Throughput Fairness in Indirect Interconnection Networks

Cruz Izu
School of Computer Science
The University of Adelaide
Adelaide SA 5005 Australia
Email: cruz@cs.adelaide.edu.au

Enrique Vallejo
University of Cantabria
Avda Los Castros s/n
39005 Cantabria, Spain
Email: enrique.vallejo@unican.es

***Abstract*—The performance of an interconnection network is typically measured by two metrics: average latency and peak network throughput. Average network throughput is usually reported in the belief the network is fair and all source nodes are supposedly able to inject at the same rate. However, most systems exhibit significant network unfairness under non-uniform loads. At high loads, if link utilization is uneven, the injection matrix will also become uneven. This unfairness significantly degrades the performance of some nodes, and eventually the whole system.**

Fairness issues have been previously reported for direct topologies such as mesh and torus, but this work evaluates throughput fairness in indirect networks, specifically the fat-tree topology. We will see fairness is still an issue for indirect networks in the presence of hot-spots. The SAT protocol was initially proposed to provide throughput fairness for ring networks. This paper extends the original protocol to implement a fairness injection mechanism that works for indirect networks. A thorough evaluation will show that for most scenarios it is possible to achieve throughput fairness without a significant lost of peak throughput.

I. Introduction

The communication subsystem is a key component of a parallel computer, which can range from tens of cores in a multi-core microprocessor to hundreds or thousands of nodes in a Massively Parallel Processor(MPP). Interconnection networks (IN) such as the mesh, torus or fat-tree have been widely deployed in parallel computers [1], [2] and are also proposed for NoC systems [3].

There is a large body of research on network topologies and routing strategies, deadlock and livelock [4]. Deadlock is also a critical issue which can be prevented or detected and recovered from. Livelock can be avoided by using minimal paths or prevented by limiting the number of misrouting steps a packet can take. By contrast, there is limited work on the problems of unfairness and starvation. Starvation is the worst-case scenario of network unfairness in which a particular computation node remains unable to access the network resources for an unbounded time limit. Starvation is generally avoided by having a fair arbitration scheme that guarantees a bounded waiting time for any packet requesting an output channel. Less severe cases of network unfairness will allow network nodes to inject packets at different rates, resulting in some of the nodes experiencing congestion while others are still able to inject at their full rates.

In order to evaluate fairness, we must measure throughput as seen by each node, i.e. by the number of packets that can be sent by that node per unit of time. Fairness is achieved when the minimum and maximum node throughput are in close proximity; only in that case, average throughput is a valid metric for network performance.

The SAT protocol [5] was initially designed to provide throughput fairness for local-area networks with ring topology. A variation of this protocol has been shown to be effective in a mesh or torus network of small to medium size to eliminate throughput unfairness [6]. This work extends that evaluation to the fat-tree network under both static and adaptive routing.

This paper is structured as follows: Section II introduces fairness, the SAT protocol and its extension to indirect networks. Section III introduces the evaluation infrastructure and how to measure and report peak throughput at high loads, and Section IV presents an evaluation of network fairness with and without the SAT mechanism for indirect networks. Finally, Section V summarizes the findings of the work.

II. Network unfairness and the SAT Protocol

We will start this section describing the motivation for fairness issues. A locally-fair arbiter (such as round-robin) allocates the bandwidth evenly amongst its incoming links so if two network flows request the same output link, each gets 50%. However, as a packet travels through the network, it merges with both newly injected packets and packets coming from other directions. This merging causes packets that travel longer and traverse busy areas to receive less bandwidth that those travelling shorter paths, generating throughput unfairness issues. Unfairness in direct networks (mesh, torus) has been studied in detail in [6].

