Adaptive Routing Strategies for Modern High Performance Networks

Patrick Geoffray
Myricom, Inc.
Software Development Laboratory
Arcadia, CA, USA
patrick@myri.com

Torsten Hoefler
Indiana University
Open Systems Laboratory
150 S Woodlawn Ave, Bloomington, IN 47405
htor@cs.indiana.edu

Abstract

Today's scalable high-performance applications heavily depend on the bandwidth characteristics of their communication patterns. Contemporary multi-stage interconnection networks suffer from network contention which might decrease application performance. Our experiments show that the effective bisection bandwidth of a non-blocking 512-node Clos network is as low as 38% if the network is routed statically. In this paper, we propose and analyze different adaptive routing schemes for those networks. We chose Myrinet/MX to implement our proposed routing schemes. Our best adaptive routing scheme is able to increase the effective bisection bandwidth to 77% for 512 nodes and 100% for smaller node counts. Thus, we show that our proposed adaptive routing schemes are able to improve network throughput significantly.

1 Introduction

As High Performance Computing (HPC) systems continue to increase in size, a growing part of the applications' performance is driven by the interconnect implementation. Beyond simple point-to-point metrics, one key measure of the quality of the interconnect is the bisection bandwidth. It is defined as the total bandwidth between two equal sections of the network with the minimum number of links between them. However, most vendors only define this metric in terms of the number of links, totally oblivious to the effect of contention on these links.

A new metric, called *effective bisection bandwidth*, has been introduced in [6] to provide a more application-oriented definition. The effective bisection bandwidth is defined as the average link bandwidth that is available to an application for all possible bisect communication patterns. A bisect communication pattern is a pattern where each node communicates with exactly one other node.

The scalability of many applications is determined by the scalability of collective communication. Most implementations of collective communication assume certain bandwidth guarantees from the network. In reality, contention due to head-of-line blocking limits the effective bisection bandwidth of common Multistage Interconnection Networks (MINs), such as Myrinet [1], Quadrics [13] and InfiniBand [18]. Simulations of statically-routed Clos and Fat tree networks show that the average bandwidth of certain collective patterns, such as the *dissemination pattern* [3], is reduced to 40% on statically-routed networks that offer full bisection bandwidth.

In this paper, we first measure the impact of static routing on the effective bisection bandwidth. Then, we implement and evaluate basic dispersive routing mechanisms, before proposing an adaptive routing scheme that increases the effective bisection bandwidth.

The following section defines the context of our experiments, followed by a discussion of related work in Section 3. Section 4 explains our approach to the routing problem followed by the presentation of our benchmark results.

2 Context

This section defines the environment for our implementation. We explain certain distinctive properties of the interconnection network we use, the target MIN topology, and several routing-related issues. The following section discusses the Myrinet/MX interconnect.

2.1 Myrinet / MX

Myrinet[1] is a well-known high-speed interconnect developed in 1994 by Myricom, based in part on prior research performed at Caltech under DARPA sponsorship. Myrinet uses source-routing, cut-through switching, and byte-level hardware flow-control to implement a lossless low-latency fabric. The Network Interface Card (NIC) contains an embedded processor executing firmware that implements the

1550-4794/08 $25.00 © 2008 IEEE
DOI 10.1109/HOTI.2008.21

low-level communication stack. This firmware-based approach allows for greater flexibility in refining existing semantics or adding new functionality. Myri-10G is the latest generation of Myricom hardware. It uses the physical layer of 10-Gigabit Ethernet, and either Myrinet or Ethernet protocol at layer 2. The 10-Gigabit Ethernet link has a data rate of 10 Gb/s (12.5 Gbaud/s or 10.3125 Gbaud/s signal rate, depending on the PHY used).

The Myrinet Express (MX) software interface is composed of the firmware running on the NIC, the driver, and the user-level library implementing the MX API. MX has connection-oriented semantic, but a connection-less implementation. No Queue Pairs are established between processes; a single MX endpoint in a process can communicate with any other endpoint on the fabric, while using a constant memory footprint (including buffers). At its heart, MX uses a mix of Active Messages and Remote Memory Access primitives, implementing fully asynchronous communications, even for large messages. MX provides in-order matching but imposes no ordering requirement regarding delivery and completion. It is thus particularly well-suited for multi-rail configurations where packets can be sent over multi-path networks.

