Impact of Network Fairness on the Performance of Parallel Systems

Cruz Izu
The University of Adelaide
Adelaide, Australia
cruz.izu@adelaide.edu.au

Enrique Vallejo
University of Cantabria
Santander, Spain
enrique.vallejo@unican.es

ABSTRACT

Execution time of parallel applications depends on the balanced execution and synchronization of all its processes. However, most interconnects exhibit significant throughput unfairness, introducing load unbalance. At high loads, such unfairness significantly degrades the performance of some nodes, and eventually the whole system. Different strategies have been developed to guarantee network fairness, fundamentally based on two different approaches: traffic prioritization in the network switches or injection throttling at the source. This work qualitatively and quantitatively compares two fairness mechanisms which are representative of these two types of explicit fairness strategies: *Age arbitration*, which prioritizes old packets in transit, and the *SAT* injection protocol, which throttles traffic to provide even access to all nodes. Multiple synthetic traffic patterns as well as parallel application loads at different transmission rates are considered in the evaluation.

Our research identifies key limitations of each fairness mechanism: a realistic implementation of Age arbitration fails to achieve perfect fairness due to a delayed assignment of packet timestamps, while SAT may restrict maximum throughput when it is not necessary to do so. Both mechanisms do not hinder performance at low loads, and improve throughtput fairness at congested loads. With real applications, performance improvements up to 14.2% are observed, with SAT being particularly efficient at high loads.

CCS CONCEPTS

• **Computer systems organization** → **Interconnection architectures**; System on a chip;

KEYWORDS

interconnection networks, node throughput, network fairness, system performance

ACM Reference Format:
Cruz Izu and Enrique Vallejo. 2019. Impact of Network Fairness on the Performance of Parallel Systems. In *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '19), January 29–31, 2019, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3290688.3290706

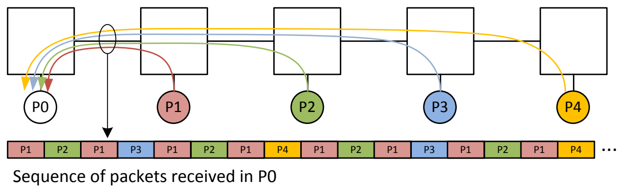

P0
P1
P2
P3
P4
P1 P2 P1 P3 P1 P2 P1 P4 P1 P2 P1 P3 P1 P2 P1 P4 …
Sequence of packets received in P0

Figure 1: Unfairness in a network using round-robin arbitration. When all nodes send traffic to P0, P1 gets half of the bandwidth.

1 INTRODUCTION

The performance of a parallel system depends on the balanced execution and synchronization of all its processes. This balanced execution relies on each process experiencing a similar access to the system resources; that balanced access to system resources is denoted *fairness*.

In a fair network, nodes that have similar communication demands should experience the same throughput. For example, if 4 nodes send packets to the same destination at full rate, as shown in Figure 1, each of them should get 25% of the bandwidth. We should note that locally-fair router arbitration mechanism does not result in a globally fair scheduling [3]. In the example above, P1, as is closest to the destination node, gets half of the available bandwidth, P2 gets 1/4 of the bandwidth and P2 and P3 get 1/8. In a large network, throughput differences between nodes could be large. In fact, reporting only average performance actually masks system unfairness, which has been measured to be significant at heavy loads in both direct and indirect INs [20]. Without a proper fairness mechanism, the service received by different sources is highly dependent on their network location, introducing an unbalance which could reduce performance in parallel applications.

Fairness mechanisms is often *implicit* to congestion control mechanisms that rely on variable-sized congestion windows, implemented in a per-flow [21, 29] or per-node [1, 35] basis. However, in many cases congestion control is not used in parallel applications for different reasons, among them: *i)* in practice, in situations without victim flows, congestion control introduces performance penalties [16]; *ii)* adaptive routing, which mitigates congestion issues, also complicates the detection of collisions between flows; and *iii)* by relying on topologies with full-bisection bandwidth, some network technologies do not implement congestion control mechanisms, such as the recent Bull BXI [13]. Finally, some proposed congestion control mechanisms are orthogonal to the fairness policy, such as RECN [14].

In this paper, we study fairness mechanisms for Interconnection Networks that are independent of congestion control. Overall, there are two approaches to implement global fairness: modify switch arbitration or throttle injection. The first approach deals with in-transit traffic, modifying the allocation mechanism and exploiting some type of global network information, but not modifying the traffic sources. Age-based arbitration [12], which has been employed in Cray XC series [3], is representative of this class of mechanisms and is often used as an ideal reference or network fairness. The second approach throttles injection at the sources, without requiring modifications of the network devices. The SAT distributed mechanism, originally proposed for LAN ring networks [11] and later extended to different topologies such as meshes, torus [19] and fat-trees [20], is a representative of this case. It implements perfectly balanced injection throttling by relying on a control signal (denoted SAT) which circulates among the network nodes.

This paper evaluates these two approaches in terms of (1) how they provide network fairness and (2) what is the impact on network fairness on parallel application performance using MPI traces from the NAS Parallel Benchmarks [8]). Both adaptive and oblivious routing algorithms are considered for each network scenario. In particular, the main contributions of this work are:

- Unfairness is evaluated for a range of topologies, routing mechanisms and traffic patterns.
- We identify the main limitations of Age arbitration and SAT. In particular, a realistic implementation of Age-arbitration using bounded queues fails to provide throughput fairness due to delayed assignment of timestamps when the network is in saturation. By contrast, SAT reduces average throughput when it refrains some nodes unnecessarily if their traffic does not collide with other congested flows.
- SAT and Age are thoroughly evaluated in under both synthetic loads and traces of real applications. Results confirm that the modified arbitration provides higher average throughput and the injection throttling policy is more fair. The impact on real applications is moderate, but performance improvements up to 14.2% are observed on specific applications, particularly when using fast computing nodes.

The rest of this paper is structured as follows: Section 2 introduces the SAT and Age mechanisms. Section 3 describes our evaluation methodology and simulation environment. Section 4 evaluates evaluates the tuning parameters, limitations and fairness levels of the two selected mechanisms. Section 5 measures the impact of both mechanisms on system performance. Section 6 reviews related work, and Section 7 concludes the paper.

2 FAIRNESS MECHANISMS

This section introduces the two fairness mechanisms evaluated for Interconnection Networks, SAT and Age, representative of the two approaches to implement network fairness in edge or network devices respectively.

2.1 The SATisfied Global Fairness Protocol

The SAT protocol balances the number of packets sent by each node by using a control signal, called SAT, that circulates among the network. A SAT interval T_{SAT} is defined as the (variable) time between consecutive receptions of the SAT signal on the same node.

The injection policy is implemented using two thresholds k and l, with $k \geq l$, which represent the maximum and minimum number of packets sent by active nodes during each interval T_{SAT}; these parameters could be also specified in bytes for different-sized packets. The policy is formally specified at the network interface by the following conditions:

- *Send Packet Condition:* A packet at the head of the injection queue only proceeds if its injection count is $< k$.