The fat-tree (or folded Clos) is an indirect topology, since it employs transit routers not directly connected to processing nodes. A tree splits transit routers into several levels and a router at a given level is connected to routers in the next (if any) and previous levels without creating loops. The computing nodes are connected to level 1; the highest level is the root of the tree. Trees typically provide a limited bisection bandwidth since the root becomes a bottleneck; fat-trees employ links with more capacity in the higher levels of the topology towards the root. This can be accomplished in

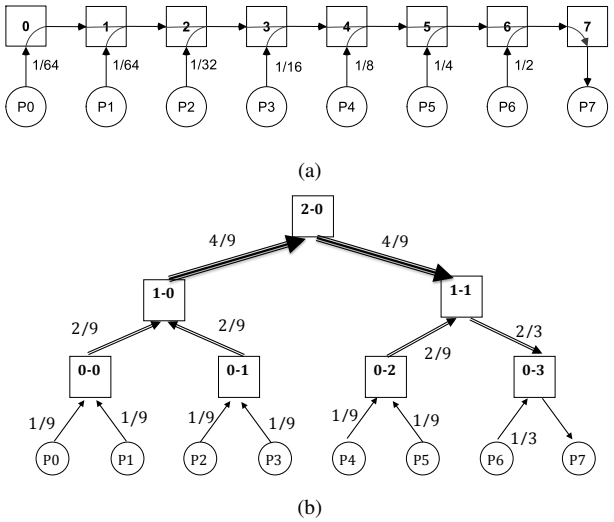

Fig. 1. All nodes are sending to P7 and merging traffic at each hop for (a) a 8-node 1D mesh and (b) a 8-node fat-tree network

two ways: using faster links in the higher levels (as depicted in Figure 1(b), typically implemented using link trunking), or using multiple root nodes with constant-degree routers (a folded Clos, as depicted in Figure 2). This multi-rooted topology is mathematically denoted as *k-ary n-tree*, and provides maximum Bisection Bandwidth for any size of the network. This topology has been largely used at system level, and has been also explored for NoC designs [7].

Figure 1 depicts an example of unfairness for two different 8-node networks, in which nodes 0 to 6 are all sending packets to node 7. The topology is either a 8-ary 1-cube network or a 2-ary 3-tree. In an ideal network in which all nodes have similar communication demands, all nodes should experience the same throughput. The differences for the array are significant, due to the *parking lot* effect. In fact, meshes are known to be unfair even under uniform traffic [8]. By contrast, the fat-tree has shorter paths and increased bandwidth, so this effect is reduced but not eliminated. As before, node 6 gets most of the available bandwidth, due to being one hop away (a round robin arbiter would allocate $1/3$ of the bandwidth towards node 7 to its neighbour 6), although any other source gets the same fraction of the network bandwidth towards node 7, $1/9$. Note that the specific result depends on the number of physical links entering a given router, so an alternative design (using oversubscription or faster links instead of aggregation) would result in different node throughput, but it would still be globally unfair. In short, a fair local arbiter such as round robin is not a guarantee for node throughput fairness.

As unfairness is a frequent issue after saturation, we would like to find a mechanism to provide network fairness at high loads. Next section describes the SAT mechanism, which is based on the fairness protocol of the Metaring network [5], with some simplifications due to the fact we assume a lossless network (with no loss or duplication of signals). We will also explain how to adapt this protocol to an indirect network.

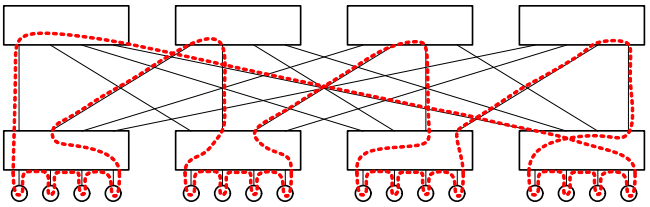

Fig. 2. Hamiltonian path for SAT propagation in a $k-ary\ n-tree$ network

A. The SATisfied Global Fairness Protocol

The SAT protocol balances the number of packets sent by each node. This is guaranteed by a control signal called SAT that circulates across the network. An interval is defined as the time between consecutive receptions of the SAT signal on the same node. Each node counts the number of packets sent during each interval. To provide fairness, nodes that send too many packets will stop injecting until they receive the SAT signal again, and nodes that could not inject enough packets will retain the SAT signal.

