Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2009-011 March 27, 2009

Oblivious Routing in On-Chip Bandwidth-Adaptive Networks

Myong Hyon Cho, Mieszko Lis, Keun Sup Shim, Michel Kinsy, Tina Wen, and Srinivas Devadas

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Oblivious Routing in On-Chip Bandwidth-Adaptive Networks

Myong Hyon Cho, Mieszko Lis, Keun Sup Shim, Michel Kinsy, Tina Wen and Srinivas Devadas
Massachusetts Institute of Technology, Cambridge, MA 02139
{mhcho, mieszko, ksshim, mkinsy, tinaw, devadas}@mit.edu

Abstract

Oblivious routing can be implemented on simple router hardware, but network performance suffers when routes become congested. Adaptive routing attempts to avoid hot spots by re-routing flows, but requires more complex hardware to determine and configure new routing paths. We propose on-chip bandwidth-adaptive networks to mitigate the performance problems of oblivious routing and the complexity issues of adaptive routing.

In a bandwidth-adaptive network, the bisection bandwidth of a network can adapt to changing network conditions. We describe one implementation of a bandwidth-adaptive network in the form of a two-dimensional mesh with adaptive bidirectional links, where the bandwidth of the link in one direction can be increased at the expense of the other direction. Efficient local intelligence is used to reconfigure each link, and this reconfiguration can be done very rapidly in response to changing traffic demands.

We compare the hardware designs of a unidirectional and bidirectional link and evaluate the performance gains provided by a bandwidth-adaptive network in comparison to a conventional network under uniform and bursty traffic when oblivious routing is used.

1. Introduction

Routers can be generally classified into oblivious and adaptive [21]. In oblivious routing, the path is completely determined by the source and the destination address. Deterministic routing is a subset of oblivious routing, where the same path is always chosen between a source-destination pair. Thanks to its distributed nature where each node can make its routing decisions independent from others, oblivious routing such as dimension-order routing [8] enables simple and fast router designs and is widely adopted in today's on-chip interconnection networks. On the other hand, today's oblivious routing algorithms can have difficulty with certain traffic patterns, especially when bandwidth demands of flows vary with time, because routes are not adjusted for different applications.

In adaptive routing, given a source and a destination address, the path taken by a particular packet is dynamically adjusted depending on, for instance, network congestion. With this dynamic load balancing, adaptive routing can potentially achieve better throughput and latency compared to oblivious routing. However, adaptive routing methods face a difficult challenge in balancing router complexity with the capability to adapt. To achieve the best performance through adaptivity, a router ideally needs global knowledge of the current network status. However, due to router speed and complexity, dynamically obtaining a global and instantaneous view of the network is often impractical. As a result, adaptive routing in practice relies primarily on local knowledge, which limits its effectiveness. If it is necessary to avoid out-of-order packet receipt at the destination, packets in transit on the original path of a flow all have to reach the destination before the network is reconfigured and packets are injected into the new path.

We propose *bandwidth-adaptive networks* to mitigate the problems of oblivious routing and avoid the complexity of adaptive routing. In a bandwidth-adaptive network, the bisection bandwidth of a network can adapt to changing network conditions. We describe one implementation of a bandwidth-adaptive network in the form of a two-dimensional mesh with adaptive bidirectional links[1], where the bandwidth of the link in one direction can be increased at the expense of the other direction. Efficient local intelligence is used to appropriately reconfigure each link, and this reconfiguration can be done very rapidly to respond to changing traffic demands. Reconfiguration logic compares traffic on either side of a link to determine how to reconfigure each link.

One can think of a bandwidth-adaptive link as a multi-lane freeway, where a subset or all of the lanes can be set up to carry traffic in either direction. Each lane carries traffic in one particular direction at any point of time, but can be easily switched to carry traffic in the opposite direction depending on the number of cars wishing to travel in each

[1] Bidirectional links have been referred to as half-duplex links in router literature.

Figure 1. Motivation for Bandwidth Adaptivity (from www.panaramio.com)

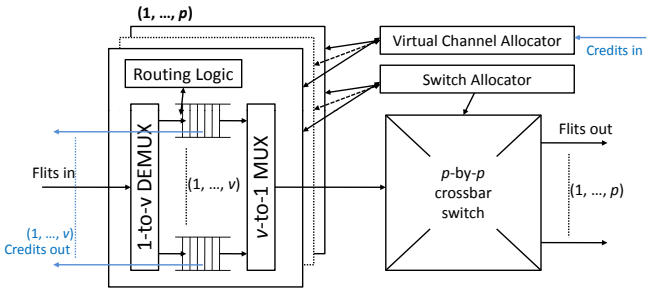

Figure 2. Typical virtual-channel router architecture with p physical channels and v virtual channels per physical channel.

direction. Figure 1 illustrates a scenario where this would be helpful!

We compare the hardware designs of a unidirectional and bidirectional link and argue that the hardware overhead of implementing bidirectionality and reconfiguration is reasonably small. We then evaluate the performance gains provided by a bandwidth-adaptive network in comparison to a conventional network through detailed network simulation of oblivious routing methods under uniform and bursty traffic, and show that the performance gains are significant.