- *Forward SAT Condition:* A node forwards SAT if its injection buffer is empty or its injection count is $\geq l$. Otherwise, it holds SAT until one of these conditions holds.

Each NIC resets its injection count when it forwards SAT, which can be seen as a credit signal that allows the NIC to inject up to k packets. The SAT mechanism works because starved nodes retain the SAT signal, increasing the interval time T_{SAT} and making other nodes reach their k packet limit; as other nodes stop injecting, the starved nodes eventually have their chance to inject l packets. The selection of values for l and k and their impact on performance is evaluated in section 4.1. Note that k and l only have an effect on nodes that have data on their injection buffer; nodes with low injection rates are likely to be idle at some point during the SAT interval, making them forward the signal.

2.2 Age-based arbitration

The Age-based arbitration mechanism (Age for short), employs an oldest-first arbitration policy in each network router, using the requesting packet's injection timestamp as the age indicator. This mechanism was initially proposed to provide latency fairness at medium loads. However, at saturated loads it also eliminates the parking-lot effect of a round-robin arbiter, as older packets have higher priority regardless of their distance to the current router.

In order to assign the injection timestamp, a global clock coherent along the network is assumed. While this is unrealistic, there are approaches to approximate the ideal performance by using the transit latency of each packet and coarser time granularity [3]. Such realistic approaches won't be as fair as the ideal mechanism, this should be taken into account when considering the comparison of Age with SAT. Besides, section 4.2 identifies a further limitation to Age derived from small injection buffer sizes.

3 SIMULATION INFRASTRUCTURE

This section describes the simulation environment and the network configurations used to evaluate fairness. We employed FSIN [33], a network simulator which models networks based on virtual cut-through (VCT) flow control. The key simulation parameters and network loads are shown in table 1.

Network topologies. Mesh and torus (*k-ary n-cube*) have been traditionally used in parallel system networks and they are also common in NoCs, since they allow for regular, tiled architectures, using a simple layout and routing. Multidimensional meshes and torus have been widely used in system-level HPC interconnection networks, such as [4, 6, 10]. In these multi-dimensional networks paths are typically longer, which can increase the parking-lot effect presented in Figure 1. However, the application of SAT or Age to

Table 1: Simulation parameters.

	Parameter	Value
Configuration	Topology	Mesh, torus, fat-tree
	Number of nodes N	64
	Virtual channels	1 (fat-tree) / 2 (mesh, torus)
	Packet size	16 phit
	Injection queues size	24 packets
	Transit queue size	4 packets (64 phits)
	Traffic Pattern	**Description**
Synthetic	Random uniform	Equal probability to send to any node
	Permutation	Node sends to its pair (transpose, perfect shuffle)
	Hot-region	75% uniform, 25% packets sent to $\{0, N/8-1\}$
	Hot-spot	X % uniform, (1-X)% send to node 0
NAS	Pseudo applications	BT, SP, LU
	kernels	IS, CG, MG,FT

these topologies is independent of the number of dimensions, so for simplicity we focus only on 2D systems.

We include an indirect topology, the fat-tree (or folded Clos), implemented using multiple root nodes with constant-degree switches. The fat-tree is built using a butterfly connection pattern between each two contiguous levels. This topology is widely used at system level.

Router model. We employ an input-buffered router model with several virtual channels per input port and a two-phase allocator. For meshes and torus networks, the router employs a 5×5 switch and input queues with capacity for 64 phits.

The fat-tree implementation is modelled as a *4-ary n-tree*, with n the number of stages of switches and $k = 4$ the number of links going upward (or downward) from the switch. Our configuration has no virtual channels. The network can use oblivious or adaptive routing algorithm.

The routing function is either oblivious or adaptive. Adaptive routing in meshes avoids deadlock using 2 VCs. For the torus network, the routers use bubble flow control and the routing is either dimensional-order-routing (DOR) in both channels or adaptive in the second channel as in the adaptive bubble router [36]. In the fat-tree, the oblivious implementation selects the upward path depending on the source node, whereas downward path always depends on the destination. Adaptive routing selects the output port with more available link-level flow control credits [24].

Network workloads. We employ synthetic traffic or traces of applications to feed our simulator. For synthetic traffic, each node is modeled as an independent traffic source, following a Bernoulli distribution with a parameter that depends on the applied load. We have chosen a range of well-known synthetic traffic patterns [12], including two patterns that generate endpoint congestion, *hot-region* and *hot-spot*, to emulate situations such as congestion in the access to storage nodes. The simulator runs for a warm-up period of 50,000 cycles, followed with a stationary period in which 5 *batches* are measured to verify that results are stable over time and have statistical significance; each batch has $10 * N^2$ delivered packets, for N nodes in the network. Per-node throughput and the different fairness metrics are obtained from this phase. We consider min-to-average or individual node throughput to quantify fairness.

For real application loads we employ MPI traces of the NAS Parallel Benchmark programs [8], and measure the execution time of the region of interest, which is the parallel section of each application. We simulate all the pseudo applications (BT, SP, LU) and kernels (IS, CG, MG, FT) from the NPB suite except for EP, which is embarrassingly parallel so the network has a negligible impact. We use the Extrae MPI tracing tool [2] to obtain communication traces from the NAS Parallel Benchmarks (NPB) 3.2 [8], using problem sizes A. The traffic of these applications has been characterized in other works, such as [25]. The FSIN simulator replays MPI traces, preserving casual dependencies and taking into account the length of computation phases between messages [31]. We implement collective communications as a series of unicast messages (e.g., broadcast) or a series of pairs of messages interchanged among different pairs of nodes (e.g. all-to-all), which are common implementations in networks without multicast support. Additionally, the simulated processor frequency can be modified; a faster frequency shortens computation phases, introducing more pressure in the interconnect.

4 TUNNING OF FAIRNESS MECHANISMS

This section considers the required tuning of each mechanism and provides an initial evaluation of their characteristics and limitations.

4.1 Parameter configuration in SAT

SAT employs two parameters, l and k, which represent the minimum and maximum number of packets sent during a SAT interval T_{SAT}. Minimum values need to be assigned to l and k to prevent degradation of the network average throughput. The values of these parameters depend on the packet length L (in phits), T_{SAT} and the average accepted load A of the network without using SAT.

The parameter l needs to be larger than the SAT interval (measured in packets) in a case without congestion, to prevent unnecessary injection restrictions. In a direct network the hamiltonian path length H equals the network size N. If the injection buffer is empty, the SAT signal is immediately forwarded to the next node; otherwise, an additional cycle is required to check the injection count and decide to hold/propagate the signal. Therefore, in a non-saturated network of N nodes, each node receives a SAT at least every $H + N$ cycles, simplifying to single-cycle link latency.

For an indirect *4-ary n-tree* network, the length of its Hamiltonian path can be calculated using the following recursive relation: $H(0) = 0; \qquad H(n) = 8 + 4 \times H(n-1)$

Therefore, to reach an accepted load A, we require to set the SAT parameters as follows:

$$k \geq l \geq A \times (H + N)/L$$

Since the expected accepted load A for an application might not be known in advance, we assume its maximum value $A = 1$ phit/node/cycle. Using L = 16 phits, the minimum value l is 8 for a 64-node direct network and 16 for the 64-node fat-tree. As many global traffic patterns achieve peak loads in the 0.15-0.25% range, setting l to 8 is a quite conservative approach for most loads.