This is implemented using two thresholds k and l, with $k \geq l$. On a given interval a node cannot inject more than k packets; if the limit is reached, the node has to wait for the SAT signal. By contrast, a node that has injected less than l packets during an interval (with more packets ready) is said to be *starved*, or not SATisfied. *Starved* nodes will retain the SAT signal until they become SATisfied or have no more packets to send. In other words, an starved node will force satisfied nodes to slow down injection until he is also satisfied.

The implementation of this global fairness protocol can be done with minimal hardware support. The injection interface will have an injection count register, incremented each time a packet is injected and reset on SAT departure. Additionally, it will require either sending a small control packet or adding a 1 bit control line to the routers to propagate the SAT signal from one node to the next.

B. Applying the SAT protocol to a k-ary n-tree network.

Previous work [6] have shown how to extend the SAT protocol to other network configurations by using a Hamiltonian circuit as a virtual ring to propagate the SAT signal from node to node. Figure 2 shows the virtual ring chosen for the fat-tree network, which visits nodes in ascending order, going from node i to node $(i+1) mod\ N$. The length of this Hamiltonian length is longer than N, since there are hops between nodes and intermediate transit switches.

For a $k-ary\ n-tree$ network, the Hamiltonian path length can be calculated using the following recursive relation:

$$H(0) = 0$$

$$H(n) = 2 \times k + k \times H(n-1)$$

a) Parameter tuning: The parameters k and l determine the fairness level and the performance of the system. The parameter l must be set to achieve the maximum throughput considering the SAT interval: If l is too small, nodes will have

to wait for the SAT signal and throughput will decrease; by contrast, if l is too large, starved nodes will wait longer for the SAT signal, increasing average latency and generating bursts of traffic when SAT is received.

In absence of network contention, the length of the SAT interval will be $I = H(n) + N$ cycles, being N the number of processing nodes and assuming single-cycle latencies for the links and single-cycle node latency. To reach a full load we need to set $l > (H(n) + N)/L$. For a 64 node example network built as a 4-ary 3-tree, the value H(3) is 168, the SAT interval is 232 cycles and with a packet length of $L = 16$ phits we need to set $l > 14$.

Using $k = l$ makes the differences between minimum and maximum node throughput less than 1% [6], while using $k > l$ increases average throughput at the cost of reducing fairness.

III. Network simulation

This Section describes the network configurations studied and the simulation environment used to measure fairness.

A. Network and Router model

Performance and fairness is evaluated using FSIN [9], a network simulator which models virtual cut-through (VCT) networks. FSIN can be parameterized, specifying the topology of the network (direct or indirect), number of dimensions, the traffic workload and the architecture of the router: number of virtual channels (VCs), buffer sizes, routing function, crossbar arbitration, etc.

The fat-tree implementation is a k-ary n-tree [10], whereby k is half the radix of the routers; actually, the number of links going upwards (or downwards) from the router. A butterfly connection pattern is employed between contiguous levels. In order to keep things simple, we do not employ virtual channels. Packet size is set to 16-phits and each input queue can store 4 packets. The network can use static or adaptive routing. The adaptive algorithm uses shortest paths and a credit-based flow-control mechanism, so when several output ports are possible, the one with more available credits is selected. The arbiter is round-robin in both cases.

B. Network workload

Since we are interested in performance at saturated loads, each node is modeled as an independent traffic source with an applied load fixed to 1 phit/cycle/node. Network throughput is measured at messages delivered per cycle and it is usually normalized, dividing it by the message size (in phits) and the network size. Thus throughput is reported as phits/node/cycle. We have chosen a range of well-known traffic patterns: random uniform traffic and permutations such as transpose, butterfly and bit reversal [4]. We also considered a hot-region pattern in which 25% of network packets have destinations in the range (0, N/8 -1); the other 75% traffic is uniformly distributed.

