects each injection queue to its port, so that the injection bandwidth will grow with the number of ports. As the interconnection links of any router are a limited resource, the bandwidth devoted to the transit network links could be constrained by this decision. To avoid this, the injection interface can use virtual channels to connect its multiple queues to their associated ports. This choice depends on technological and implementation factors, such as if the router is part of an integrated chip or a separate chip, package constrains, etc.

3. Evaluation procedure

In order to evaluate the various injection interface designs, we have used a cycle-driven register level simulator [16] to model the router architecture depicted in Figure 2.

We have evaluated the two alternatives described in the previous section for the injection interface: a dedicated physical channel per injection port (as shown in the figure) vs. a single physical channel shared by a collection of (virtual) injection ports. Note that the internal router architecture is the same in both cases, having an injection input buffer per port or virtual channel; the only difference between them is the number of packets the processing node can send to the router core at a time: 1 or 4.

The injection bandwidth will have little impact on network performance for most point-to-point traffic patterns, although it will expedite local and collective communication. For example, a 16x16 torus with virtual injection ports can inject up to 1 phit/cycle/node compared to 4 phits/cycle/node when using 4 physical injection ports. In both cases, the network bisection limit is 0.5 phits/cycle/node. Therefore, sharing a single injection channel won't affect throughput under random traffic. Simulations show that it only increases latency at medium loads by less than 3%. Thus, this paper will present results only for the case of 4 physical injection ports, although the insights it provides are also valid for virtual injection channels.

Figure 2 also shows that the consumption interface can receive several packets (from different input ports) simultaneously, so that consumption bandwidth is never a bottleneck.

This router architecture provides adaptive routing by following Duato's approach [12]: a subset of virtual channels is configured as a safe virtual network, also called the escape network, in which packet deadlock never occurs. The remaining VCs are configured as a minimal adaptive virtual network (routing is adaptive, but using only minimum distance paths). Packets can move freely from the safe to the adaptive network, and channel request policies do favor the use of adaptive VCs in the current dimension. If none is available, an adaptive VC in another profitable dimension may be requested. The escape network is used as the last resource. When several input channels (including injection channels) request the same virtual output, a random arbitration policy is used.

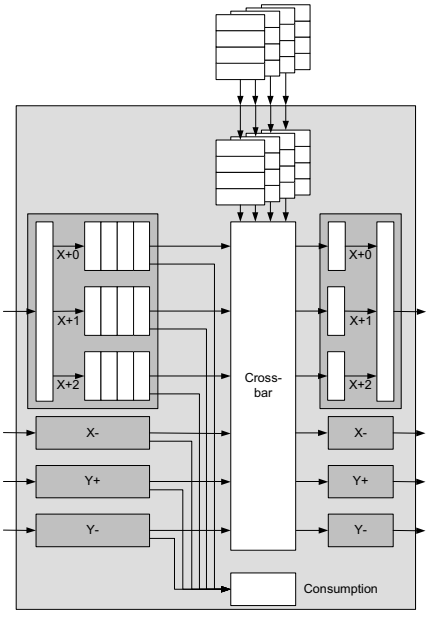

Figure 2. Adaptive VCT router architecture with 4 injection ports. The number of physical injection links (and ports) may be either 4 (as shown) or 1.

Both the Alpha 21364 and the BlueGene/L torus networks are instances of this architecture. They differ in their deadlock avoidance mechanism:

- In the ***Classic*** router (alike the Alpha 21364 and the Torus Routing Chip) two VCs are used for the escape sub-network, while the rest (in this case, just one) implement the adaptive sub-network. In the escape sub-network, packets follow static paths as per DOR routing, and deadlock is avoided as in [9].
- In the adaptive ***Bubble*** router (alike the BG/L torus network), just one VC is used to form the escape sub-network, and the rest (in this case, two) implement the adaptive sub-network. In the escape sub-network, packets follow static paths as per DOR routing, with the injection regulated by bubble flow control [15], which prevents the node from injecting a packet if such action exhausts the local buffer capacity in the direction/way (that is, in the ring) it is advancing. Packets turning from an X-ring to a Y-ring inside the escape virtual network are considered as new injections into the receiving Y-ring. Packets in transit inside a safe ring are regulated by VCT. Note that injection into the escape sub-network is banned if there is only one free buffer, as per bubble condition.