When a SAT signal arrives to a node with its counter equal to k, it resets the counter and allows the node to inject again. In saturation, the injection queue is full, so this will result in a burst injection of up to k packets, followed by a non-injecting phase until the next SAT signal arrives. The time between bursts allows for transit packets

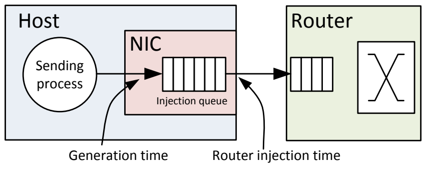


Figure 2: Timestamp assignment policies for Age: packet generation time and router injection time.

to advance in their paths and be delivered to their destinations. The larger l, the longer it takes a node to be *satisfied* by sending l packets and propagate the SAT, so the burst lasts longer and is followed by a longer wait. Explicit rate control (or *traffic pacing*) mechanisms such as [7, 9, 23] are useful to mitigate the problems of traffic bursts. However, they make the network slower to respond to changes in congestion [5] and in general it is better to adjust the parameters to the network conditions. Thus, when A is known we should adjust l to its lowest value.

The parameter k determines the level of fairness. We consider $k = l$ and $k = 2l$; the first option forces all nodes to inject at the same rate once saturation is reached, which results in the best minimum throughput. The second option reduces fairness but increases peak average throughput.

4.2 Age arbitration with realistic NIC buffers

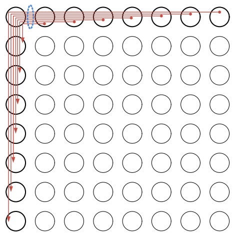
There are two alternatives for the instant of timestamp assignment in Age, as illustrated in Figure 2:

(1) *Router injection time:* From the point of view of the network, a packet is injected when it moves from the queue of the network interface card (NIC) of the computing node into the router's injection port.
(2) *Generation time:* from the point of view of the thread, a packet is injected when it is first queued at the computing node's NIC. This implementation accounts for the time a packet waits at the NIC, which could be significant at congested levels.

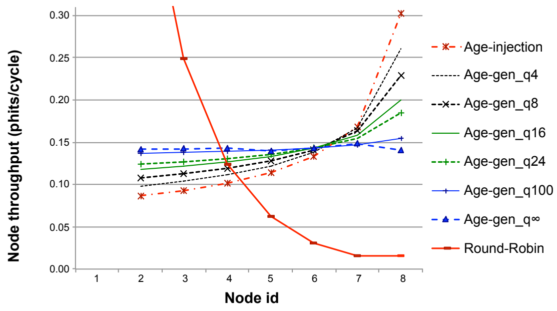
Lee *et al* [27] used the *Booksim* simulator [22] to show that Age arbitration avoids throughput imbalance between traffic flows and improves network stability in a mesh with XY (DOR) routing. Booksim employs an individual injection clock per node that runs behind the simulated clock time. This artifact is used to emulate an infinite NIC injection queue that provides an ideal global ordering between all packets that are sent on the network. The fairness evaluation in this study uses an injection queue with finite capacity.

We modified the Booksim simulator to support three models for timestamp assignment: router injection time (*inj_time*), realistic packet generation time (gen_q_x, NIC injection time using a limited capacity of x packets in the NIC queue), and ideal generation time ($gen\_q\infty$, ideal NIC injection time with an unlimited capacity).

We have measured the impact of the previous parameters in node throughput using an 8×8 oblivious mesh under the transpose permutation pattern. This pathological pattern generates congestion among the nodes in the same row, as presented in Figure 3a. Figure 3b shows the node throughput achieved by the nodes of row 0, all of which communicate with nodes in column 0. The ideal implementation of Age-based arbitration ($Age - gen\_q\infty$) results in an even injection rate of 0.14 flits/cycle. When router injection time ($Age - injection$) is used, fairness decreases as nodes in less congested areas inject at a faster rate. The minimum throughput is then 0.087 flits/cycle for node 1, 38% less than the ideal $gen\_q\infty$. Using generation time with realistic NIC queues ($Age - gen\_qx$) provides intermediate results. With a short queue the packets' age becomes closer to the router injection time; conversely, larger NIC's queues store *older* packets, increasing fairness.



(a) Transpose traffic sent to row 0.



(b) Peak throughput at full load for nodes in row 0

Figure 3: Transpose traffic pattern: (a) under XY routing, all nodes in row 0 compete for the bandwidth of a single link and (b) node throughput for the nodes of that first row with Age and round-robin arbitration.

Table 2: Minimum node throughput (phits/cycle) obtained under Age with different timestamp selection mechanisms on a 8x8 oblivious mesh.

Traffic Pattern	Age arbitration Ideal ($gen - q_\infty$)	$gen - q_{24}$	$inj.time$
Uniform	0.4073	0.3631	0.3323
Perfect shuffle	0.2377	0.1892	0.1061
Bit reversal	0.1392	0.1235	0.0867
Transpose	0.1416	0.1237	0.0867

Table 2 shows the same trends for other traffic patterns. Using injection time instead of generation time reduces minimum throughput between 18% (random traffic) and 63% (perfect shuffle permutation). Assuming a queue with capacity for 24 packets (q24), the minimum throughput is reduced by 10-20% compared with the ideal case q_∞. In short, sustaining minimum throughput at high loads requires large NIC's queues.