In Section 2, we describe a hardware implementation of an adaptive bidirectional link, and compare it with a conventional unidirectional link. In Section 3, we describe schemes that determine the configuration of the adaptive link, i.e., decide which direction is preferred and by how much. The frequency of reconfiguration can be varied. Related work is summarized in Section 4. Simulation results comparing oblivious routing on a conventional network against a bandwidth-adaptive network are the subject of Section 5. Section 6 concludes the paper.

2. Adaptive Bidirectional Link

2.1. Typical Virtual Channel Router

Althoguh bandwidth adaptivity can be introduced independently of network topology and flow control mechanisms, in the interest of clarity we assume a typical virtual-channel router on a two-dimensional (2-D) mesh network as a baseline.

Figure 2 illustrates a typical virtual-channel router architecture and its operation [10, 23, 19]. As shown in the figure, the datapath of the router consists of buffers and a switch. The input buffers store flits waiting to be forwarded to the next hop; each physical channel often has multiple input buffers, which allows flits to flow as if there were multiple "virtual" channels. When a flit is ready to move, the switch connects an input buffer to an appropriate output channel. To control the datapath, the router also contains three major control modules: a router, a virtual-channel (VC) allocator, and a switch allocator. These control modules determine the next hop, the next virtual channel, and when a switch is available for each packet/flit.

The routing operation comprises four steps: routing (RC), virtual-channel allocation (VA), switch allocation (SA), and switch traversal (ST); these are often implemented as four pipeline stages in modern virtual-channel routers. When a head flit (the first flit of a packet) arrives at an input channel, the router stores the flit in the buffer for the allocated virtual channel and determines the next hop node for the packet (RC stage). Given the next hop, the router then allocates a virtual channel in the next hop (VA stage). The next hop node and virtual channel decision is then used for the remaining flits of the given packet, and the relevant virtual channel is exclusively allocated to that packet until the packet transmission completes. Finally, if the next hop can accept the flit, the flit competes for a switch (SA stage), and moves to the output port (ST stage).

2.2. Bidirectional Links

In the typical virtual-channel router shown in Figure 2, each output channel is connected to an input buffer in an adjacent router by a unidirectional link; the maximum bandwidth is related to the number of physical wires that constitute the link. In an on-chip 2-D mesh with nearest neighbor connections there will always be two links in close proximity to each other, delivering packets in opposite directions.

We propose to merge the two links between a pair of network nodes into a set of bidirectional links, each of which can be configured to deliver packets in either direction, increasing the bandwidth in one direction at the expense of the other. The links can be are driven from two different sources, with local arbitration logic and tristate buffers ensuring that both do not simultaneously drive the same wire.

Figure 3 illustrates the adaptivity of a mesh network us-

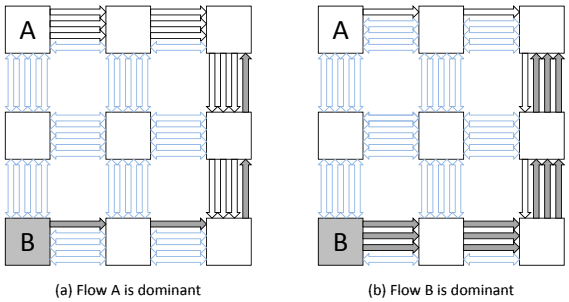

Figure 3. Adaptivity of a mesh network with bidirectional links

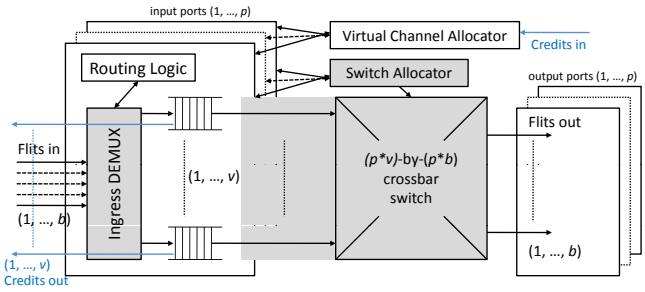

Figure 5. Network node architecture with u unidirectional links and b bidirectional links between each of p neighbor nodes and itself.

ing bidirectional links. Flow A is generated at the upper left corner and goes to the bottom right corner, while flow B is generated at the bottom left corner and ends at the upper right corner. When one flow becomes dominant, bidirectional links change their directions in order to achieve maximal total throughput. In this way, the network capacity for each flow can be adjusted by the flow burstiness without changing routes.

Figure 4 shows a bidirectional link connecting two network nodes (for clarity, only one bidirectional link is shown between the nodes, but multiple bidirectional links can be used to connect the nodes if desired). The bidirectional link can be regarded as a bus with two read ports and two write ports that are interdependent. A bandwidth arbiter governs the direction of a bidirectional link based on *pressure* (see Section 3) from each node, a value reflecting how much bandwidth a node requires to send flits to the other node. Bold arrows in Figure 4 illustrate a case when flits are delivered from right to left; a tri-state buffer in the left node prevents the output of its crossbar switch from driving the bidirectional link, and the right node does not receive flits as the input is being multiplexed. If the link is configured to be in the opposite way, only the left node will drive the link and only the right node will receive flits.