We deal only with packets of fixed size: 16 phits. A phit is the number of bits that are conveyed in parallel through a physical link (note that in a VCT router a packet is the minimal unit of flow, or flit). Each input queue has capacity for 8 packets, or 128 phits. Injection queues have an additional buffer capacity of 128 phits. Two network sizes are considered: 256 nodes (16x16), which is a medium size commonly used in network evaluation, and 1024 nodes (32x32), to represent larger systems.

It is common to evaluate performance under a range of synthetic traffic patterns, such as uniform, hot-spot/region, and permutations. In this paper, we have chosen depth instead of breadth, by providing detailed results of performance beyond saturation under a single pattern: uniform traffic. This allows us to focus on any given router to study how congestion builds up inside the network. As it makes an even use of resources, the network is able to reach higher throughput than with other synthetic patterns that have less balanced resource utilization. We have checked that our findings are still valid for non-uniform patterns, such as permutations: the routers that reach saturation first exhibit the same behavior of a congested router under uniform traffic, whereas routers in less busy areas behave as if they were below saturation.

Many studies normalize their loads to the network bisection limit [14]. For 16x16 and 32x32 torus networks, this limit is 0.5 and 0.25 phits/cycle/node respectively. On intensive communication phases some parallel applications may send tens of packets in a burst, well above the theoretical limit. Besides, we are interested in observing the behavior of a saturated network. Thus, in most of our experiments we apply a load of 1.0 phits/cycle/node, which for uniform traffic is 2 or 4 times the network bisection limit.

4. Network evaluation

This section presents the results obtained for adaptive 2D torus networks with different injection subsystems. Firstly, we evaluate the impact that the number of injection queues and the allocation policy have on the performance of a 256-node torus network of *Bubble* routers. Secondly, we analyze their impact when the network size increases to 1024 nodes. Lastly, we perform similar evaluations with a network of *Classic* routers.

4.1 Medium-size *Bubble* network

Figure 3 shows accepted load (throughput) versus offered load in a 16x16 torus of *Bubble* routers for the following interfaces: *1_Inj* (one injector), *4_Inj_sh* (4 injectors, shortest selection policy) *4_Inj_lpath* (4 injectors with *lpath* selection policy) and *4_Inj_shp* (4 injectors with *shp* selection policy). Note that curves for *1_Inj* and *4_Inj_lpath* are almost identical, and the same happens with *4_Inj_sh* and *4_Inj_shp*. When using one injector, or 4 with the *lpath* policy, the network reaches a peak throughput close to the bisection limit, and is able to keep it for loads beyond that saturation point. However, in the other two cases, we observe an important drop in performance once the saturation point has been surpassed.

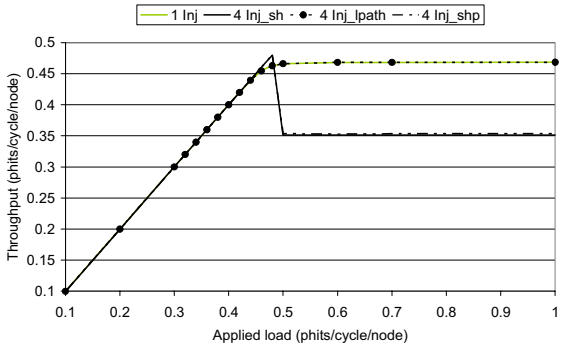

Figure 3. Throughput vs. offered load for a 16x16 adaptive *Bubble* torus with different injection interfaces.