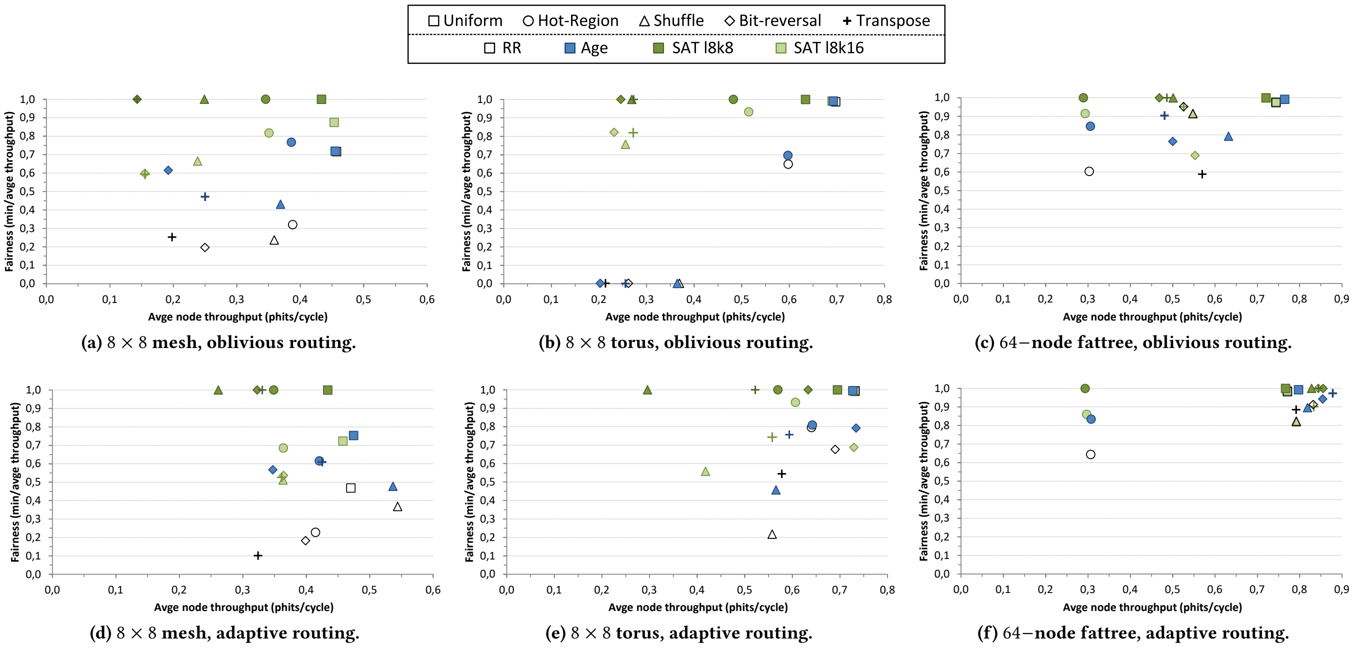


Figure 4: Fairness (min/average throughput ratio) versus average node throughput in a 64-node network under a range of traffic patterns.

5 IMPACT OF FAIRNESS ON SYSTEM PERFORMANCE

In this section we evaluate and compare the impact of injection throttling and age arbitration in system performance. We first identify the level of unfairness in different networks and routing mechanisms, identifying the mesh as the most unfair between the ones under evaluation. Next, we evaluate the impact of the fairness mechanisms at loads under saturation, under uniform or variable synthetic traffic. Finallyt, we present two cases that emulate realistic traffic: a situation with a single contended resource in the network, and simulation of MPI traces of parallel applications.

5.1 Base fairness evaluation

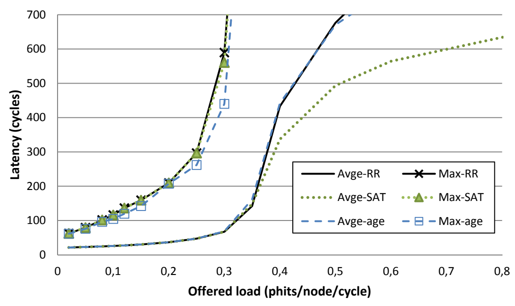
This section compares average throughput and fairness of the base case (round-robin arbitration, *RR*) and the SAT and Age mechanisms, at maximum load. We employ a finite injection queue with capacity for 24 packets and we select $l = 8$ for SAT, with both $k = l = 8$ and $k = 2 \cdot l = 16$. The ratio k/l limits the differences in throughput, but for most traffic patterns increasing k has only a minor effect in fairness.

Figure 4 compares fairness (measured as the min/average throughput ratio) versus average throughput under 5 network loads: *uniform*, *hotregion* and 3 permutation patterns (*shuffle*, *bit-reversal* and *transpose*). Fair results fall close to the top of the figure, and higher average throughput falls to the right.

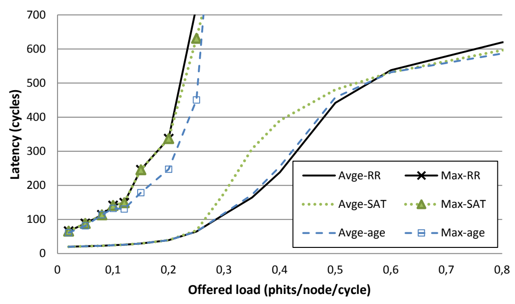
In the base *RR* case (in black and white), except for pathological cases commented later, the mesh topology suffers the largest unfairness under any traffic pattern, including uniform traffic. This is due to its lack of node symmetry, which causes overloads in its centre. The torus topology is regular, but under certain traffic patterns (*shuffle*, *bit-reversal* and *transpose*) with oblivious routing, the bubble-based deadlock avoidance mechanism generates pathological starvation issues [36]. By contrast, the fat-tree suffers the minimum unfairness. As expected, adaptive routing (Figures 4d-4f) increases average throughput and in most cases also minimum throughput, especially in the torus in Figure 4e where the previous pathological effect is mitigated by path diversity. However, even in absence of link failures, the three networks require a fairness mechanism to provide throughput guarantees at heavy loads.

Both SAT and Age improve fairness, in most cases at the cost of some reduction in average throughput. In general, Age provides intermediate improvements, except in the pathological cases of starvation in the oblivious torus which see no improvement. These cases are avoided using SAT, because it throttles injection and allows all nodes to send traffic, avoiding starvation. The fairness improvement obtained with SAT depends on the parameters selected; using $l = k$ provides maximum fairness and all markers sit in the top line. The configuration with $k = 2l$ reports lower fairness but, in most cases, a slight improvement in average throughput.

Average throughput, considering the average of the five patterns, decreases with the introduction of a fairness mechanism. In the mesh and torus, which suffer the highes unfairness in the base RR case, the reduction from using SAT ranges 4.8-20.4%, depending on the case and the SAT parameters. The penalty of Age is negligible under oblivious routing, and actually benefits (around 2%) under adaptive routing. In the fat-tree, throughput decreases when using



(a) Hot-region.



(b) Perfect-shuffle.

Figure 5: Maximum and average network latency in a 8×8 mesh network using different fairness mechanisms under *hot-region* and *Perfect-shuffle* traffic.

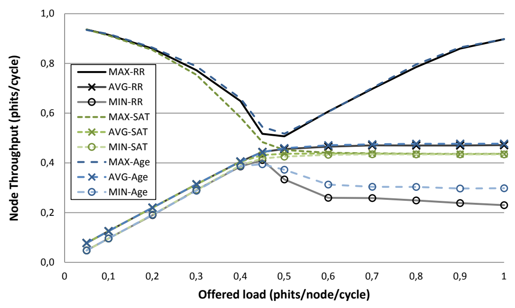


Figure 6: Node throughput in an 8×8 mesh with uniform traffic; Nodes 20 and 51 inject at full rate(1 phit/cycle).

oblivious routing but increases under adaptive routing. As an example, with the transpose permutation pattern in adaptive mesh or fat-tree, using any of the three fairness mechanism improves both fairness and average throughput.

In general, the results in this section show that any network topology presents unfairness and the considered mechanisms are effective to mitigate it. The penalty in average throughput is, in most cases, moderate. The following section focuses in the most unfair case of a mesh topology, evaluating the impact of both fairness approaches on different scenarios and application performance.

