LAPSES: A Recipe for High Performance Adaptive Router Design *

Aniruddha S. Vaidya Anand Sivasubramaniam Chita R. Das
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {vaidya,anand,das}@cse.psu.edu

Abstract

Earlier research has shown that adaptive routing can help in improving network performance. However, it has not received adequate attention in commercial routers mainly due to the additional hardware complexity, and the perceived cost and performance degradation that may result from this complexity. These concerns can be mitigated if one can design a cost-effective router that can support adaptive routing. This paper proposes a three step recipe — Look-Ahead routing, intelligent Path Selection, and an Economic Storage implementation, called the LAPSES approach — for cost-effective high performance pipelined adaptive router design.

The first step, look-ahead routing, reduces a pipeline stage in the router by making table lookup and arbitration concurrent. Next, three new traffic-sensitive path selection heuristics (LRU, LFU and MAX-CREDIT) are proposed to select one of the available alternate paths. Finally, two techniques for reducing routing table size of the adaptive router are presented. These are called meta-table routing and economical storage. The proposed economical storage needs a routing table with only 9 and 27 entries for two and three dimensional meshes, respectively. All these design ideas are evaluated on a (16×16) mesh network via simulation. A fully adaptive algorithm and various traffic patterns are used to examine the performance benefits. Performance results show that the look-ahead design as well as the path selection heuristics boost network performance, while the economical storage approach turns out to be an ideal choice in comparison to full-table and meta-table options. We believe the router resulting from these three design enhancements can make adaptive routing a viable choice for interconnects.

1 Introduction

Multiprocessor interconnection network designers have always strived to design scalable interconnects with low latency and high throughput. While low message latency helps to reduce the communication overhead in any parallel application, it is particularly beneficial for short messages encountered in shared memory systems. High throughput/bandwidth is essential for bulk data transfer that could occur due to remote page migration, I/O traffic or other data intensive applications. Since low latency in general translates to high throughput, a network should provide minimal latency over the entire, anticipated workload on the system. Moreover, network architectures, designed originally for multiprocessors, are increasingly being accepted in demanding application domains such as web servers and multimedia servers [16]. A more general environment such as a system area network is likely to experience high and fluctuating workloads. Enhancing network performance is thus imperative to fuel continued improvements not only in parallel architectures but also on many other fronts. The building block of a network being its router or switch fabric, the router design should have the necessary features to aid in building a high performance interconnect, and is the focus of this paper.

Network research over the years has converged towards wormhole switching mechanism and virtual channel flow control to provide improved performance in scalable direct networks. These research ideas have manifested into many commercial switch designs today [12, 20, 21, 13, 19, 3, 26, 18, 1]. Table 1 shows a non-exhaustive list of commercial routers and the features they support. A third component, in addition to switching and flow control, that has a significant impact on network performance is the routing algorithm. Several research studies have shown performance benefits of various adaptive routing schemes compared to oblivious routing [15, 14, 9, 17, 7, 2, 5]. Theoretically, it is known that routing adaptivity is a desirable feature since it can lower average message latency at moderate to high load. Further, the ability to use alternate paths improves fault-tolerance properties of the network. In spite of these advantages, very few commercial router designs have adopted this idea (T3E router [21], Servernet-II [13] and Transputer/C-104 [18] switch to a limited extent).

Complexity and cost are the main reasons attributed to the limited commercial adoption of adaptive routing. We believe that if we can provide a cost-effective solution to implementing adaptive routing in the context of the current router architectures, then adaptive routers will have a better commercial viability. Since adaptivity helps in boosting network performance, we can see its benefit translated to fine grain parallel applications and to emerging applications that are likely to inject high network load.