We hypothesize that this drop is due to the lack of congestion control. To verify this, we have plotted in Figure 4 queue occupation and channel utilization under 1 phit/cycle/node applied load (when network is saturated). Plots for *4_Inj_shp* are not included, because they are identical to those of *4_Inj_sh*.

Observe that, for those cases without performance drop (*1_Inj* and *4_Inj_lpath*), most packets use the adaptive sub-network. Physical channel utilization[1] is 95%, and average occupation of queues is low. In these cases, the usually harmful HOLB of the single port plays an important role as an indirect way of throttling injection—and, therefore, preventing congestion.

The scenario for *4_Inj_sh* (and *4_Inj_shp*) is drastically different: population of the adaptive –X queues is close to their 128-phit capacity, and most packets advance in that direction using escape VCs. Congestion in the adaptive sub-network spreads to +Y and –Y. The +X direction is not congested: buffer occupation is low and most packets use adaptive VCs.

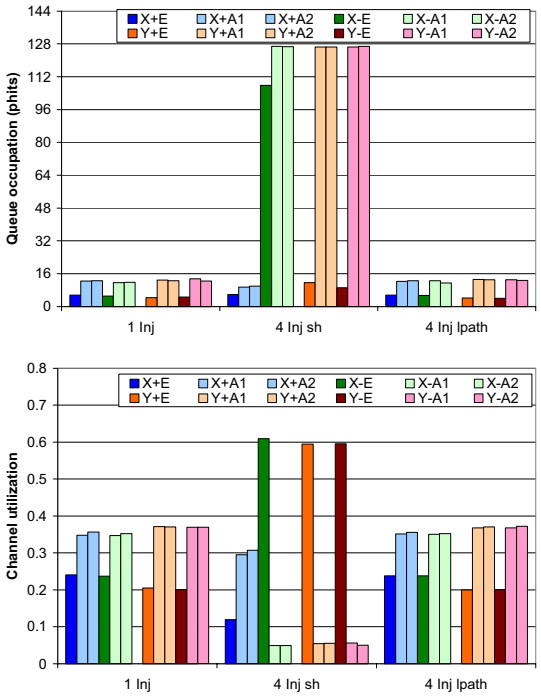

Figure 4. Transit queue occupation (top) and channel utilization (bottom) in a 16x16 torus at 1 phit/cycle/node applied load, for 3 alternative injection interfaces.

We may understand better how congestion builds up by looking at the buffer occupation in the X dimension over time for both *sh* and *lpath* policies, as shown in Figure 5. In the case of the *sh* policy (top), after a warm-up period, all the injection ports contain packets requesting the same output, –X in this case. The adaptive network gradually fills up its transit queues and injection pressure moves to its escape network. This explains why the adaptive channel utilization (except for +X) drops dramatically when routers are flooded with new packets, leading to the throughput loss shown in Figure 3.

Meanwhile, buffer occupation in the opposite direction (+X) falls, illustrating the asymmetry in resource utilization discussed in [13]. The degree of congestion in –X is so deep, due to the excessive pressure from packets at the 4 injection ports, that the system is unable to break out from it. Meanwhile, new packets going towards +X must wait at the injection queues; when finally injected, they can easily reach their destination using mainly the adaptive channels.

Under *lpath* policy (Fig. 5(bottom)), packets traveling on +X and those traveling on –X use different injectors. Thus, several injection ports cannot impose constant pressure over the same axis/direction. A blocked packet in one port (for example, a packet in the +X queue) prevents packets in the same class to access to the network (reducing congestion in that axis) and, when that queue is full, forces the node to stop new injections. This explains why channel and buffer utilization keeps at reasonable levels: around 3 packets per physical channel. Additional buffer space is still necessary, to cope with traffic fluctuations over time.