The simulator runs for a warm-up period of 50,000 cycles, followed plus a stationary period in which 5 batches (each batch has $10 * N^2$ delivered packets) are measured to verify that results are stable over time. The output of the network simulation produces an injection matrix with the number of packets injected by each node during the simulation time. From this matrix we can calculate the throughput experienced by each node with/without the SAT protocol. Fairness is achieved when the minimum node throughput is not far from the network average throughout.

We should note that the SAT protocol will not introduce penalties at low and medium loads. In such cases, most of the injection buffers would be empty so the SAT signal will travel fast, and provided that the parameter *l* is not too small, nodes will never have to wait for the signal to inject a new packet. Thus no results will be presented for loads below saturation.

IV. Network Evaluation

This section presents the evaluation using a network of 64 nodes (4-ary 3-tree), to be able to examine node throughput in detail. We first quantify the unfairness level when no fairness mechanism is employed. Secondly, we show the impact that the SAT protocol has on fairness under the range of traffic patterns, assuming $l = k$. Then, we discuss the impact of using $k > l$, and look at the node throughput for three of the traffic patterns in more detail. Finally, the last subsection extends the evaluation to larger networks, 256 and 1024 nodes, to see how the protocol scales as the network grows.

A. Measuring unfairness in the k-ary n-tree

We have measured the minimum, maximum and average node throughput for the two network topologies with a full applied load of 1 phit/node/cycle. The main objective of this evaluation is to check whether the average peak throughput of the network is representative of individual node throughput; in other words, to see if the network is fair.

Figure 3 shows the throughput values for the 4-ary 3-tree network under the different traffic patterns. The network achieves fairness under both uniform and butterfly traffic patterns. Peak node throughput of 0.95 phits/cycle is achieved for the butterfly permutation as its communication topology matches the interconnection pattern of the network, with all nodes able to inject at nearly full rate. However, the network is unfair for the other traffic patterns, particularly hot-region and perfect shuffle. Adaptive routing reduces unfairness but there are still significant differences between maximum and minimum node throughput for these patterns.

Compared with the mesh [6], the fat-tree network is more fair due to two factors: firstly, contention is lower as the bisection bandwidth grows with the network size; secondly, as congestion occurs farther from the source nodes that in direct topologies, its impact is less severe.

B. Using SAT to achieve Throughput Fairness

This section analyzes the impact that SAT protocol has on node throughput at full applied load. We initially set $k = l$, what imposes strict fairness (min and max throughput differ in less than 1%) so there is no point in showing all the results. Instead, we will compare average throughput, which can decrease at the cost of the increased fairness. We tested

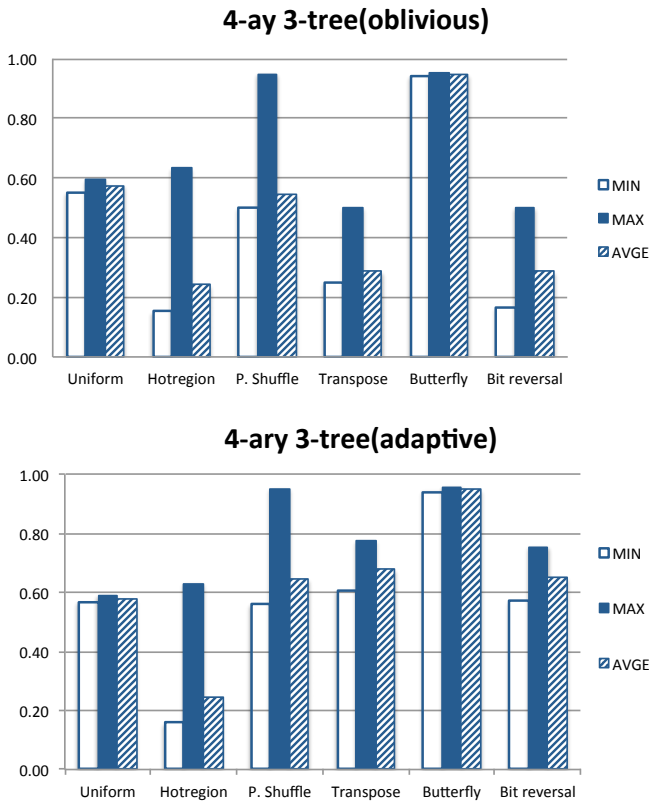

Fig. 3. Node throughput range (phits/cycle) for the indirect 64 Node network under a range of synthetic traffic patterns at full applied load.