Router logic invalidates the input channel at the driving node so that only the other node will read from the link. The switching of tri-state buffers can be done faster than other pipeline stages of router architecture so that we can change the direction without dead cycles at which no flits can move in any direction. Note that if a dead cycle is required in a particular implementation, we can minimize performance loss by switching directions relatively infrequently. We discuss this tradeoff in Section 5.

2.3. Router Architecture with Bidirectional Links

Figure 5 illustrates a network node with b bidirectional links, where each link has a bandwidth of one flit per router cycle; gray blocks highlight modules modified from the baseline architecture shown in Figure 2. Adjacent nodes are connected via p ports (for the 2-D mesh we consider here, $p = 4$ at most). At each port, b input channels and b output channels share the b bidirectional links via tri-state buffers: if a given link is configured to be ingressive, its input channel is connected to the link while the output channel is disconnected, and vice versa (the output channels are not shown in the figure).

We assume, as in typical routers, that at most one flit from each virtual channel can be transferred in a given cycle—if there are v virtual channels in the router, then at most v flits can be transferred in one cycle regardless of the bandwidth available. If i out of b bidirectional links are configured to be ingressive at a router node, the node can receive up to i flits per cycle from the node across the link and send out up to $(b-i)$ flits to the other node. Since each incoming flit will go to a different virtual channel queue,[2] the ingress demultiplexer in Figure 5 can be implemented with b instances of a v-to-1 demultiplexer with tri-state buffers at the outputs; no additional arbitration is necessary between demultiplexers because only one of their outputs will drive the input of each virtual channel.

In a bidirectional router architecture, the egress link can be configured to exceed one flit per cycle; consequently, the crossbar switch must be able to consider flits from more than one virtual channel from the same node. In our architecture, the output of each virtual channel is directly connected to the switch and competes for an outgoing link, although one could equally well imagine a hierarchical solution where the v virtual channels are multiplexed to a smaller number of switch inputs.

In addition, the crossbar switch must now be able to drive all $p \cdot b$ outgoing links when every bidirectional link is configured as egressive. Consequently, the router requires a

[2]Recall that once a virtual channel is allocated to a packet at the previous node, other packets cannot use the virtual channel until the current packet completes transmission.

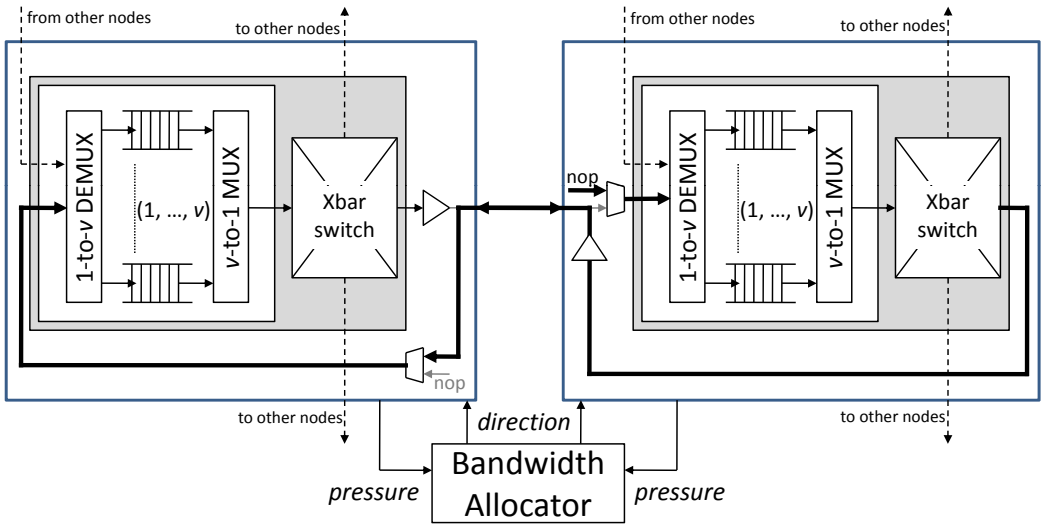

Figure 4. Connection between two network nodes through a bidirectional link (configured going left).

$p \cdot v$-by-$p \cdot b$ crossbar switch, compared to a $p \cdot v$-by-$p \cdot u$ switch of a conventional router with u unidirectional links; this larger switch is the most significant hardware cost of bidirectional router architecture. In designs where only w of v virtual channels compete for the switch at any given cycle, this cost difference increases, since the bidirectional router requires w to be larger to satisfy its potentially larger egress bandwidth.

To estimate the increased cost, we can parametrize architectures with and without bidirectional links by the number of unidirectional links u and the number of bidirectional links b; in this scheme, the conventional router architecture in Figure 2 has $u = 1$ and $b = 0$. Such a router requires a $p \cdot v$-by-$p \cdot u$ crossbar (or $p \cdot u$-by-$p \cdot u$ if only one virtual channel from each port competes for the switch at any given time). When each connection comprises more than two links, a hybrid architecture with some of the links bidirectional and some unidirectional (that is, $u > 0$ and $b > 0$) can reduce the hardware cost.

Table 1 summarizes the sizes of hardware components of unidirectional, bidirectional and hybrid router architectures assuming four virtual channels per ingress port (i.e., $v = 4$). While switch allocation logic grows as the size of crossbar switch increases and bidirectional routers incur the additional cost of the bandwidth allocation logic shown in Figure 4, these are insignificant compared to the increased size of the demultiplexer and crossbar. In our simulation experiments we used the crossbars that are in bold in the table.