Myrinet networks support arbitrary topologies. However, the common topology for full-bisection-bandwidth configurations is the Clos network. The following section describes this network topology and its parameters in detail.

2.2 Clos Topology

The Clos network was designed as a telephone switching network in 1953 [4]. It falls into the category of Multistage Interconnection Networks and can be built with any odd number of stages. Each network can be described by three parameters: n and m that describe the input and output port counts of the first layer switches and r which defines the number of first layer switches. The middle layer consists of m r-port switches. Clos networks are strictly non-blocking if $m \geq 2n-1$ and rearrangably non-blocking if $m \geq n$. Technology and economics drive the usage of fully-connected crossbars, for which $m = n$, as basic building blocks. Thus, Clos networks used in today's HPC networks are rearrangably non-blocking.

This design guarantees that, in a full Clos network, for a given traffic pattern, there is at least one set of routes that provides full bisection bandwidth. However, in statically-routed networks, the same set of routes may generate heavy contention for a different pattern. In other words, the optimal set of routes depends on the traffic pattern.

Clos networks using small crossbars can be rather inefficient. The k-ary n-tree network [8], also called *fat tree*, is commonly used to optimize this network design by combining the crossbars in the middle. Those networks have very similar properties to Clos networks, and k-ary n-tree networks can be built with full bisection bandwidth like Clos networks. Similarly, the effective bandwidth depends on the traffic pattern when using static routes.

2.3 Source Routing

Myrinet, like other high-performance interconnects such as Quadrics, uses source routing. The entire path of each packet is known when it is injected into the network fabric. The routing information, composed of a byte for every crossbar traversed, is prepended to each packet header. This design limits the complexity and the processing latency of the crossbars because no local routing decision has to be performed at each hop. Furthermore, source routing allows for concurrent use of multiple paths in the network. In fact, the sender NIC can arbitrarily choose a different route on a per-packet basis.

When using the Up*Down* [16] routing algorithm, the Clos topology provides multiple shortest-path deadlock-free redundant routes between each pair of nodes. For a full p-stage Clos composed of n-port crossbars, there are $(n/2)^{\lfloor (p/2) \rfloor}$ different routes available. It is important to note that packet ordering is only guaranteed for a single route. Packets sent along different routes in the fabric may (and in case of contention most likely will) not arrive in order.

2.4 Flow Control

Myrinet crossbars implement a cut-through switching, variant known as *wormhole* switching. When the head of a packet arrives at an input port of a crossbar, the first routing byte is used to find the requested output port and is then discarded. If the output port is unused at this time, the packet is immediately forwarded; if the packet is long enough, it may in fact traverse several crossbars simultaneously, thus the term *wormhole* switching.

If the requested output port is busy, the packet will be blocked. This situation is called *Head of Line* (HOL) blocking and this is the main source of contention in modern interconnects. While the head is blocked, the tail of the current packet and eventually the following packets will be temporarily received into the input port buffer. When this buffer becomes full, the hardware uses STOP/GO symbols to back-pressure the previous hop, eventually stopping the sender NIC itself. Since packets are finite in size and since the crossbars' scheduling is fair, the worst HOL blocking at each crossbar is bounded in time to guarantee progress. However, this contention does reduce the effective bandwidth of the links.

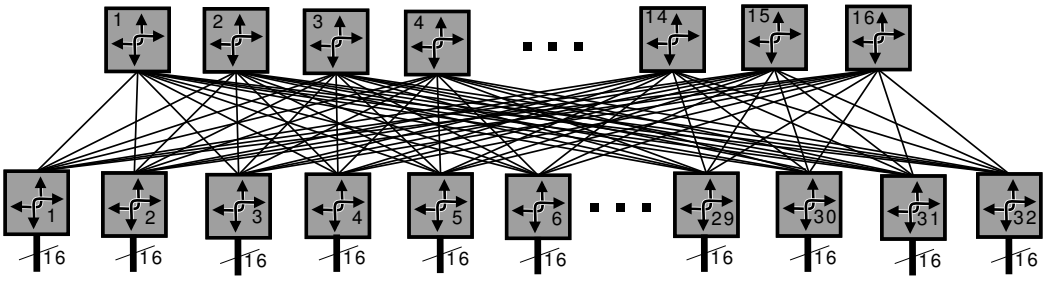

Figure 1. A 512-node Clos network with full bisection bandwidth built from 32-port crossbars