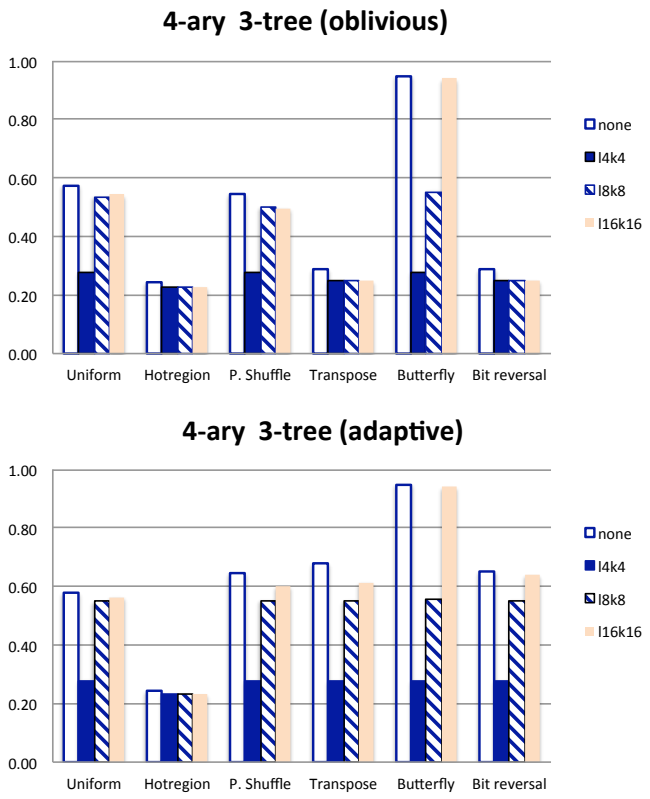

Fig. 4. Average node throughput for different values of $k = l$.

different values of l from 2 to 20. Provided l is large enough to cover for the SAT interval, there is little difference in network performance, so we only show the most relevant results.

Figure 4 compares the average node throughput for fat-tree with or without SAT. We can see that there are significant gains in minimum throughout when applying SAT for all the unbalanced workloads.

A small value of k may prevent nodes to inject at their peak rate. As discussed in Section II-B, a 64-node network get the best performance with $l \geq 16$, although some loads perform well with $l = 8$.

Restricting injection reduces message latency and buffer utilization, as observed in our simulations; the base case for uniform traffic under static routing has an average message delay of 551 cycles from packet injection, while $l8k8$ halves that delay to 226 cycles. As we increase the SAT parameters, buffer utilization and network delay grows to 297 cycles for $l12k12$ and 321 cycles for $l16k16$. Thus, we should use the lowest values of l and k that do not restrict network throughput.

Overall, SAT is effective to guarantee node throughput at high loads for indirect networks. In the rest of this section we will look into the performance and tunning of the SAT protocol in more detail.

C. Node throughput analysis under 3 traffic patterns

This section tested SAT with $k > l$, to explore the impact of k on fairness and looks at the node throughput distribution in more detail.

Figure 5.(a) shows the injection rate of each of the 64 nodes under uniform traffic; as the network usage is balanced, the distribution is reasonable flat, with minor variations due to the randomness of the load. In this case, increasing k have little impact on node throughput. This changes when the traffic load is unbalanced. The hot-region pattern, reflects the appearance of hot-spots and shows significant unfairness as shown in figure 5.(b). We can see that the nodes which belong or are close to the hot-region (nodes 0 to 7) are able to inject more packets overall (as explained by figure 1) while in the rest of the network are all injecting at the same rate of 0.16 phits/cycle. The SAT protocol with $l12k12$ increases the minimum rate up to 0.22 phits/cycle, which goes down to 0.19 phits/cycle with $l12k24$. Using a value $k > l$ reintroduces unfairness, but limits the differences between minimum and maximum throughput to the ratio $k : l$.