Since the number of the crossbar input ports remains the same in both the unidirectional case and the bidirectional case, the number of crossbar output ports is the only factor increasing the crossbar size in bidirectional routers $(u,b) = (0,4)$ and $(1,2)$ when compared with the unidirectional $(2,0)$ case; this increase is size is roughly equal to the ratio of the output ports. Considering that a 32×32 crossbar takes approximately 25% of the gate count of a switch [16] with much of the actual area being accounted for by queue memory and wiring which is not part of the gate count, we estimate that a $1.5\times$ increase in crossbar size for the *u=1, b=2* case will increase the area of the node by $< 15\%$. If the queues are smaller, then this number will be larger.

Architecture	Ingress Demux	Xbar Switch
(u, b) = (1, 0)	one 1-to-4 demux	4-by-4 or **16-by-4**
(u, b) = (0, 2)	two 1-to-4 demuxes	**8-by-8**
(u, b) = (2, 0)	two 1-to-4 demuxes	8-by-8 or **16-by-8**
(u, b) = (0, 4)	four 1-to-4 demuxes	**16-by-16**
(u, b) = (1, 2)	three 1-to-4 demuxes	**16-by-12**

Table 1. The summary of differences in hardware components between 4-VC router architectures

To evaluate the flexibility and effectiveness of bidirectional links, we compare, in Section 5, the performance of bidirectional routers with $(u,b) = (0,2)$ and $(u,b) = (0,4)$ against unidirectional routers with $(u,b) = (1,0)$ and $(u,b) = (2,0)$, which, respectively, have the same total bandwidth as the bidirectional routers. We also consider a a hybrid architecture with $(u,b) = (1,2)$ which has the same total bandwidth as the $(u,b) = (2,0)$ and $(u,b) = (0,4)$ configurations.

3. Bandwidth Allocation in Bidirectional Links

Bidirectional links contain a bandwidth arbiter (see Figure 4) which governs the direction of the bidirectional links connecting a pair of nodes and attempts to maximize the connection throughput. Key to our approach are the locality and simplicity of this logic: the arbiter makes its decisions based on very simple information local to the nodes it connects.

Each network node tells the arbiter of a given bidirectional links how much *pressure* it wishes to exert on the link; this pressure indicates how much of the available link bandwidth the node expects to be able to use in the next cycle. In our design, each node counts the number of flits ready to be sent out on a given link (i.e., at the head of some virtual channel queue), and sends this as the pressure for that link. The arbiter then configures the links so that the ratio of bandwiths in the two directions approximates the pressure ratio, additionally ensuring that the bandwidth granted does not exceed the free space in the destination node. Consequently, if traffic is heavier in one direction than in the other, more bandwidth will be allocated to that direction.

The arbitration logic considers only the next-hop nodes of the flits at the front of the virtual channel queues and the available buffer space in the destination queues, both of which are local to the two relevant nodes and easy to compute. The arbitration logic itself consists of treshold comparisons and is also negligible in cost.

When each packet consists of one flit, the pressure as defined above exactly reflects the traffic that can be transmitted on the link; it becomes approximate when there are multiple flits per packet, since some of the destination queues with available space may be in the middle of receiving packets and may have been assigned to flows different from the flits about to be transmitted. Although more complex and accurate definitions of pressure are possible, our experience thus far is that this simple logic performs well in practice.

In some cases we may not want arbitration to take place in every cycle; for example, implementations which require a dead cycle after each link direction switch will perform poorly if switching takes place too often. On the other hand, switching too infrequently reduces the adaptivity of the bidirectional network, potentially limiting the benefits for quickly changing traffic and possibly requiring more complex arbitration logic. We explore this tradeoff in Section 5.

When analyzing link bandwidth allocation and routing in a bidirectional adaptive network, we must take care to avoid additional deadlock due to bidirectional links, which may arise in some routing schemes. Consider, for example, the situation shown in Figure 6: a flow f_B travels from node B to node C via node A, and all links connecting A with

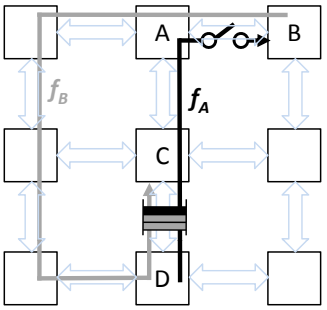

Figure 6. Deadlock on deadlock-free routes due to bidirectional links

B are configured in the direction $B \rightarrow A$. Now, if another, smaller flow f_A starts at D and heads for B, it may not exert enough pressure on the $A \rightarrow B$ link to overcome that of f_B, and, with no bandwidth allocated there, may be blocked. The flits of f_A will thus eventually fill the buffers along its path, which might prevent other flows, including f_B, from proceeding: in the figure, f_B shares buffering resources with f_A between nodes C and D, and deadlock results. Note that the deadlock arises only because the bidirectional nature of the link between A and B can cause the connection $A \rightarrow B$ to disappear; since the routes of f_A and f_B obey the west-first turn model [13], the deadlock does not arise in the absence of bidirectional links. One easy way to avoid deadlock is to require, in the definition of pressure, that some bandwidth is always available in a given direction if some flits are waiting to be sent in that direction.