5.2 Impact of fairness mechanisms at loads below saturation

This section evaluates network latency at low/medium network loads below saturation to verify that the fairness mechanisms do not hinder performance. We run simulations with network load from 0.02 to 1 phit/cycle. Figure 5 presents result for *hot-region* and *perfect-shuffle*. For low/medium load, neither Age nor SAT hinder average latency. The maximum packet latency for each fairness policy is also shown; since network paths differ between flows, average and maximum latency do not need to be equal in a fair network. As expected, SAT shows the same values that the base case at low loads (all nodes are satisfied). As the initial goal of Age arbitration is to reduce the worst case latency, it is not surprising to observe that Age reduces maximum latency by 10-30%.

The average latency at saturation of Age and the base RR case is similar. In most traffic patterns, this latency is reduced with SAT, as throttling injection reduces network congestion. This is observed for *hot-region* in Figure 5a, and occurs for the other patterns which are not shown. By contrast, the *perfect-shuffle* pattern in Figure 5b exhibits a long transition phase until all nodes saturate (from load 0.25 to 0.6), during which SAT latency is higher than the base case.

5.3 Fairness under variable node injection rates

In previous evaluations all network sources inject packets at the same rate. However, realistic loads could have significant variations in node's injection rate [17]. In this subsection we consider two variable injection rate workloads.

Workload 1. this load considers a mesh in which all nodes inject traffic at the same given rate except two nodes (20 and 51) which always generate packets at full rate. Figure 6 shows minimum, average and maximum node throughput, with the variable injection rate for the other nodes indicated in the X axis. When the network load is low, nodes 20 and 51 can inject at high rates, as observed in the maximum throughput, using the spare bandwidth that the other sources do not need. For example, at 20% load these nodes are injecting at 84%; at low loads all policies exhibit the same behaviour, and neither SAT nor Age restrict throughput for those nodes.

Saturation occurs around 0.45 phits/cycle. At saturated loads, SAT is the most effective policy to maintain minimum node throughput (0.43 phits/cycle) and provide fairness, while both RR and Age provide slightly higher average throughput at the cost of fairness.

At saturated loads, nodes 20 and 51 have their best throughput with SAT, and suffer lower injection rates with round-robin (0.4 phits/cycle) and Age (0.35 phits/cycle). In the case of Age, unfairness comes from the finite injection buffering described in Section 4.2. SAT could provide a higher QoS level for those nodes, if needed, by increasing parameter l at their NIC; by contrast, RR or Age would require additional mechanisms to provide variable QoS levels.

Workload 2. one third of the nodes inject at their maximum rate, and the rest are set to a random value in the range (0, 1). Figure 7 shows the offered load per node, the node throughput in an 8×8 adaptive mesh under RR, and the impact of both fairness mechanisms in node throughput. Under both Round-Robin and Age arbitration, some nodes can inject at high rates, but this never occurs near the centre of the mesh which is saturated. With SAT,

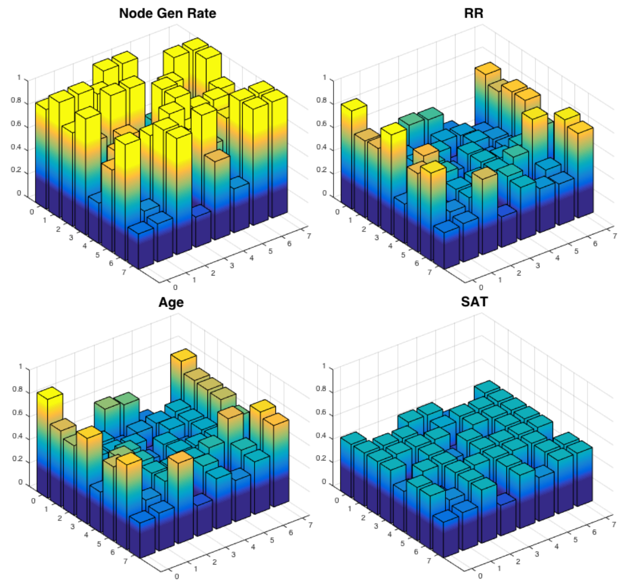


Figure 7: Node variable injection rates, and individual node throughput for round-robin (RR) base case, Age and SAT fairness mechanisms.

throughput is evenly capped at the rate of the first node that saturates, providing fairness but preventing outer nodes from using any spare bandwidth as Age does.

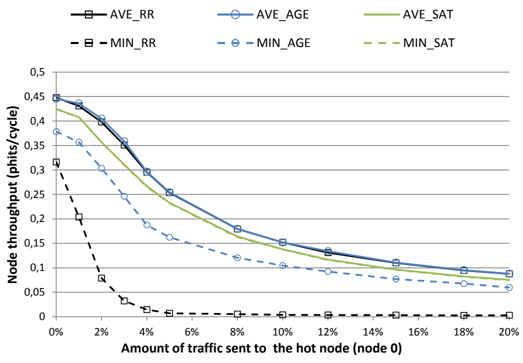
Fairness cannot be reported in this test by the lowest minimum node throughput, as some nodes present very low injection rates because their needs are low. When we consider *only* saturated nodes the minimum throughput of this set of nodes grows from 0.403 to 0.483 (20% more) when going from Age to SAT. By contrast, SAT also caps maximum node throughput from 0.913 to 0.487.

The two experiments in this section confirm the intuitions from previous sections: with variable injection rates, both fairness mechanisms have no significant impact on the nodes injecting below saturation. Both mechanism improve over the base RR case; Age obtains higher average throughput but fails to provide complete fairness, while SAT obtains more fairness and can be tuned with the parameter k, but restricts throughput of some nodes which could otherwise inject faster.

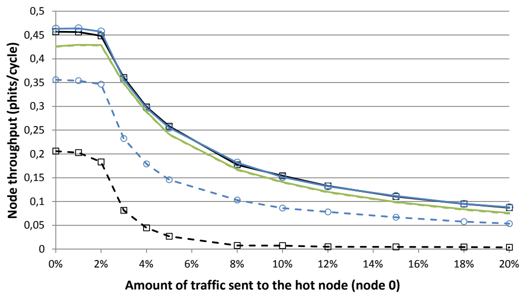
5.4 Fairness accessing a congested resource

Fairness in accessing a congested resource, for example to the I/O node linked to storage resources in a data-intensive grid application, is particularly relevant. This section analyzes the fairness on accessing such congested resource by considering a network that has a "hot" I/O node located at position 0. We model this scenario by making each node send a fixed percentage of its traffic load to node 0; the remaining traffic load follows a random uniform distribution.

Figure 8 shows the minimum and average node throughput at full load with 0% (only uniform traffic) to 20% traffic to the hot node. This traffic may correspond to requests from multiple applications running concurrently, from a single parallel application, or from a mixture of them. Average throughput drops quickly as the traffic



(a) Oblivious mesh.



(b) Adaptive mesh.

Figure 8: Average and minimum throughput as a function of the amount of traffic sent to the hot network resource.

sent to the hot node increases. When adaptive routing is used (Figure 8b), there is a significant improvement for the range 1-3%, because the pressure is now spread over the -X and -Y input channels. As observed in previous sections, Age improves minimum throughput compared to RR, while SAT consistently provides the best minimum node throughput.