3 Related Work

Several researchers identified the problems in modern interconnect fabrics [14]. One approach is to reduce the impact of hot-spot contention and head-of-line blocking by throttling the sender [15]. Martinez et al. proposed a fully adaptive routing scheme for InfiniBand networks [10] that requires significant changes to the crossbar switches and thus is currently not implemented. A simpler scheme for InfiniBand which uses multiple endpoint addresses (LIDs) per node has been proposed in [9] and implemented in [19]. Another scheme to optimize routing in fat-tree networks that requires global routing information and a specialized hardware has been proposed in [5]. Boppana et al. [2] compare different adaptive wormhole algorithms by simulation.

One practical implementation that leverages a decentralized routing scheme and does not require global routing knowledge has been implemented for InfiniBand [19]. However, several technical problems, for example the requirement of large routing tables in all switches and the huge number of endpoint contexts (InfiniBand Queue Pairs) render this approach rather impractical for medium to large-scale networks. Furthermore, the used simple round-robin scheme still suffers from head-of-line blocking.

Quadrics[13] implements a form of adaptive routing by selectively using destination routing instead of source routing: the crossbars themselves select a free path on the ascent in the fat tree, and use the corresponding fixed route on the descent. However, this method is still subject to HOL blocking on the way down.

More recently, Woven Systems demonstrated full bisection bandwidth through their 144-port switch on a 128-node Ethernet cluster at Sandia[17]. There are few details on Woven's Active Congestion Management besides the use of custom chips on each port of the switch to monitor the traffic and rebalance the streams.

We will discuss a different approach to the problem, called *dispersive routing*, in the next section.

4 Dispersive Routing

In source-routed networks, each packet carries its own routing information. Therefore, it is possible to assign different routes through the fabric on a per-packet basis. This method of sending packets along different paths is called *dispersive* routing.

4.1 Randomized Oblivious Routing

A first approach to solve the contention problem is to randomly change route for each packet. This *randomized oblivious* routing algorithm always tries to spread traffic on all available links, without considering any other factors.

This method statistically reduces contention by transforming any structured traffic into pseudo-random traffic. As presented in [7], a single full crossbar yields a theoretical efficiency of 58.6% under such random traffic. Altough this result cannot be directly applied to a Clos network composed of multiples crossbars, it indicates that ramdomized oblivious routing could be more efficient for the average case than static routing.

Besides the statistical limitation of the network efficiency, this algorithm generates the worst amount of out-of-order packets. Fortunately, MX was designed to not require strong ordering on the wire. However, other low-level network protocols such as InfiniBand do currently rely on per-connection in-order arrival at the receiving NICs and are forced to drop out-of-order packets.

4.2 Adaptive Routing

All modern high-speed networks implement some sort of flow control in hardware, such as the one described in Section 2.4. When there is contention in the fabric, the sender NIC will eventually stop injecting packets. On Myrinet, the NIC firmware knows when the back-pressure flow control is stopping outgoing traffic. This information can be used to sense contention in the network and only use dispersive routing when contention occurs.

This *adaptive* routing behaves as static routing when no contention is sensed, sending packets in order along a single route. However, it does change routes randomly when a packet is stopped due to flow control. In the worst case, when all packets are blocked, this method degenerates into the randomized oblivious routing.

By keeping a single route when there is no contention, adaptive routing does improve the effective bisection by not using resources on other links, and thus reduces the probability of head-of-line blocking. This algorithm can reach full bisection bandwidth if all pairs eventually settle on a non-blocking route.

4.2.1 Adaptive dispersion threshold

However, this routing scheme has several limitations. First of all, back-pressure is not a precise indicator of contention. It is certainly evidence of some contention somewhere in the fabric, but input buffers on each crossbar are large enough to contain a full 4KB packet. Thus, the contention may be generated by any of the last few packets sent.

As a practical example, a series of packets is sent to a remote NIC: the first packet is blocked at the third hop on its route, the subsequent second and third packets are buffered along the way and the NIC detects that the fourth packet is stopped by flow control. It changes route for the fifth packet for this destination, hoping to work around the contention. However, the second and third packet are still using the first route, and will probably be blocked as well. The NIC will thus change route two more times whereas the second route may have been the non-blocking one.