Figure 5.(c) shows the node throughput distribution for the adaptive network under the perfect shuffle permutation. As expected, SAT with $l12k12$ or $l16k16$ produces a flat line, with all nodes injecting 0.57 phits/cycle; $l12k16$ increases the injection rate for nodes in less busy areas and it starts to resemble the distribution of the base case. As the minimum throughput is above 0.5 flits/cycle, SAT with $l12k24$ shows the same node

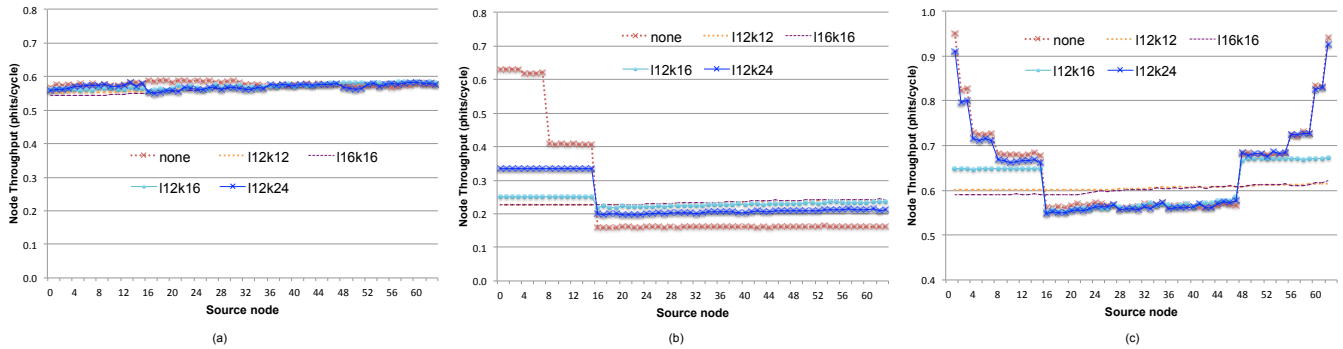

Fig. 5. Node throughput distribution (phits/cycle) for adaptive 4-ary k-3 network under (a) uniform, (b) hot-region and (c) perfect shuffle traffic patterns.

distribution that the base case. This representation shows best the trade-off between node throughput and performance. We can increase average node throughput by favoring nodes whose traffic uses the less congested areas but this will penalize to some degree (depending on the topology and workload) the other nodes in the network.

In short, each traffic load will have its own particular node distribution. Using the SAT protocol with an adequate parameter selection will provide fairness without significant loss of throughput.

D. Scalability of the SAT protocol

Previous sections have studied the performance of the SAT protocol in a 64 node network . In this Subsection we evaluate the performance on larger networks: 256 nodes (4-ary 4-tree) and 1024 nodes (4-ary 5-tree). We will focus only on adaptive routing as the difference in performance between oblivious and adaptive routing grows with network size. In addition, we know that SAT's parameters need to grow with peak performance which will be higher in the adaptive network.

We start with the case of networks with 256 nodes. Using the calculations made in Subsection II-B, the SAT propagation delay is 936 cycles. Thus, the minimum values to reach full load are $l > 58$. However, peak throughput for the base case ranged from 0.13 phits/cycle for the bit reversal permutation to 0.94 for the butterfly permutation. Thus we need to set l in the range 24-56. For a network of 1024 nodes, the SAT propagation delay will be 3752 cycles for the fat-tree. This translates into a required $l \geq 234$ for the fat-tree, considering full load. The fat tree was measured with *l* from 64 up to 256.

Table I shows a selected summary of the results for the fat-tree topology, using different parameters *l* and *k*. Although we have evaluated all the previous traffic patterns, we have selected three which represent the different observed behaviors.