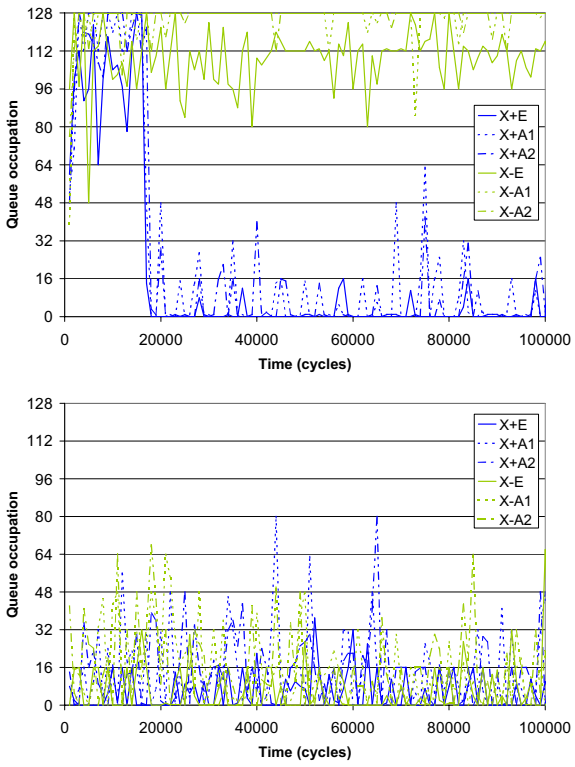

Figure 5. Transit X queue's occupation over time in a 16x16 torus with 4 injection ports, for *sh* (top) and *lpath* (bottom) port selection policies.

In short, HOLB at injection prevents congestion in the 256-node network. This explains why most studies

[1] Aggregated utilization of all the virtual channels that share a physical channel.

of adaptive torus networks did not exhibit performance degradation at high loads. Adding more injection ports requires an adequate selection policy so that the adaptive network is not flooded with new packets beyond its saturation point.

4.2 Large-size *Bubble* network

This section evaluates the impact that the injection interface has on the performance of a 32x32 torus. Although is common for large direct networks to arrange nodes in three dimensions, our experiments reflect realistic designs such as the BlueGene/L, whose rings can be even larger that the ones we test here.

Figure 6 shows the throughput versus offered load for this network with different injection organizations. In this large network, all configurations achieved nearly 100% peak throughput but then exhibited a significant loss (around 20 %) beyond saturation.

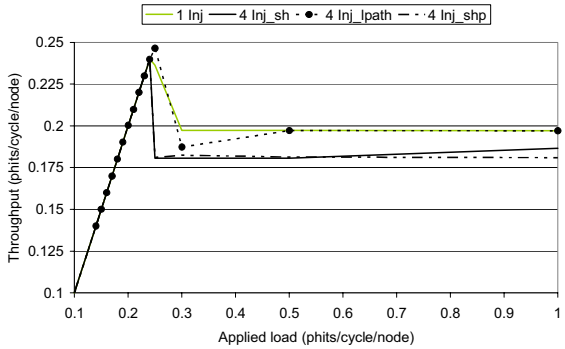

Figure 6. Accepted versus offered load for a 32x32 adaptive *Bubble* torus with 1 (light line) or 4 injection queues and different injection port selection policies.

Buffer occupation and channel utilization (not shown due to limited space) exhibits a usage pattern similar to that seen for *4_Inj_sh* in Figure 4 for all the interfaces. Such pattern indicates network congestion soared for loads above saturation (in fact, their transit queue occupations along the time behave as shown in Figure 5 (top)). As before, the performance drop after saturation is accompanied by an asymmetric use of network resources in the X dimension.

In large networks, the injection policy has little impact on sustained performance; packets travel longer paths, and more nodes contribute to the saturation of a given link. At the same time, network backpressure takes longer to reach its sources. Thus, the restrictive effect of HOLB arrives too late, when network buffers are already (almost) full.

4.3 *Classic* network

The reader may wonder if the conclusions from the previous subsections apply only to *Bubble* routers or are valid for most VCT networks. To clarify this, we have extended this evaluation to networks built using *Classic* routers.