To alleviate this issue, the dispersion is enabled only after the number of blocked packets exceeds a specified threshold, at least as large as the number of packets that can be buffered in the switch fabric. Increasing this threshold further allows for better contention detection, at the price of slower convergence.

4.2.2 Adaptive dispersion probability

Interference between pairs is another limitation. As the Myrinet crossbars are fair, two incoming streams of packets competing for the same output port will be processed fairly, one packet of each stream at a time. So when two routes share a link in the network, the hardware flow control will eventually stop both sender NICs, for half of the time if all packets are of the same size. Using the adaptive routing, both sender NICs will change routes at the same time, potentially interfering with two other streams, and so on.

A dampening mechanism is needed to prevent such a chain reaction. Upon sensing a contention, the sender NIC will have a specific probability of changing route, while keeping the current route otherwise. This probability can be set to reduce the cases where both senders disperse at the same time, again at the cost of slower convergence.

4.3 Probing Adaptive Routing

The instability of the adaptive routing is due to the fact that the new route may not be better than the old one in terms of contention. In fact, the new route is very likely to share a link with another communication pair. In order to converge to a non-blocking set of routes, the dispersive routing mechanism needs to change route only when the new route is known to be contention-free.

For this purpose, it is possible to *probe* an alternate route to estimate the suitability of the new path. This Myrinet-specific implementation allows the probing of a new path in the order of microseconds, without interfering with potentially existing traffic since the probe packets require very little bandwidth. If it has been established that the alternate route is already in use, another route is randomly probed. This repeats until an unloaded path is finally found or until all possible routes have been probed multiple times, at which point the route is changed to force another sender to evaluate other routes.

4.4 Convergence

None of the adaptive routing schemes presented above rely on global knowledge. Such knowledge of all current communications on the fabric at a given time would allow for a near-perfect scheduling of resources, resulting in full bisection bandwidth. However, the time required to gather all this information in a central location for a global analysis and then disseminate the scheduling back to each NIC would be substantial for large networks, possibly rendering the scheduling information obsolete by the time is it disseminated.

On the contrary, a local decision fueled by local knowledge does not require communication between peers and can adapt faster to changes in the local context. However, trying to converge to a global solution (that we know always exists on a Clos topology) using only local decisions is a well-known problem that genetic algorithms and other statistical refinement techniques have been tackling for years. Similar to the global knowledge case, the system may not converge to a global solution in time to be relevant.

4.4.1 Number of routes

Several parameters influence the speed of convergence. The size of the crossbars dictate the number of possible paths in the Clos network. The larger the crossbars, the larger the number of routes from which to choose. Furthermore, as the next route to disperse (or to probe) is chosen randomly, the cost of trying all possible paths is not linear.

However, since it is possible to react to contention on a per-packet basic, the granularity of the route change is in the order of the transmission time of a packet, modulated by the dispersion threshold/probability. For large messages, the chosen Maximum Transfer Unit (MTU) on Myrinet/MX is 4KB, which takes 3.5 μs to be injected into the network. For the probing adaptive scheme, the probing time is equivalent to a round-trip time, in the order of 4 μs. If it takes on average $2n$ packets to find a non-blocking path among n routes, the minimum message size needed to converge would be 128 KB for 4 KB packets on a Clos with 16 possible routes.

Furthermore, for a given crossbar size, the number of possible routes between two peers grows dramatically with the number of stages. Per Section 2.3, there are 16 possible routes in a 512-node Clos network built from 32-port crossbars, but this number jumps to 256 routes for 8192 nodes. At this scale, it is not practical to store all the routes in the NIC memory, and it becomes necessary to fetch them from host memory when needed or use only a subset.

4.4.2 Local optimum

The Clos topology guarantees that there is at least one globally optimal solution. However, it is possible for part of a random bisection system to converge to a local optimum that prevents the rest of the traffic to converge. For example, a NIC may never find an unused path among all possible routes because each path has a link that is in use by another communication. In this case, it is necessary to break the local optimum and hope for later convergance toward a global optimum.