4. Related Work

4.1 Routing Techniques

A basic deterministic routing method is dimension ordered routing (DOR) [8] which becomes XY routing in a 2-D mesh. ROMM [20] and Valiant [27] are classic oblivious routing algorithms, which are randomized in order to achieve better load distribution. In o1turn [24], Seo *et al* show that simply balancing traffic between XY and YX routing can guarantee provable worst-case throughput. A weighted ordered toggle (WOT) algorithm that assumes 2 or more virtual channels [12] assigns XY and YX routes to source-destination pairs in a way that reduces the maximum network load for a given traffic pattern. While we have focused on dimension-ordered routing in this paper due to its speed and simplicity, other methods can be used in conjunction with bandwidth adaptivity.

Classic adaptive routing schemes include the turn routing methods [13] and odd even routing [5]. These are general

schemes that allow packets to take different paths through the network while ensuring deadlock freedom but do not specify the mechanism by which a particular path is selected. An adaptive routing policy determines what path a packet takes based on network congestion.

Adaptive routing policies can be classified as either congestion-oblivious or congestion-aware, based on whether they take output link demand into account [14]. Some examples of congestion-oblivious routing strategies are random [11], zigzag [2] and no-turn [13]. Congestion-aware routing policies use various metrics to determine congestion. For example, Dally and Aoki [7] favor the port with the largest number of available virtual channels, and give results that have better performance than congestion-oblivious algorithms. In [15] a scheme that switches between deterministic and adaptive modes depending on the application is presented, where local FIFO information is used to adapt routes. Buffer availability at adjacent routers has been used as a congestion metric [18], as well as output queue length [25, 26]. These routing algorithms all rely on local congestion indicators. Regional Congestion Awareness (RCA) [14] is an adaptive routing approach that propagates congestion information across the network in a scalable manner, improving the ability of adaptive routers to spread network load.

We have used oblivious routing methods in this paper, and therefore the hardware requirements are smaller than for conventional adaptive routing methods. The router only has to support DOR, and we have used simple, local congestion metrics to determine how best to configure each link. Rather than making decisions on a per-packet basis, our network makes decisions on a per-link basis.

4.2 Router Designs

Dally's virtual channels [6] allocate buffer space for virtual channels in a decoupled way from bandwidth allocation. Many designs of virtual channel routers have been proposed (e.g., [19], [3], [17], [22]). Our virtual channel router design is modified to enable adaptive bidirectional links in the network.

Router designs with bidirectional or half-duplex links have been proposed. For example, Ariadne [1], the Intel Cavallino [4] and NetworkDesignFrame [9] use half-duplex links, with the Cavallino using simultaneous bidirectional signalling.

5. Results and Comparisons

5.1. Experimental Setup

A cycle-accurate network simulator was used to model the bidirectional router architectures with different combinations of unidirectional (u) and bidirectional (b) links in each connection (see Figure 7 and Table 2 for details). To evaluate performance under various loads, we employed a set of standard synthetic traffic patterns (*transpose*, *bit-complement*, *shuffle*, and *uniform-random*) both without burstiness and with a Markov Modulated Process (MMP) bursty traffic model. We also examined several frequencies of bandwidth allocation to estimate the impact on architectures where a dead cycle is required to switch the link direction.

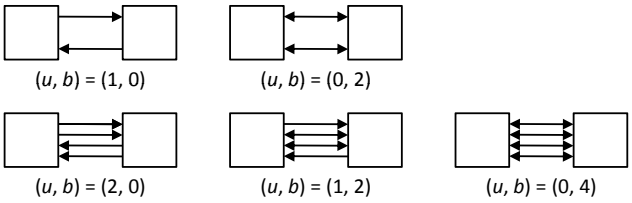

Figure 7. Link configurations used in the experiments

Characteristic	Configuration
Topology	8x8 2D MESH
Link configuration	(u, b) = (1,0), (0,2) (2,0), (1,2), (0,4)
Routing	DOR-XY and DOR-YX
Per-hop latency	1 cycle
Virtual channels per port	4
Flit buffers per VC	4
Average packet length (flits)	8
Traffic workload	transpose, bit-complement, shuffle, uniform random
Burstiness model	Markov modulated process
Warmup cycles	20,000
Analyzed cycles	100,000

Table 2. Summary of network configuration

Although the bidirectional routing technique applies to various oblivious routing algorithms, we have, for evaluation purposes, focused on Dimension Ordered Routing (DOR), the most widely implemented oblivious routing method. While our experiments included both DOR-XY and DOR-YX routing, we did not see significant differences in the results, and consequently report only DOR-XY results. In all of our experiments, the router was configured for four virtual channels per ingress port under a dynamic virtual channel allocation regimen.