The two main cost factors associated with providing adaptivity in a router are the number of VCs required and the implementation of the algorithm itself. It was argued in [4] that addition of each VC slows down the router clock by 25–30%. This is only true for a non-pipelined design. Current commercial routers are being designed with increasingly large transistor and area budgets. Also, faster design cycles due to ASIC design approaches and general-purpose applicability of routers (for use in arbitrary topologies and system-area interconnects) have lead to the adoption of programmable routing tables, virtual channels, and increased fault-tolerance features using pipelined router designs [21, 12, 13]. An adaptive router should, therefore, adhere to a pipelined design and should exploit all available facilities such as VCs and table based routing. VCs are thus not considered an additional expense.

Even though these key enabling technology trends set the broad guideline for designing routers to support adaptive routing, there are still a few challenges that need to be addressed for making adaptive routing commercially viable and popular:

- Additional work required for adaptive routing can potentially increase the number of pipe-stages in the router and/or result in a slower router clock cycle as compared to a deterministic router. In particular, the serially dependent operations of table-lookup and path-selection cum arbitration (Fig. 1 shows the pipelined stages in a router.) are a key part of the critical path through the router. Decoupling of table-lookup and path-selection cum arbitration functions through a technique called *look-ahead routing* has the potential to reduce

*This research is supported in part by NSF grant MIPS-9634197, NSF Career Award MIPS-9701475, and equipment grants from NSF and IBM.

Router	R-Tbl	Design	Max Nodes	Ports	VCs	Port Type	Routing
SGI SPIDER	Y	ASIC	512	6	4	P	Det
Cray T3D	Y	ASIC	2K	7	4	P	Det
Cray T3E	Y	ASIC	2176	7	5	P	Adpt
Tandem Servernet-II	Y	ASIC	1M	12	No	P	Lim. Adpt
Sun S3.mp	Y	ASIC	1K	6	4	2P + 4S	Adpt
Intel Cavallino	N	Custom	$>$ 4K	6	4	P	Det
HAL Mercury	N	Custom	64	6	3	P	Det
Inmos C-104	Y	Custom	Any	32	Any	S	Lim. Adpt
Myricom Myrinet	N	Custom	Any	8/16	No	P	Det

Table 1: A non-exhaustive list of state-of-the-art commercial wormhole and virtual-cut through routers.

pipeline-latency in adaptive routers.

- When multiple paths are available for routing to a given destination, a unique path has to be selected for the next route [10]. Selection of a good path among the available alternatives is important for improved performance. This issue does not arise in deterministic routers.
- Adhering to a table-based router design, adaptive routing requires multiple path choices to be stored in routing tables, thus increasing table storage cost. Offsetting this cost, especially in the design of routers for large scalable interconnects, may become important.

We investigate these three issues in this paper and propose the design of high-performance, low-latency, table-based routers using the *LAPSES* approach — look-ahead (LA) routing, good path-selection (PS) and economical storage (ES). Look-ahead routing decouples the table lookup and path selection/arbitration stages of the pipelined design and uses the current routing table entry for the next routing step. In essence, by making lookup and arbitration concurrent, we reduce one pipe stage and thus the overall delay. This technique has been implemented in the SGI SPIDER, which uses oblivious routing. We extend the look-ahead technique to adaptive routers by providing the valid path options in the header flit. Next, we propose three path selection strategies, called *least frequently used* (LFU), *least recently used* (LRU) and *maximum credit* (MAX-CREDIT) for selecting one of the available paths provided by the adaptive routing algorithm. Finally, we discuss various options to implement adaptive routing using table-lookup. Since multiple table entries are required for maintaining alternate routing options, increased storage cost is another concern here. We study three different designs - (i) full table implementation, where the table size is proportional to the number of nodes in the network, (ii) a meta-table design that partitions the network into groups (clusters) and uses hierarchical lookup for inter cluster and intra cluster routing and (iii) an elegant economic routing table implementation, which needs only 3^n entries for an n-dimensional interconnect.