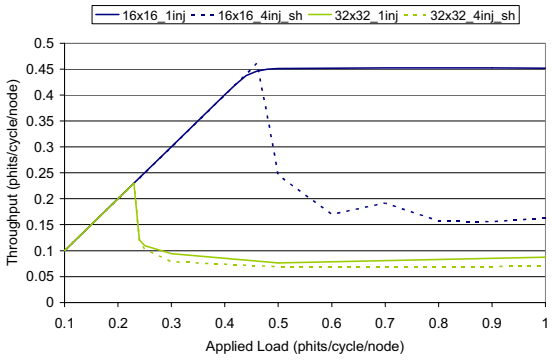

Figure 7. Throughput vs. applied load for 16x16 and 32x32 *Classic* networks, with 1 or 4 injectors.

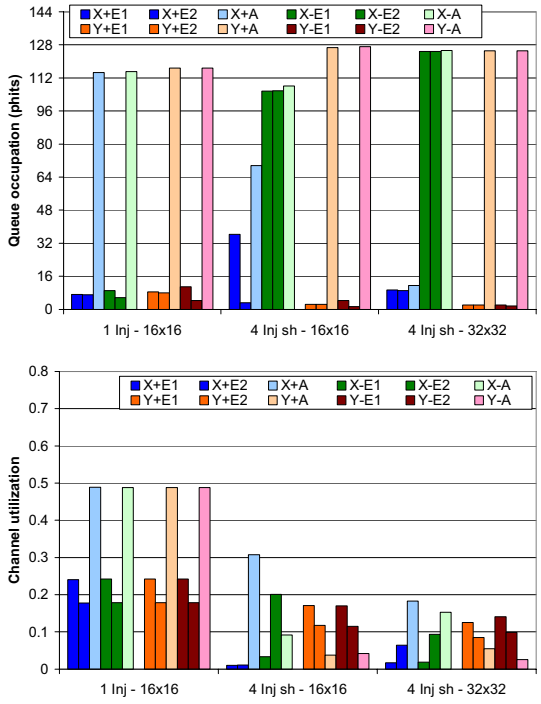

Figure 8. Transit queue occupation (top) and channel utilization (bottom) for a network of *Classic* routers with different injection interfaces.

Figure 7 shows throughput versus offered load for two *Classic* network sizes with 1 or 4 injectors. For clarity, only the *sh* policy is shown, but other policies behaved as discussed in the previous sections. Peak throughput is above 0.45 for the medium network, and

close to 0.25 for the large network—that is, at loads near their bisection bandwidth limits. After saturation, throughput stays high only for the 16x16 network with one injector. In the other three configurations, throughput falls severely, even more than in the case of their *Bubble* counterparts (compare Fig. 7 with Figures 3 and 6).

Figure 8 shows queue occupation and average channel utilization for a set of representative configurations, always at 1 phit/cycle/node applied load. With 1 injector, channel utilization is high, split evenly between the escape and the adaptive sub-networks, although the adaptive queues are quite full. When multiple injectors are added, the –X direction saturates first, and all three virtual channels are congested. This is reflected on their high buffer occupation and low channel utilization. This congestion spreads as well to the adaptive channels +Y and –Y. On the other hand, the rest of the escape network (all but –X) is underused. In other words, the *Classic* router suffers severely from network congestion because the first channel to saturate does it in both its escape and adaptive channels. The larger the network, the higher the congestion and the greater the throughput drop.

4.4 Throttled injection

Results of experiments show that both the adaptive *Bubble* and the *Classic* VCT networks suffer a sharp throughput drop after the saturation point is reached, due to network congestion. As we have observed in the previous section, the throughput loss for *Bubble* is minor compared to *Classic*. This is because of the constraints set by bubble flow control, which prevents new packets from flooding the escape sub-network. In this section we explore the effect that such throttling mechanism has on the adaptive sub-network.