5.2. Non-bursty Traffic

Figure 8 shows the throughput in the unidirectional and bidirectional networks under non-bursty traffic. When traf-

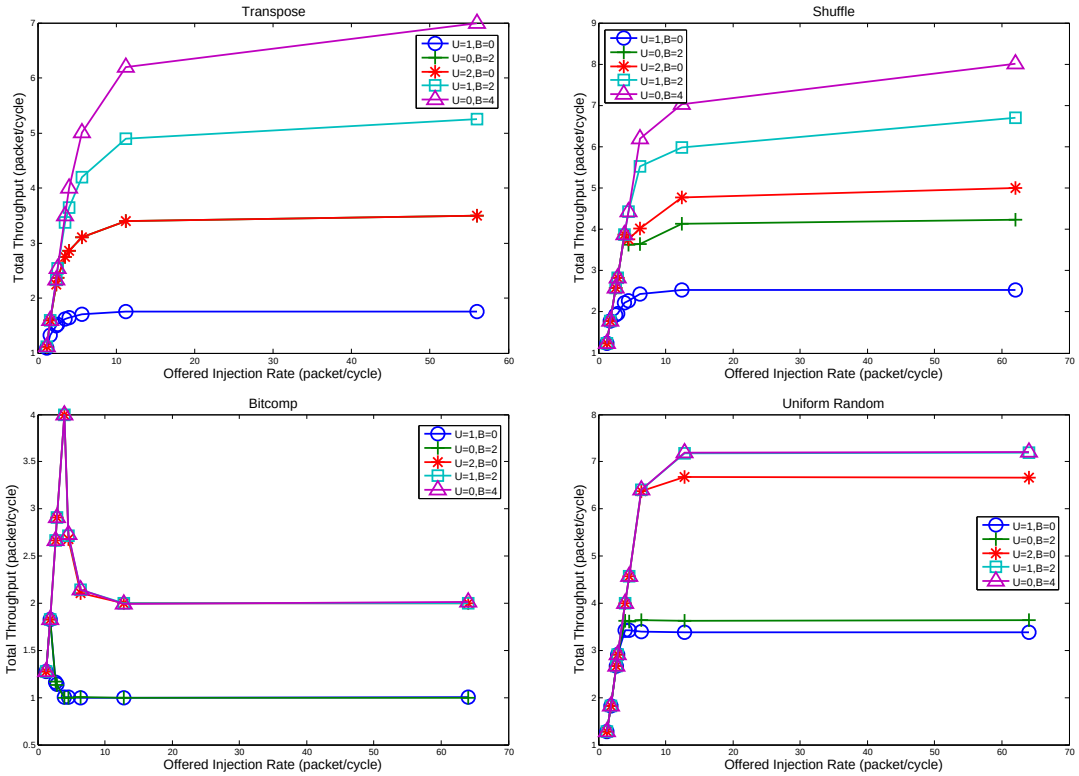

Figure 8. Throughput under non-bursty traffic

fic is consistent, the improvement offered by bidirectional links depends on how symmetric the flows are. On the one extreme, *bit-complement*, which in steady state is entirely symmetric when routed using DOR and results in equal traffic in each direction on any link, shows no improvement; on the other extreme, in *transpose*, packets move in only one direction over any given link, and bidirectional links improve throughput twofold. *Shuffle* lies between the two extremes, with the bidirectional network outperforming the unidirectional solution by 60% when total bandwidth is equal.

Uniformly random traffic is also symmetric when averaged over a period of time. For very short periods of time, however, the symmetry is imperfect, allowing the bidirectional network to track the traffic shifts as they happen and outperform the unidirectional network throughput by up to 8%.

5.3. Bursty Traffic

The temporary nature of bursty traffic allows the bidirectional network to adjust the direction of each link to favor whichever direction is prevalent at the time, and results in throughput improvements across all traffic patterns (see Figure 9). With bursty traffic, even *bit-complement*, for which the bidirectional network does not win over the unidirectional case without burstiness, shows a 20% improvement in total throughput because its symmetry is broken over short periods of time by the bursts. For the same reason, *shuffle* and *uniform-random* outperform the unidirectional network by 66% and 26% respectively, compared to 60% and 8% in non-bursty mode. Finally, *transpose* performance is the same as for the non-bursty case, because the traffic, if any, still only flows in one direction and requires no changes in link direction after the initial adaptation.

5.4. Link Arbitration Frequency

So far, our results have assumed that the bandwidth arbiter may alter the direction of every link on every cycle. While we believe this is very realistic, we also considered the possibility that switching directions might require a dead cycle, in which case changing too often could limit the throughput up to 50% in the worst case. We therefore reduced the arbitration frequency and examined the tradeoff between lessening the impact of a dead cycle every N cycles to $\frac{1}{N+1}$ and limiting the network's adaptivity to rapid changes in traffic patterns. The results in this section illustrate the relevant tradeoffs.