All these design options are examined by simulating a (16×16) 2-D mesh network using these routers. As an example, Duato's fully adaptive algorithm [9] is used in this study. We use various synthetic traffic patterns to evaluate the impact on average network latency. The results indicate that the look-ahead feature reduces the average latency at low load while the routing flexibility becomes advantageous at high load, thereby making look-ahead adaptive router a good choice across the entire spectrum. The effect of look-ahead is significant for short messages as the latency could reduce by as much as 15%. The LRU, LFU and MAX-CREDIT path selection strategies outperform previously proposed static-XY [10] and MIN-MUX [9] policies for all of the nonuniform traffic patterns. The LRU and MAX-CREDIT policies seem to be better choices because of their low cost of implementation. Finally, we observe that meta-table implementation is not an appropriate choice for 2-D mesh networks, although it is cost-effective compared to full-table design. The proposed scheme (economical storage) that needs only 3^n entry-tables can provide identical performance as full-table routing, making it attractive for adaptive routers. This scheme needs only 9 and 27 routing table entries for 2-D and 3-D networks, which are the common topologies. These three techniques used in conjunction with todays enabling router design technology can make high-performance adaptive routers a commercially viable and successful design choice.

The rest of the paper is organized as follows. In section 2, a pipelined router model, called PROUD, and the experimental setup are presented. The look-ahead routing scheme is discussed in section 3. The next section introduces the path selection policies and analyzes their performance. Section 5 presents the table implementation details, followed by the concluding remarks in in Section 6.

2 Preliminaries

2.1 Router Architecture

Since this paper is on the design and analysis of router architectures, we begin with a logical description of its building blocks. A typical wormhole router has synchronization and hand-shaking logic at the ports, input/output flit buffers, flit-decoders as well as a crossbar and its control unit (consisting of a routing decision block and arbiter). Message flits enter the router at an input port and eventually exit the router at an appropriate output port as determined by the crossbar setting. The routing information contained in the header flit of a message is used by the routing decision block to determine the appropriate crossbar setting. The crossbar arbitration unit arbitrates between messages contending for the same crossbar output port. Flits of a message that temporarily cannot make progress because of currently unavailable network resources, are held in the flit buffers.

To support VCs, a VC de-multiplexor unit precedes the input flit buffers and a VC multiplexor precedes the output port of the router [6, 24]. In an adaptive router, the routing decision block may have a choice of crossbar output ports to route a message. A *path selection function* is required in the routing decision block to select one of multiple valid output ports. To support flexibility in network designs and routing algorithms, some routers use table-lookup routing (see Table 1). The routing decision block in these routers is implemented as a programmable lookup-table. The table is typically indexed by the destination node address and the corresponding table entry determines the crossbar output port to route the message on. By providing multiple entries in the routing table for every index, the routing table can provide support for adaptive routing. The routing table entries are configured based on the routing algorithm to be used.

2.2 Router Models and Experimental Setup

Router Models: The cumulative delay through the router that is experienced by a message flit is determined by the individual delays of the functional units within the router. However, in order to increase the throughput, modern routers use a pipelined design [12, 21, 8].

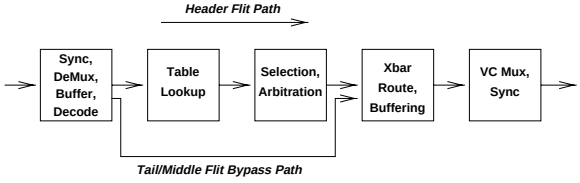


Figure 1: Pipeline stages in the PROUD router

The pipelined router models used in this study are called PROUD (for Pipelined ROUter Design) and LA-PROUD (for PROUD with Look-Ahead). The five-stage pipeline for PROUD is given in Fig. 1. In this study, we use the PROUD model to study the performance of deterministic and adaptive routers without lookahead. Note that in deterministic routers, the selection cum arbitration stage simply reduces to arbitration (as no path-selection is required). However, path-selection does not contribute significantly to the delay of this stage and hence we assume identical delays for deterministic and adaptive routers in our analysis.