Figure 9 depicts the bandwidth distribution per node (note the different scales for each alternative). As expected, RR presents the highest variability with nodes in row 0 getting the bulk of the I/O bandwidth under oblivious routing; Age improves access to the congested resource for nodes close to node 0 and SAT provides complete equality. With adaptive routing, Age becomes quite unfair because congestion sets on the centre of the mesh and only the nodes whose paths can avoid this centre get higher access to the I/O node. By contrast, SAT gets very close to the ideal network, allocating 1/64th of the bandwidth to each node (0.0156 phits/cycle/node).

5.5 Execution time of parallel applications

Finally, we evaluate the fairness policies under the NAS parallel benchmarks described in Section 3. We simulate an 8×8 mesh with each processing node running at the same frequency as the reference machine in which the trace was obtained. We also simulate the system with a frequency 4× faster and 4× slower to evaluate the impact of traffic pressure on both fairness mechanisms. We focus on adaptive routing with each of the three fairness models under study, using $l12k12$, $l12k18$ and $l12k24$ in SAT. The selection of

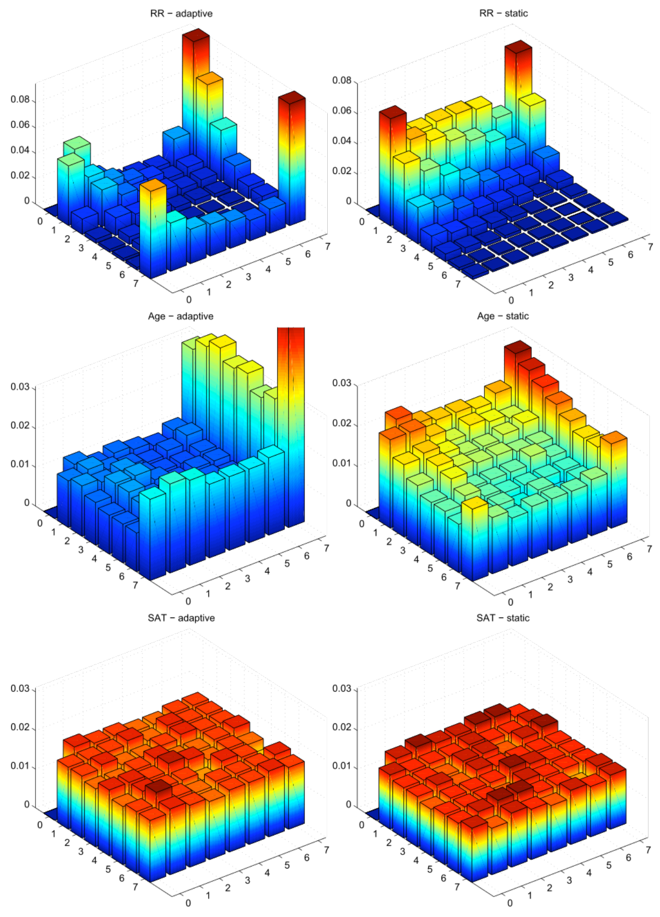
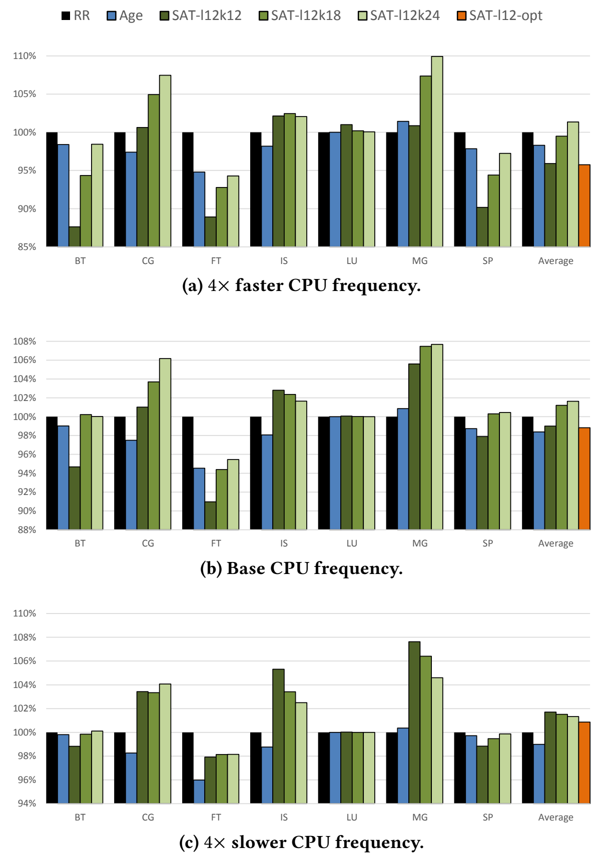


Figure 9: Map of bandwidth accepted (in phits/cycle) for the three network alternatives at 10% hotspot load. Left column is for adaptive mesh and right column for oblivious mesh.



(a) 4× faster CPU frequency.

(b) Base CPU frequency.

(c) 4× slower CPU frequency.

Figure 10: Normalized execution time of different NPB benchmarks, 8x8 adaptive mesh.

the parameter $l = 12$ in this case is discussed later in Section 5.5.2; omitted results with oblivious routing are similar.

We simulate a single parallel application at the time; fairness is applied to the traffic of the 64 processes in the same application, not between different applications. Fairness guarantees in Age and SAT provide a similar network service for all of them, which may contribute to reduce execution time. However, SAT restricts injection, which can be disadvantageous when the waiting process is in the critical path of the execution.

5.5.1 Performance. Figure 10 reflects execution time normalized to the base RR case. With the base frequency in Figure 10b Age improves performance from RR in almost all cases, except for MG. SAT with $l = k = 12$ (darkest green) improves the performance from the base RR cases in the pseudo-applications BT and SP. The result of LU is almost unmodified since it is the application with the lowest network load. In the kernels, by contrast, the results of SAT are mixed: while there is a large performance improvement in FT (speedup of 9.9%), the result in CG, IS and MG is negative. On average, Age and SAT with $l = k$ improve the execution time in 1.64% and 1.01% respectively from RR. The use of $k > l$ in SAT (lighter green bars) generally gets worse performance than $l = k$, except for IS. This suggests that the performance of this kernel depends more on average throughput (increases with higher k) than on network fairness (decreases with higher k). In fact, on average the two configurations $l12k18$ and $l12k24$ report worse performance than $l12k12$ and RR. Selecting the optimal SAT parameters for each application (*SAT-l2-opt*) is 1.19% faster than RR.

Figures 10a and 10c show results with different processing node frequency. The response of Age in both cases is similar to the original one. With 4× faster frequency, the network receives an average higher load, and the benefit of the selective injection restriction in SAT is higher. This is clear in cases such as BT, MG and SP. In particular, BT presents a speedup of 14.2%. Again, using $k > l$ is counter-productive. In the fast case in Figure 10a, on average SAT with $l = k$ improves a 4.26% and a 2.48% in overall execution time from RR and Age respectively. By contrast, the slow case in Figure 10c shows opposite results. When the network load is low, the injection restriction from SAT increases execution time on average. With $l = k$ the slowdown from RR is a 1.71. In this case, using $k > l$ is actually beneficial, since it reduces the level of injection restriction for some nodes. This is especially clear in the changes in BT, CG, IS and MG from the base case.