Although congestion control is a well-known issue in computer networks, there are only a handful of papers in the context of direct interconnection networks. This could be explained by the *implicit* but unknown congestion control provided by single injection queues in small and medium systems. Only wormhole routers using virtual channels at injection exhibited minor throughput loss and different throttled injection methods were proposed [10, 2, 3]. None of them consider the impact that their interface design has on network congestion. Besides, they used small input buffers which reflected technological constrains that do not apply anymore. NIFDY [7] is an injection interface that restricts injection based on packet destination, not in the network status. In fact, it attempts to reduce end-to-end congestion by relying on acknowledgements from the receiver's end, which is an issue different to the internal network congestion that causes the drop of performance. The Chaos router [5] deals with congestion by both misrouting packets, and giving blocked packets at its central queue priority to advance ahead of new or incoming packets. Commercial routers such as the Alpha 21364 [14] and the BG/L torus network [4] can also give priority to transit traffic over new injections.

It is important to remark that our main purpose is to show the benefits of throttling injections as a means to control congestion, and that the extension of the bubble mechanism to this purpose is provided just as an example: other restrictive mechanisms can be equally, or even more, helpful.

In a *Bubble* router, injection into an escape channel is not allowed unless there is room for 2 or more packets. Let us call this restriction *2-Esc*. We can apply this rule to packets moving from any injection port into any virtual output link: let us call it *2-All*. If we do so, in-transit traffic will have priority over new packets to use the last free buffer spaces on the adaptive sub-network. We can go further and limit the network population by increasing the number of free resources required to grant injection from the minimum of 2 packets in the local queue to 4 (*4-All*), or even 6 (*6-All*). Figure 9 shows the impact that these policies have on maximum sustained throughput in a 32x32 *Bubble* torus, with either 1 or 4 injectors.

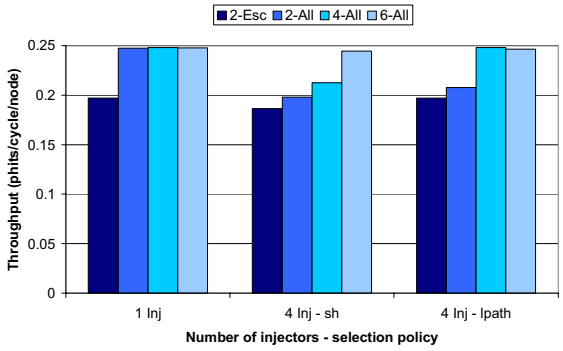

Figure 9. Sustained throughput at saturation, in 32x32 torus network, with and without for different degrees of bubble-based restrictive injection.

As expected, throttled injection reduces buffer occupation and increases channel utilization in all cases. All of them sustain maximum performance, very close to the bisection limit (0.25 phits/cycle/node). For the single-injector case, a bubble of size 2 is enough to prevent congestion in the adaptive network. On the contrary, *4_inj_lpath* with the same bubble size still exhibits asymmetry between the +X and –X, for both channel usage and buffer occupation. To curb congestion in the adaptive network, we must increase

the bubble size to 4. This reduces buffer occupation in all +X transit queues, which is the direction that saturates first in this set of experiments, and balances the use of the three virtual channels.

5. Conclusions

This work has provided an insight into the impact that the number of injection channels and their management has on the sustained performance of adaptive VCT networks of medium to large size.

Results show that for small and medium size networks, the HOLB of a single injection queue is a blessing in disguise, as it prevents the processing node from flooding the router in *all* directions. Once a channel is saturated, and a packet is blocked at the head of the injection queue, channels in other directions have a chance to drain their load. In larger networks, this is not enough to prevent new packets from accessing the scarce network resources, so that network congestion rises and channel utilization drops.

Adding injection ports does not increase network throughput for point-to-point traffic. Instead, it increases the injection pressure at high loads and allows the nodes to flood their routers with packets, leading to higher congestion and significant throughput loss at heavy loads. Large networks (and/or routers with multiple injection ports) need to throttle injection in order to sustain maximum throughput. A simple, local congestion control mechanism can improve throughput beyond saturation up to a 20-25%.