Only the probing adaptive scheme is affected by this situation. Indeed, a route dispersion occurs when an alternate path is determined to be unused. If none of the possible routes are eligible, the sender NIC will continuously probe different paths, never dispersing. This situation can be handled by limiting the number of unsuccessful probes and force dispersion when this limit has been reached. This reaction can be enhanced by temporarily elevating the dispersion threshold following such forced dispersion, in order to force the node part of the local optimum to lose the battle and disperse first. In any event, breaking a local optimum is costly in term of packets, and would require very large messages to finally converge.

5 Performance Evaluation

In this section, we present a benchmark methodology to assess the effective bisection bandwidth and show results of the different techniques discussed in the previous sections.

5.1 Experimental Testbed

The performance evaluation was performed on a 512-node Myrinet cluster at the University of Southern California. Each node is equipped with two Intel Quad-Core Xeon E5420 at 2.5 GHz, 12 or 16 GB of memory and one single-port Myri-10G 10-Gigabit Myrinet NIC in a PCI Express x8 slot. The Myrinet switch is a single 21U enclosure with 512 copper ports using DensisShield ™(mini-CX4) connectors. This is a 3-stage full Clos topology, using 32-port crossbars: each of the 32 switch line cards provide 16 connections to the nodes, and 16 links to the spine which is composed of 16 crossbars, as shown in Figure 1.

There are 16 possible routes between each pair of nodes, a total of 8192 routes stored in the firmware of each NIC. The nodes runs Linux 2.6.9 and MX 1.2.6 with a few firmware additions to dynamically select the type of route dispersion and the dispersion parameters at runtime.

5.2 Benchmarks

The benchmark we use is measuring the bisection bandwidth for different random bisection communication patterns. We generate random communication patterns using the Mersenne Twister [11] pseudo-random number generator. A pattern for P endpoints (nodes) is generated by the following algorithm:

- randomly split the network of size P into two equally sized groups A and B
- randomly create $P/2$ pairs such as that every pair consists of a node from A and B
- guarantee that no node is in more than a single pair (has more than one partner in the other group)

Thus, a large enough number of measurements with random bisect patterns represents the effective bisection bandwidth of the network.

We used the Message Passing Interface (MPI) implementation MPICH-MX 1.2.7..5 to run the benchmarks. After generating a pattern ω, every node begins to receive and send data from/to its designated peer in a non-blocking way (cf. ping-ping in the Pallas/IMB benchmarks [12]). In parallel, this pattern is known as *pair-wise exchange* and is sometimes used by implementations of MPI_Alltoall on large messages. Each node $p\,(\forall p = 1, \ldots, P)$ takes the time t_p^ω that is needed to exchange 50 messages of size $s = 1\,MiB$ with its peer.

The benchmark keeps record of all the achieved bandwidths $b_p^\omega = s/t_p^\omega$ and computes the average bisection bandwidth of every pattern ω:

$$b_{avg}^\omega = \frac{\sum_1^p b_p^\omega}{p}$$

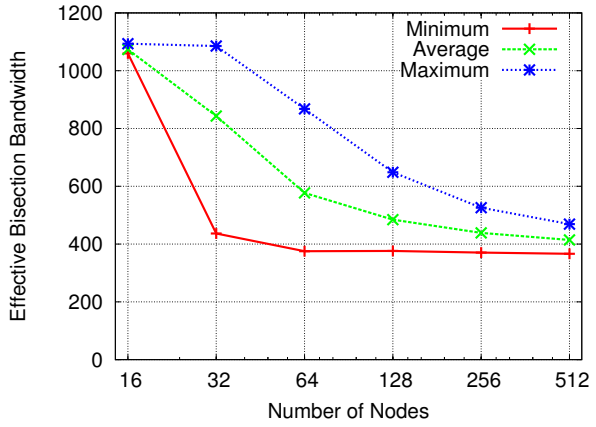

Figure 2. Static routing

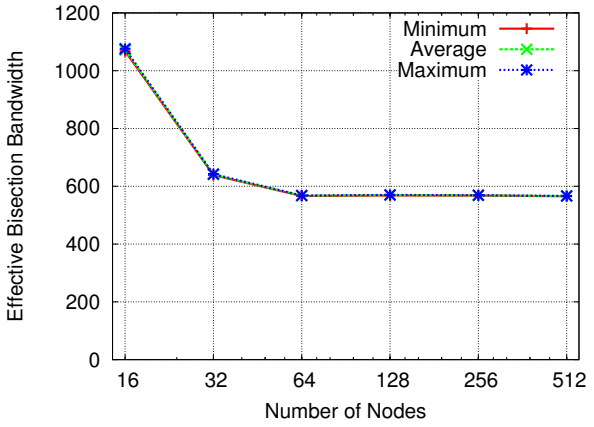

Figure 3. Randomized oblivious routing

.