The fat-tree scales well, due to is larger bisection bandwidth that grows linearly with network size. Network is fair for uniform traffic, but throughput unfairness slowly grows ith network size: for the hot region the minimum value is now 57% of its average value (compared with 65% in 64-node fat-tree), and for the transpose it is 77% of its average(compared with 89% for the 64-node fat-tree). SAT with a value $l \geq 32$ performs well for most patterns. The exception is the butterfly permutation (not presented in the Table) which still reaches 0.95 phits/cycle and it work best with either l or k greater than 56 as expected. Similarly, the 1024-node network works well with $k = 128$, as the time to inject 128 each packets is 55% of the SAT propagation delay and they can reach up to 0.55 phits/cycle. In the butterfly permutation (not shown) the average throughput is near full load, so it needs at least $l192k256$ to reach its maximum value, which is 0.94 phits/cycle.

As we can see in Table I, SAT improves throughput for the transpose permutation. It only reduces the peak throughput for the hot region traffic pattern, so it performs better in the larger network by providing fairness with nearly no lost of peak throughput. However, as the SAT parameters must increase with network size, the traffic will be more bursty at high loads.

To improve scalability, we must reduce the propagation time of the SAT signal. We could replace the Hamiltonian path with a spanning-tree so that the propagation time of the SAT grows logarithmically with the network size. We called this alternative implementation **Spanning-tree SAT (SS)**. The root switch distributes one SAT signal to each of the network switches in the lower level as shown in figure 6. Upon reception, each switch distributes it again using an one-to-all communication to the lower level, and wait for an all-to-one reply before sending an acknowledgement up. Therefore, the SAT interval only depends on the number of levels of the fat-tree, n, which need to be traversed twice, up and down, and the node response latency. Assuming single-cycle latencies the length of the SAT interval will be $I = 2n + 1$. In the 256-

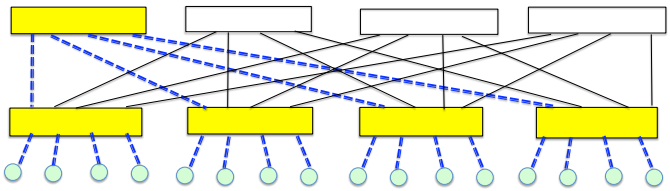

Fig. 6. Spanning tree path for SAT propagation in a $4 - ary\ 2 - tree$ network

TABLE I
NODE THROUGHPUT (PHITS/NODE/CYCLE) FOR LARGE INDIRECT NETWORKS UNDER 3 TRAFFIC PATTERNS.

Size	Injection policy	Uniform			Hot region			Transpose		
		Avge	Min	Max	Avge	Min	Max	Avge	Min	Max
256	none	0.531	0.517	0.542	0.213	0.126	0.622	0.418	0.320	0.779
	l32k32	0.529	0.528	0.530	0.205	0.205	0.206	0.548	0.547	0.550
	l48k48	0.528	0.527	0.530	0.205	0.204	0.206	0.555	0.553	0.556
	l56k56	0.529	0.526	0.530	0.206	0.205	0.208	0.546	0.545	0.549
256	SS-l2k2	0.529	0.529	0.530	0.204	0.204	0.205	0.575	0.574	0.575
	SS-l2k3	0.530	0.514	0.545	0.206	0.192	0.241	0.585	0.552	0.679
	SS-l4k4	0.529	0.528	0.529	0.204	0.204	0.205	0.573	0.573	0.574
1024	none	0.501	0.496	0.506	0.197	0.112	0.598	0.446	0.362	0.702
	l64k128	0.500	0.493	0.507	0.194	0.156	0.298	0.502	0.466	0.552
	l128k128	0.499	0.498	0.501	0.192	0.186	0.205	0.482	0.476	0.489
	l192k192	0.500	0.498	0.502	0.191	0.186	0.205	0.451	0.441	0.461
	l192k256	0.501	0.493	0.507	0.191	0.176	0.229	0.486	0.452	0.558
1024	SS-l2k2	0.501	0.499	0.502	0.191	0.190	0.191	0.536	0.535	0.537
	SS-l2k3	0.503	0.494	0.515	0.192	0.177	0.223	0.540	0.506	0.618
	SS-l4k4	0.501	0.500	0.502	0.191	0.190	0.192	0.535	0.534	0.536

node network, the SAT interval is reduced from 936 to only 9 cycles and in the 1024-node network the SAT interval is only 11 cycles. Thus, any small value of the parameter l will suffice.