Figure 10 shows how often each bidirectional link actu-

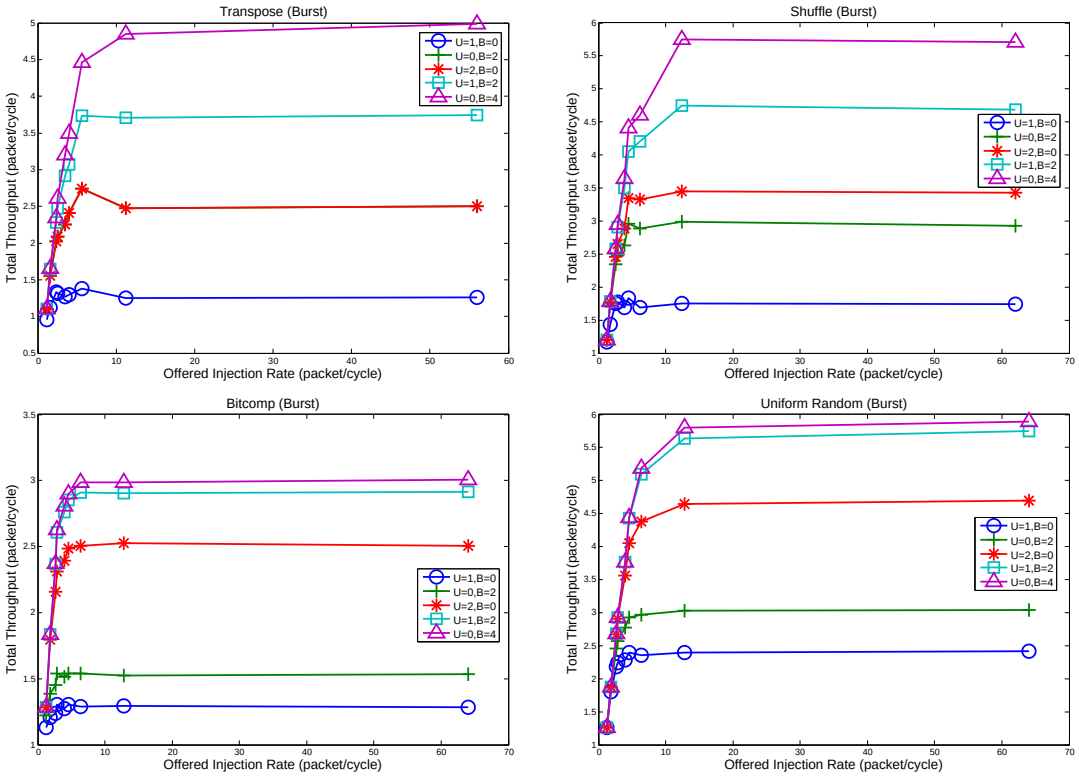

Figure 9. Throughput under bursty traffic

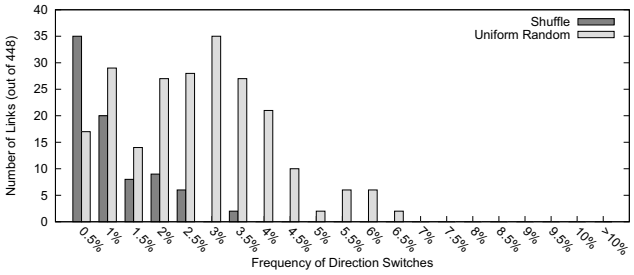

Figure 10. The frequency of direction changes on bidirectional links

ally changes its direction under bursty *shuffle* and *uniform-random* traffic: the x-axis shows how frequently links directions change and the y-axis shows how many links switch that often. For example, under *shuffle* traffic, about 8% of bidirectional links change their direction less than once every two hundred cycles. Traffic exhibiting the *uniform-random* pattern, in comparison, is more symmetric than *shuffle*, and so the link directions change more often.

The observation that no link changes its direction more frequently than once in ten cycles led us to investigate how infrequent the link switches could be without significantly affecting performance. In Figure 11 we compare the performance of the bidirectional network under different link arbitration frequencies; as expected, throughput decreases when the links are allowed to switch less often.

Even with a switching period as large as 100 cycles, the bidirectional network still significantly outperforms the unidirectional design under many loads (e.g., by more than 20% for *shuffle*). In the case of *uniform-random*, however, the bidirectional network performance trails the unidirectional design when switching is infrequent. This is because, when each link arbitration decision lasts 100 cycles, any temporary benefit from asymmetric bandwidth allocation is nullified by changes in traffic patterns, and, instead of improving throughput, the asymmetric allocations only serve to throttle down the total throughput compared to the unidirectional router.

Infrequent link switching, therefore, demands a more sophisticated link bandwidth arbiter that bases its decisions on the pressures observed over a period of time rather than on instantaneous measurements. For *uniform-random*, for example, the symmetry of uniform random traffic over time would cause the link bandwidths to be allocated evenly, allowing it to match the performance of the unidirectional network.

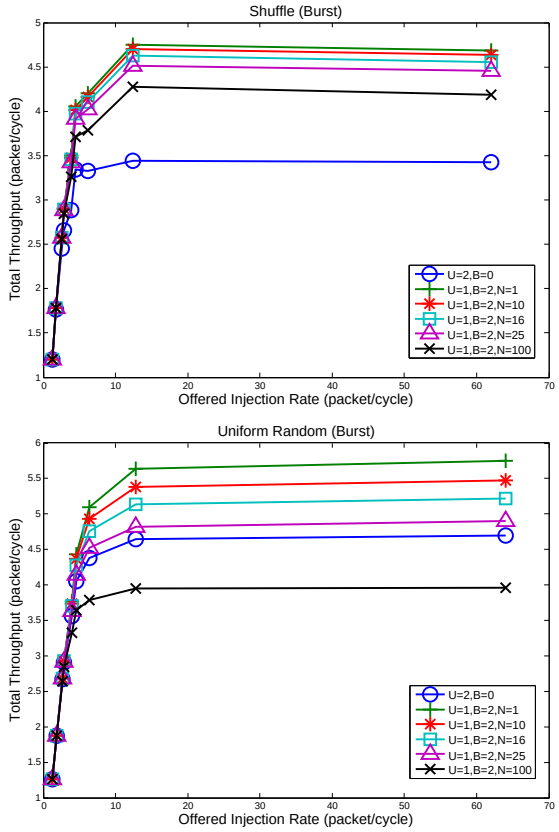

Figure 11. Bursty traffic throughput with different link arbitration periods (N)