In conclusion, providing fairness between the different processes of a single application may benefit application performance, but it is highly dependent on each specific application, showing improvements up to 14.2% but also slowdowns. The injection restriction of SAT is particularly beneficial for cases of high traffic pressure, and in general using $l = k$ reports better results than $k > l$.

5.5.2 Sensitivity to SAT parameters. Previous sections evaluate the performance of SAT with an appropriate parameter l according to the minimums in Section 2.1. As discussed in Section 4.1, a value of l too large can increase the burstiness of traffic, decreasing performance. This section evaluates the impact of such bursty traffic on performance using parallel application traces.

Figure 11 shows normalized execution time varying l, with the base CPU frequency and $l = k$. Values $l < 8$ (or even larger in MG) show an increased execution time due to an excessive throttling of injection, consistent with the condition $l \geq 8$ for an 8×8 mesh in Section 2.1. The performance with l in $\{12, 24\}$ is relatively flat ($l = 12$ is used for the evaluation in Section 5.5), but from values around $l = 32$ the average execution time clearly increases with l. Since SAT provides maximum fairness for any $l = k$, performance changes are caused by the increased delays.

Overall, the in-depth evaluation in this section has shown that SAT is superior to Age in terms of absolute fairness, while Age provides better average throughput for most synthetic loads. Under realistic application loads, moderate improvements are observed with both mechanisms, with SAT being especially effective when the network load is relatively high.

6 RELATED WORK

In this paper we have explored the behavior of two basic *explicit* fairness mechanisms. SAT has been considered representative of mechanisms performing injection throttling. Similar approaches have been incorporated into more elaborated control systems in the datacenter, such as FastPath [34] or Explicit Priority Notification [28]. These are centralized implementation relying on a network controller, and have a flow-level granularity instead of per-node injection throttling; such approaches are similar depending on the mechanism being implemented in the OS or the NIC. Other recent proposals focus on providing fine-grain differentiation, such as qJUMP [18] or NUMFabric [30].

Similar injection control mechanisms have been applied in Networks-on-Chip. Walter *et al* [37] proposed a low-cost credit based distributed access regulation technique to provide fair access rights to hot-modules in the NoC such as DRAM controller. Injection throttling has also been proposed for buffer-less Network-on-chips [15, 32], not to achieve fairness but to reduce congestion and increase throughput at heavy loads. Finally, different systems employ injection throttling to provide fairness as part of their medium-access control (MAC) mechanism, such as [11].

Age-based arbitration has been proposed to achieve latency fairness and in doing so it indirectly provides throughput fairness as well, [3, 12]. Alternative arbitration policies such as weighted round-robin or a probabilistic distance-based arbitration [27] scheme have been shown to be effective under XY routing in NOCs. However, these mechanisms require significant changes to the arbitration mechanism including tuning multiple weights, and may result in lower fairness under saturation than the reference Age arbitration according to their own evaluations. Specific improvements exist in the context of networks-on-chip; for example, globally synchronized frames [26] guarantees minimum throughput based on coarse intervals denoted as *frames*, generalizing Age arbitration to support multiple levels of Quality-of-Service (QoS) based on the concept of deadline-aware arbitration. However, their implementation requires specific hardware to implement global barriers on a NoC.

7 CONCLUSIONS

This study has presented a comprehensive evaluation of network fairness in a range of interconnection networks. Firstly, we measured fairness at saturation for three popular topologies, mesh, torus and fat-tree, showing that variations in node throughput under non-uniform traffic are significant and cannot be ignored. Adaptive routing improves fairness but not completely. We expect *any* network design to show unfairness for uneven loads, although some topologies are intrinsically more fair that others. Therefore, the expectation of average peak throughput being indicative of the interconnection network performance at high loads is unfounded.

We explored two classes of explicit fairness mechanisms: Age, which modifies arbitration to give priority to older packet, and SAT, which throttles injection to maintain fairness. In particular, we evaluate their impact on node throughput, under a range of traffic loads in different topologies.

Secondly, we have verified that none of the two approaches limit performance at network loads below saturation, but we have identified other limitations. Realistic implementations of Age arbitration with finite buffers suffer from unfairness issues under saturation due to the delayed assignment of timestamps. SAT, by contrast, enforces a strict fairness level configurable by its parameters l and k, at the cost of reducing peak throughput of all nodes, even when they might not contend be involved in path contention. Thus, when network contention only involves a few nodes, Age might be more competitive since it does not penalize other nodes. By contrast, with widespread network contention, SAT is better at guaranteeing minimum node throughput and fairness.

Finally, this study has considered two realistic scenarios for an adaptive mesh. The baseline access to a single "hot" resource, such as a congested I/O node, presents significant throughput unfairness, with some individual nodes receiving less than 50% of the average per-node throughput. Age arbitration mitigates this issue, but it is still unfair because of the identified limitation with finite buffers, and injection throttling results in a completely fair access. When considering traces of parallel applications, the impact is modest because the coupled behaviour of the application processes avoids starvation issues. Still, we have observed improvements larger than 10% in several applications, particularly with fast computing nodes. On average, injection throttling results more efficient than Age arbitration in such cases, but it is counter-productive when the network pressure is low. In all situations, Age arbitration slightly outperforms a baseline round-robin system.

Since SAT restricts maximum throughput of all nodes, regardless of being involved in path contention or not, future research might focus on automated mechanisms that selectively enable injection

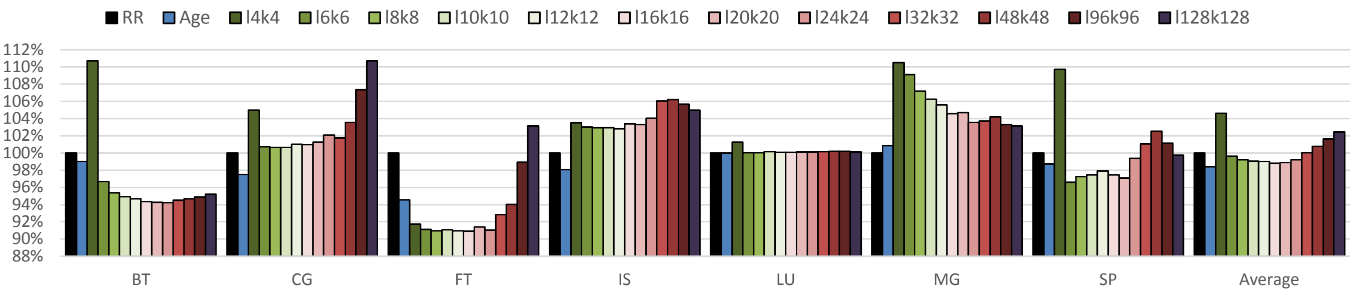


Figure 11: Sensitivity of execution time to the SAT parameter l. Normalized execution time of NPB codes on an 8×8 mesh using adaptive routing and $l = k$.

restriction only on the nodes involved in network congestion. Additionally, further work is required to monitor system performance in order to detect when unfairness is being detrimental to applications.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Science, Innovation and Universities under contract TIN2016-76635-C2-2-R (AEI/FEDER, UE).