The benchmark then reports the minimum, maximum and average bandwidth achieved across all different patterns. We measured 5000 different patterns for each configuration. The following section shows the results for different node counts and configurations.

5.3 Results

Even though the link rate of Myri-10G is 10+10 Gb/s (1.25+1.25 GB/s), the benchmark sends only one message at a time in each direction, similar to a Ping-Pong benchmark. At a size of 1 MiB, the nominal Ping-Pong bandwidth between two nodes on the cluster is $1090\,MiB/s$. This is our baseline performance for the following results, representing measurements from 16 to 512 nodes.

Figure 2 shows a plot of the results for static routing, i.e. no route changes. This shows that the average effective bisection bandwidth decreases to $414.3\,MiB/s$ at 512 nodes, which is 38% of the full bisection bandwidth. Similar results have been shown in [6] for statically-routed InfiniBand systems.

The randomized oblivious routing, introduced in Section 4.1, chooses a different random route from a set of 16 routes for every packet. Thus, it is comparable to the solution discussed by Vishnu et al. in [19]. However, Vishnu et al. only discuss a particular worst-case pattern. Our measurements show that the randomized oblivious routing improves the performance of the worst case compared to static routing. However, the average case is performing as bad as the worst case for this scheme, which is even worse than static routing for some node counts. Thus, we conclude that the randomized oblivious routing is able to improve the worst case slightly but also performs worse for some configurations. The stagnation around $566.6\,MiB/s$ also supports the results from Karol et al. in [7] who predicted

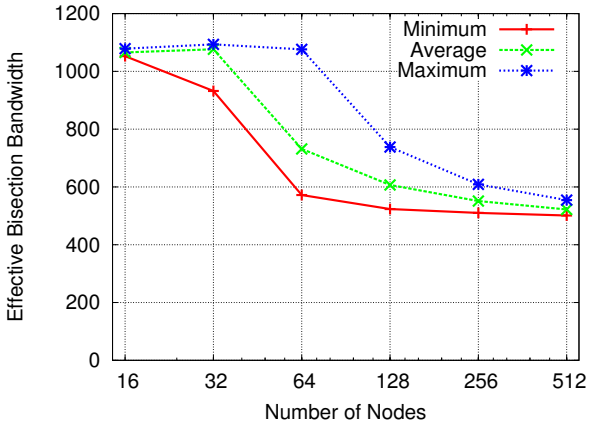

Figure 4. Adaptive routing

a upper bound of 58% ($623.5\,MiB/s$ in our case). However, as packets are systematically dispersed on all possible paths, the average load on each link of the fabric is constant and this load does not depends on the specific communication pattern. Therefore, the gap between the worst, average and best cases for the randomized oblivious scheme is very narrow. This shows great fairness and determinism, which are valuable characteristics in distributed systems.

Figure 4 shows the adaptive routing scheme which changes routes every time a contention is sensed as described in Section 4.2. We used a dispersion threshold of 4 blocked packets and a dispersion probability of 50%. This scheme achieves a performance improvement relative to the randomized oblivious routing up to 128 nodes. As the network becomes more congested with growing node count, the scheme degenerates towards randomized oblivious routing. However, the dispersion threshold and probability (4, 50%) used in the adaptive scheme limit the frequency of route changes. Thus, the performance is slightly lower for