6. Conclusions

We have proposed the notion of bandwidth-adaptive networks in this paper, given one concrete example of bidirectional links in a 2-D mesh, and evaluated it. Adaptivity is controlled by local pressure that is easily computed. While more comprehensive evaluation is underway, adaptive bidirectional links provide better performance under both uniform and bursty traffic for the tested benchmarks.

While we have focused on a mesh, adaptive bidirectional links can clearly be used in other network topologies. In adaptive routing decisions are made on a per-packet basis at each switch. In bandwidth-adaptive networks, decisions are made on a per-link basis. We believe this difference makes bandwidth-adaptivity more amenable to local decision making, though more rigorous analysis is required.

References

[1] J. D. Allen, P. T. Gaughan, D. E. Schimmel, and S. Yalamanchili. Ariadne—an adaptive router for fault-tolerant multicomputers. *SIGARCH Comput. Archit. News*, 22(2), 1994.

[2] H. Badr and S. Podar. An optimal shortest-path routing policy for network computers with regular mesh-connected topologies. *IEEE Transactions on Computers*, 38(10):1362–1371, 1989.

[3] T. Bjerregaard and J. Sparsø. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In *Proceedings of the IEEE Norchip Conference (NORCHIP 2004)*. IEEE, 2004.

[4] J. Carbonaro and S. Verhoorn. Cavallino: The teraflops router and nic. In *Proceedings of the Fourth Symp. High-Performance Interconnects (Hot Interconnects 4)*, August 1996.

[5] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):729–738, 2000.

[6] W. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 03(2):194–205, 1992.

[7] W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 04(4):466–475, 1993.

[8] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Trans. Computers*, 36(5):547–553, 1987.

[9] W. J. Dally and P. Song. Design of a self-timed vlsi multicomputer communication controller. In *Proceedings of International Conference on Computer Design (ICCD-87)*, pages 230–234, 1987.

[10] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[11] W.-C. Feng and K. G. Shin. Impact of selection functions on routing algorithm performance in multicomputer networks. In *In Proc. of the Int. Conf. on Supercomputing*, pages 132–139, 1997.

[12] R. Gindin, I. Cidon, and I. Keidar. Noc-based fpga: Architecture and routing. In *First International Symposium on Networks-on-Chips (NOCS 2007)*, pages 253–264, 2007.

[13] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *J. ACM*, 41(5):874–902, 1994.

[14] P. Gratz, B. Grot, and S. W. Keckler. Regional Congestion Awareness for Load Balance in Networks-on-Chip. In *In Proc. of the 14th Int. Symp. on High-Performance Computer Architecture (HPCA)*, pages 203–214, Feb. 2008.

[15] J. Hu and R. Marculescu. DyAD: Smart Routing for Networks on Chip. In *Design Automation Conference*, June 2004.

[16] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos. Variable packet size buffered crossbar (cicq) switches. In *2004 IEEE International Conference on Communications*, volume 2, pages 1090–1096, June 2004.

[17] N. K. Kavaldjiev, G. J. M. Smit, and P. G. Jansen. A virtual channel router for on-chip networks. In *IEEE Int. SOC Conf., Santa Clara, California*, pages 289–293. IEEE Computer Society Press, Sept. 2004.

[18] H. J. Kim, D. Park, T. Theocharides, C. Das, and V. Narayanan. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of Design Automation Conference*, pages 559–564, June 2005.

[19] R. D. Mullins, A. F. West, and S. W. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proc. of the 31st Annual Intl. Symp. on Computer Architecture (ISCA)*, pages 188–197, 2004.

[20] T. Nesson and S. L. Johnsson. ROMM routing on mesh and torus networks. In *Proc. 7th Annual ACM Symposium on Parallel Algorithms and Architectures SPAA'95*, pages 275–287, 1995.

[21] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[22] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. ViChaR: A dynamic virtual channel regulator for network-on-chip routers. In *Proc. of the 39th Annual Intl. Symp. on Microarchitecture (MICRO)*, 2006.

[23] L.-S. Peh and W. J. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proc. International Symposium on High-Performance Computer Architecture (HPCA)*, pages 255–266, Jan. 2001.

[24] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA 2005)*, pages 432–443, 2005.

[25] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles. Goal: a load-balanced adaptive routing algorithm for torus networks. *SIGARCH Comput. Archit. News*, 31(2):194–205, 2003.

[26] A. Singh, W. J. Dally, B. Towles, and A. K. Gupta. Globally adaptive load-balanced routing on tori. *IEEE Comput. Archit. Lett.*, 3(1), 2004.

[27] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 263–277, 1981.