REFERENCES

[1] 2010. IEEE Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks - Amendment: 10: Congestion Notification, 802.1Qau.
[2] 2015. Extrae MPI profiling tool. http://www.bsc.es/ssl/apps/performanceTools/
[3] Dennis Abts and Deborah Weisser. 2007. Age-based packet arbitration in large-radix k-ary n-cubes. In *ACM/IEEE Conference on Supercomputing (SC '07)*. ACM, New York, NY, USA, Article 5, 11 pages.
[4] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. 2005. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development* 49, 2.3 (march 2005), 265 –276.
[5] Amit Aggarwal, Stefan Savage, and Thomas Anderson. 2000. Understanding the performance of TCP pacing. In *INFOCOM*, Vol. 3. 1157–1165.
[6] Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu. 2009. Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers. *Computer* 42 (2009), 36–40.
[7] Mohit Aron and Peter Druschel. 1998. *TCP: Improving Start-up Dynamics by Adaptive Timers and Congestion Control.* Technical Report. Rice Univ.
[8] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. 1991. The NAS parallel benchmarks, summary and preliminary results. In *ACM/IEEE Conference on Supercomputing*. 158–165.
[9] F. Bonomi and K.W. Fendick. 1995. The rate-based flow control framework for the available bit rate ATM service. *Network, IEEE* 9, 2 (Mar 1995), 25–39.
[10] Dong Chen, Noel A. Eisley, Philip Heidelberger, Robert M. Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L. Satterfield, Burkhard Steinmacher-Burow, and Jeffrey J. Parker. 2011. The IBM Blue Gene/Q interconnection network and message unit. In *Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*. Article 26, 10 pages.
[11] I. Cidon and Y. Ofek. 1993. MetaRing-a full-duplex ring with fairness and spatial reuse. *IEEE Transactions on Communications* 41, 1 (jan 1993), 110–120.
[12] William Dally and Brian Towles. 2003. *Principles and Practices of Interconnection Networks.* Morgan Kaufmann, San Francisco, CA, USA.
[13] S. Derradji, T. Palfer-Sollier, J.-P. Panziera, A. Poudes, and F.W. Atos. 2015. The BXI Interconnect Architecture. In *Symposium on High-Performance Interconnects.*
[14] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo. 2005. A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks. In *Intl. Symp. on High-Performance Computer Architecture (HPCA)*. 108–119.
[15] Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt. 2010. Fairness via Source Throttling: A Configurable and High-performance Fairness Substrate for Multi-core Memory Systems. In *ASPLOS*. 335–346.
[16] Ernst Gunnar Gran and Sven-Arne Reinemo. 2011. InfiniBand Congestion Control: Modelling and Validation. In *ICST Conference on Simulation Tools and Techniques (SIMUTools '11)*. 390–397.
[17] Paul Gratz and Stephen W. Keclker. 2010. Realistic Workload Characterization and Analysis for Networks-on-Chip Design. In *Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI).*
[18] Matthew P. Grosvenor, Malte Schwarzkopf, Ionel Gog, Robert N. M. Watson, Andrew W. Moore, Steven Hand, and Jon Crowcroft. 2015. Queues Don'T Matter when You Can JUMP Them!. In *Networked Systems Design and Implementation.*
[19] C. Izu. 2009. A throughput fairness injection protocol for mesh and torus networks. In *Intl. Conf. on High Performance Computing (HiPC).*
[20] Cruz Izu and Enrique Vallejo. 2012. Throughput Fairness in Indirect Interconnection Networks. In *Intl Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. 233–238.
[21] V. Jacobson. 1988. Congestion Avoidance and Control. *SIGCOMM Comput. Commun. Rev.* 18, 4 (1988), 314–329.
[22] Nan Jiang, James Balfour, Daniel U Becker, Brian Towles, William J Dally, George Michelogiannakis, and John Kim. 2013. A detailed and flexible cycle-accurate Network-on-Chip simulator. In *Intl. Symp. on Performance Analysis of Systems and Software (ISPASS).*
[23] Srinivasan Keshav. 1991. A Control-theoretic Approach to Flow Control. In *SIGCOMM*. 3–15.
[24] John Kim, William J. Dally, J. Dally, and Dennis Abts. 2006. Adaptive Routing in High-Radix Clos Network. In *Supercomputing Conference.*
[25] J.S. Kim and D. Lilja. 1998. Characterization of communication patterns in message-passing parallel scientific application programs. *Network-Based Parallel Computing Communication, Architecture, and Applications* (1998), 202–216.
[26] Jae W Lee, Man Cheuk Ng, and Krste Asanovic. 2008. Globally-synchronized frames for guaranteed quality-of-service in on-chip networks. In *International Symposium on Computer Architecture.*
[27] M.M. Lee, J. Kim, D. Abts, M. Marty, and J.W. Lee. 2010. Probabilistic Distance-Based Arbitration: Providing Equality of Service for Many-Core CMPs. In *Intl. Symposium on Microarchitecture.*
[28] Yuanwei Lu, Guo Chen, Larry Luo, Kun Tan, Yongqiang Xiong, Xiaoliang Wang, and Enhong Chen. 2017. One More Queue is Enough: Minimizing Flow Completion Time with Explicit Priority Notification. In *INFOCOM.*
[29] J. Mo and J. Walrand. 2000. Fair end-to-end window-based congestion control. *Transactions on Networking* 8, 5 (2000), 556–567.
[30] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. 2016. NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters. In *SIGCOMM Conference*. 188–201.
[31] Javier Navaridas, José Miguel-Alonso, Jose Antonio Pascual, and Francisco Javier Ridruejo. 2011. Simulating and evaluating interconnection networks with INSEE. *Simulation Modelling Practice and Theory* 19, 1 (2011), 494–515.
[32] George P. Nychis, Chris Fallin, Thomas Moscibroda, Onur Mutlu, and Srinivasan Seshan. 2012. On-chip Networks from a Networking Perspective: Congestion and Scalability in Many-core Interconnects. In *SIGCOMM*. 407–418.
[33] Francisco Javier Ridruejo Perez and José Miguel-Alonso. 2005. INSEE: An Interconnection Network Simulation and Evaluation Environment. In *Euro-Par Conference.*
[34] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: A Centralized "Zero-Queue" Datacenter Network. In *ACM SIGCOMM 2014*. Chicago, IL.
[35] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato. 2005. Solving Hot Spot Contention Using InfiniBand Architecture Congestion Control. In *High Performance Interconnects for Distributed Computing.*
[36] V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo, and J.M. Prellezo. 2001. The Adaptive Bubble Router. *J. Parallel and Distrib. Comput.* 61, 9 (2001), 1180 – 1208.
[37] I. Walter, I. Cidon, R. Ginosar, and A. Kolodny. 2007. Access Regulation to Hot-Modules in Wormhole NoCs. In *Intl. Symp. Networks-on-Chip.*