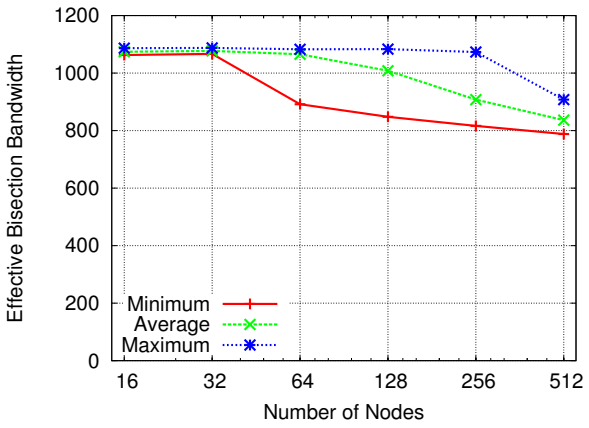

Figure 5. Probing adaptive routing

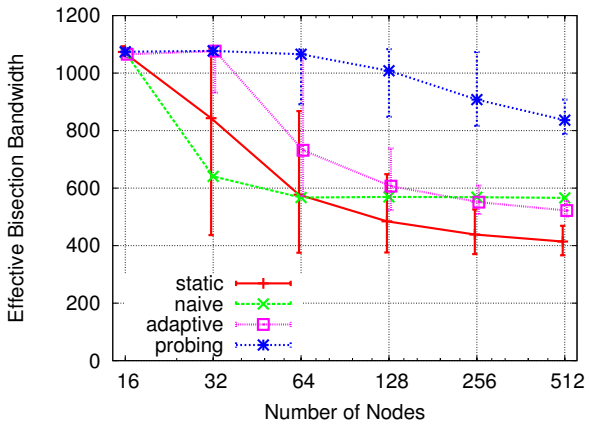

Figure 6. Comparison of all schemes

the large node counts.

The probing adaptive routing, shown in Figure 5 and described in Section 4.3 delivers the best results for all node counts. In the best case, the full bisection bandwidth is reached up to 256 nodes. In the worst case, a minimum of 72% of full bisection is sustained. On average, this scheme delivers respectively 92% and 77% of full bisection at respectively 128 and 512 nodes. We did a full parameter study for threshold and probability to get the best results, using the discrete values 0, 4, 8 and 16 for the threshold, and 50% and 100% for the probability. The best combination overall is (16, 100%) and the corresponding results were used in Figure 5. However, for small configurations (up to 128 nodes), the choice of parameters was not significant, i.e. all results were in a range of 10% of each other for most parameters. Even for large node counts, the variations between all combinations were limited.

Figure 6 shows all results plotted in a single diagram. The error bars indicate the minimum and maximum effective bisection bandwidth. It can clearly be seen that the probing adaptive routing performs significantly better than all other schemes.

5.4 Bridge pattern

One particularly interesting communication pattern that is not subject to the local optimum convergence problem is the *Bridge* pattern. In this pattern, all nodes connected to the same first crossbar of the Clos network (which corresponds to a single line card in the Myrinet switch) communicate with all nodes connected to a different crossbar. For example, on a Clos of n-port crossbars, nodes $[0 : n-1]$ are paired with nodes $[n : 2n-1]$, nodes $[2n : 3n-1]$ are paired with nodes $[3n : 4n-1]$ and so on.

In this pattern, all of the contenders for the possible routes on the way up and down on the Clos network belong

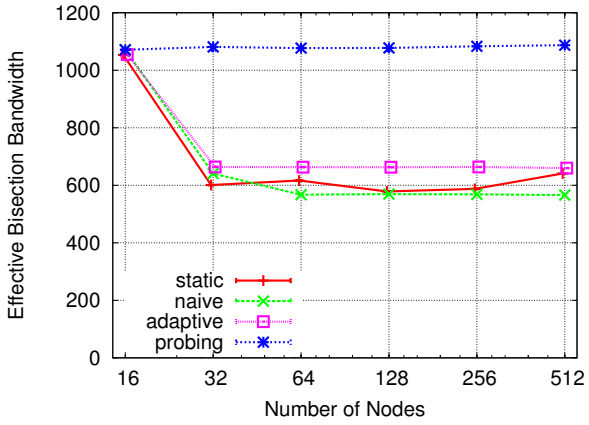

Figure 7. Comparison for the bridge pattern

to the same groups. Figure 7 shows the effective bisection bandwidth of the various routing schemes on different node counts. The probing adaptive scheme delivers over 98% of the full bisection bandwidth at any size, while all the other methods average around 56%.

This pattern is important because it could be used in the scheduling of topology-aware collective communications such as MPI_Alltoall. If the MPI implementation would know which nodes belong to the same first-level crossbars, it could use this knowledge to schedule point-to-point communications that follow a bridge pattern, leveraging the full bisection bandwidth provided by the interconnect.