As parallel systems grow larger and interfaces add more injection ports, congestion control is becoming a critical issue on network design. Further research is required to confirm our results under a larger range of traffic patterns (including those generated by actual applications), and to find simple and effective injection policies that are starvation free and provide maximum sustained throughput.

6. References

[1] D. Basak and D.K. Panda. “Alleviating Consumption Channel Bottleneck in Wormhole-routed k-ary n-Cube Systems”. IEEE Trans. on Parallel and Distributed Systems, vol. 9, no. 5, pp. 481-496, 1998.

[2] E. Baydal and P. López. “A Robust Mechanism for Congestion Control: INC”. Euro-Par 2003: 958-968

[3] E. Baydal, P. López and J. Duato. “A Simple and Efficient Mechanism to Prevent Saturation in Wormhole Networks”. IPDPS 2000: 617-62

[4] M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burrow, T. Takken, P. Vranas. “Design and Analysis of the BlueGene/L Torus Interconnection Network” IBM Research Report RC23025 (W0312-022) December 3, 2003.

[5] Bolding, M. L. Fulgham, L. Snyder, “The Case for Chaotic Adaptive Routing”. IEEE Trans. Computers 46(12): 1281-1291 (1997)

[6] R.V. Boppana and S. Chalasani, “A Comparison of Adaptive Wormhole Routing Algorithms.” 20th Annual Int'l Symp. on Computer Architecture (ISCA), pp. 351-360, May 1993.

[7] T. Callahan and S.C. Goldstein, “NIFDY: A Low Overhead, High Throughput Network Interface”, in Proc. 22nd Annual Int. Symp. on Computer Architecture (ISCA), June 2005, Santa Margherita Ligure, Italy.

[8] W.J. Dally: “Virtual Channel Flow Control”, IEEE Trans. on Parallel and Distributed Systems, vol. 3, no. 2, pp. 194-205, 1992.

[9] W.J. Dally, C.L. Seitz, “The Torus Routing Chip” Distributed Computing vol 1 pp. 187-196. 1987

[10] W.J. Dally and H. Aoki, “Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels”, IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 4, pp. 466-475, 1993.

[11] J. Duato, S. Yalamanchili and L. Ni, “Interconnection Networks: an engineering Approach. Revised Printing”, Morgan Kaufmann, 2003.

[12] J. Duato. “A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks”. IEEE Trans. on Parallel and Distributed Systems, vol. 7, no. 8, pp. 841-854, 1996.

[13] J. Miguel-Alonso, J.A. Gregorio, V. Puente, F. Vallejo and R. Beivide. "Load Unbalance in k-ary n-cube Networks". Lecture Notes in Computer Science, 3149 / 2004 (Proc. Euro-Par 2004), Pages 900-907.

[14] S. Mukherjee, P. Bannon, S. Lang, A. Spink and David Webb, “The Alpha 21364 Network Architecture”, IEEE Micro v. 22, n. 1 pp 26-35, Feb. 2002

[15] V. Puente, C. Izu, J.A. Gregorio, R. Beivide, and F. Vallejo, ”The Adaptive Bubble router”, Journal on Parallel and Distributed Computing, vol 61, no. 9, pp.1180-1208 September 2001.

[16] F.J. Ridruejo, J. Miguel-Alonso. "INSEE: an Interconnection Network Simulation and Evaluation Environment". Lecture Notes in Computer Science, Volume 3648 / 2005 (Proc. Euro-Par 2005), Pages 1014 - 1023.

[17] S. L. Scott and G. Thorson, “The Cray T3E networks: adaptive routing in a high performance 3D torus”, Proc. of Hot Interconnects IV. 1996

[18] A. Singh, W. J. Dally, A. K. Gupta, B. Towles, “Adaptive channel queue routing on k-ary n-cubes”. SPAA 2004: 11-19.