Table I shows the network throughput for SS with parameters $l2k2$, $l2k3$ and $l4k4$. As expected when $k = l$ we achieve fairness; SS-l2k3 improves throughput but reduces fairness for most loads. The main advantage of the Spanning-tree SAT compared with the Hamiltonian SAT is that there is no need to tune the parameters for each workload, as both SS-l2k2 and SS-l4k4 work well for all traffic patterns, and for a wide range of network sizes. The only drawback is that it requires the network switches to act as SAT concentrators.

V. CONCLUSIONS

This work has measured throughput fairness in the k-ary n-tree topology under synthetic loads. The level of unfairness in the fat-tree is lower that in direct networks but the variations in node throughput are significant and cannot be ignored. As an adaptive network attempts to balance the load, it show less fairness issues that its static counterpart but it cannot prevent unfairness for most loads.

The SAT protocol, which was proposed to be used in a ring topology can be extended to any other topology by using a hamiltonian path to propagate the SATisfied signal. The cost of implementing this protocol is very low, needing a counter to keep track of the number of packets sent and a control line to propagate the signal from one node to the next.

We have evaluated SAT for the k-ary n-tree network under 6 synthetic traffic patterns for both static and adaptive networks, with sizes ranging from 64 to 1024 nodes. The fairness protocol works well for small to medium networks, increasing minimum node throughput for unbalanced loads. For large networks, the propagation delay results in a large value of the parameter l, forcing the nodes to inject their packets in large bursts. By replacing the hamiltonian path with a spanning tree we reduced SAT propagation delay so that this fairness protocol is able to scale to very large networks; besides, this alternative is easy to tune as the same value $l = 4$ work well for a range of workloads and network sizes.

Finally, when network traffic is unbalanced there is a trade-off between network fairness and peak average throughput, but minimum node throughput is more important so that slower nodes won't become the bottleneck of a parallel task.

REFERENCES

[1] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker, "The IBM Blue Gene/Q interconnection network and message unit," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 26:1–26:10.

[2] "Top 500," www.top500.org.

[3] M. Azimi, D. Dai, A. Kumar, and A. S. Vaidya, *On-chip Interconnect Trade-offs for Tera-scale Many-core Processors - Designing Network On-Chip Architectures in the Nanoscale Era*. Chapman and Hall, 2011.

[4] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[5] I. Cidon and Y. Ofek, "Metaring-a full-duplex ring with fairness and spatial reuse," *Communications, IEEE Transactions on*, vol. 41, no. 1, pp. 110 –120, jan 1993.

[6] C. Izu, "A throughput fairness injection protocol for mesh and torus networks," in *High Performance Computing (HiPC), 2009 International Conference on*, dec. 2009, pp. 294 –303.

[7] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *Computers, IEEE Transactions on*, vol. 54, no. 8, pp. 1025 – 1040, aug. 2005.

[8] C. Izu, "Throughput fairness in k-ary n-cube networks," in *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ser. ACSC '06. Darlinghurst, Australia: Australian Computer Society, Inc., 2006, pp. 137–145.

[9] F. J. R. Perez and J. Miguel-Alonso, "INSEE: An interconnection network simulation and evaluation environment," in *Euro-Par'05*, 2005, pp. 1014–1023.

[10] F. Petrini and M. Vanneschi, "k-ary n-trees: High performance networks for massively parallel architectures," in *Proceedings of the 11th International Parallel Processing Symposium, IPPS'97*, Geneva, Switzerland, April 1997, pp. 87–93.