6 Conclusions and Future Work

We have shown in this paper that the effective bisection bandwidth on full bisection interconnects greatly depends on the routing strategy. We have implemented and evaluated several dispersive routing schemes on real hard-

ware and we have shown that the probing adaptive routing yields the best effective bisection bandwidth, almost twice as much as static routing.

However, dispersive routing requires the interconnect to support source routing and tolerate out-of-order packets. To the best of our knowledge, these two requirements are fulfilled by Myrinet and Quadrics; while InfiniBand relies on strong packet order. Ethernet does not currently support source routing and the performance of protocols like TCP requires strong order as well.

A large body of work remains on this topic. Beyond evaluating the effect of probing adaptive routing on real-world applications, proper attention should be paid to the convergence problem: how fast does the routing adapt to a new contention situation? Could it be more efficient to gather global knowledge? At what cost? Furthermore, designing topology-aware collective communications offer the possibility of greatly improving the efficiency and the scaling of some of the most critical communication primitives.

Acknowledgments

Computation for the work described in this paper was supported by the University of Southern California Center for High-Performance Computing and Communications (www.usc.edu/hpcc). The authors wish to particularly acknowledge Garrick Staples, HPCC System Administrator, for his continuous support.

References

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: A gigabit-per-second local area network. *IEEE Micro*, 15(1):29–36, 1995.

[2] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. *SIGARCH Comput. Archit. News*, 21(2):351–360, 1993.

[3] J. Bruck, C.-T. Ho, E. Upfal, S. Kipnis, and D. Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Trans. Parallel Distrib. Syst.*, 8(11):1143–1156, 1997.

[4] C. Clos. A study of non-blocking switching networks. *Bell System Technology Journal*, 32:406–424, 1953.

[5] Z. Ding, R. R. Hoare, A. K. Jones, and R. Melhem. Level-wise scheduling algorithm for fat tree interconnection networks. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 96, Tampa, Florida, 2006. ACM.

[6] T. Hoefler, T. Schneider, and A. Lumsdaine. Multistage switches are not crossbars: Effects of static routing in high-performance networks. Submitted to the IEEE Cluster 2008 Conference.

[7] M. J. Karol, M. G. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, 35:1347–1356, 1987.

[8] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong-Chan, S.-W. Yang, and R. Zak. The network architecture of the Connection Machine CM-5. *Journal of Parallel and Distributed Computing*, 33(2):145–158, 1996.

[9] X. Lin, Y. Chung, and T. Huang. A multiple lid routing scheme for fat-tree-based infiniband networks. In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS04)*, page 11a, Sana Fe, NM, 04 2004.

[10] J. C. Martínez, J. Flich, A. Robles, P. López, and J. Duato. Supporting fully adaptive routing in infiniband networks. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 44.1, Washington, DC, USA, 2003. IEEE Computer Society.

[11] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[12] Pallas GmbH. Pallas MPI Benchmarks - PMB, Part MPI-1. Technical report, 2000.

[13] F. Petrini, W. chun Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The quadrics network (qsnet): High-performance clustering technology. In *HOTI '01: Proceedings of the The Ninth Symposium on High Performance Interconnects (HOTI '01)*, page 125, Washington, DC, USA, 2001. IEEE Computer Society.

[14] F. Petrini and M. Vanneschi. K-ary n-trees: High performance networks for massively parallel architectures. Technical report, 1995.

[15] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato. Solving hot spot contention using infiniband architecture congestion control. In *Proceedings HP-IPC 2005*, Research Triangle Park, NC, 6 2005.

[16] M. D. Schroeder, A. Birell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker. Autonet: A high-speed, self-configuring local area network using pointto -point links. *IEEE Journal on Selected Areas in Communications*, 9(8), 10 1991.

[17] W. Systems. 10ge fabric delivers consistent high performance for computing clusters at sandia national labs. Technical report, 2007.

[18] The InfiniBand Trade Association. *Infiniband Architecture Specification Volume 1, Release 1.2*. InfiniBand Trade Association, 2003.

[19] A. Vishnu, M. Koop, A. Moody, A. R. Mamidala, S. Narravula, and D. K. Panda. Hot-spot avoidance with multi-pathing over infiniband: An mpi perspective. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 479–486, Washington, DC, USA, 2007. IEEE Computer Society.