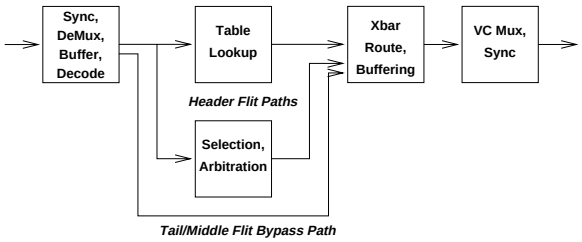


Figure 2: Pipeline stages in the LA-PROUD router

In the PROUD delay model, the routing-table lookup stage and the path-selection cum arbitration stage are serially dependent. The LA-PROUD delay model is a four-stage pipe with table-lookup functions decoupled from path-selection cum arbitration achieved via lookahead routing. This model is shown in Fig 2. Details of look-ahead routing are discussed in Section 3.

Both the PROUD and LA-PROUD models are similar to the pipelined architectures of the SPIDER or T3E routers. Conceptually, a router can be considered as a set of parallel PROUD/LA-PROUD pipes equal to the product of the number of physical input/output ports and the number of VCs per port. Contention for resources between the parallel pipes can occur only in the crossbar arbitration and VC multiplexing stages. In a contention-free environment, the key functional unit delays which determine the router cycle time are the table-lookup delay and the arbitration delay [12]. Our entire study here is confined to improving the design in these two critical stages of the pipelined router to support adaptivity.

Experimental setup: In this study we use the PROUD network simulator to simulate a 256 node two-dimensional (16×16) mesh interconnection network. Each router is modeled as a 5 port bi-directional switch. Four of the ports are connected to up to four neighboring nodes in the mesh and the fifth port is used to communicate to the local processing node network interface. We assume 4 VCs per physical channel as is available in most recent routers. Each stage of the PROUD (or LA-PROUD) is assumed to take unit cycle time under no resource contention. In addition, unit cycle delay is assumed for traversing a link between two connected routers.

Parameter	Value
Mesh Network Size	256 node (16×16)
Message Length	20 flits (unless specified)
Inter-arrival time	Exponential distrib.
Traffic	Uniform, Transpose, Shuffle, Bit-Reversal
In/Out Buffer Size	20 flits
VCs per PC	4
Network Cycle Time	1 unit
Router Latency (contention-free)	5 units (PROUD) 4 units (LA-PROUD)
Link Delay	1 unit

Table 2: Simulation parameters used in performance study

All our performance studies are for a constant message length of 20 flits (unless otherwise indicated). Messages are injected with exponential inter-arrival times for 4 different traffic patterns (uniform, transpose, bit-reversal and perfect-shuffle traffic). These traffic patterns are consistent with standard definitions for synthetic traffic patterns used in interconnection network studies [11]. We present performance results as the average network latency versus normalized load. Normalized load is defined as the ratio of the message injection rate per-cycle to that injection rate, required to reach the bisection bandwidth of the network under node-uniform traffic [11]. Results are only presented for loads leading up to network saturation. Simulation data was collected by injecting 10000 warm-up messages after which statistics was collected over 400000 message injections. A summary of the simulation parameters used in this study is given in Table 2.

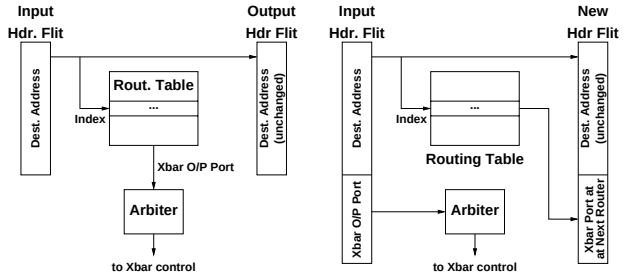
2.3 Adaptive Routing Algorithms

Several adaptive routing algorithms have been proposed for direct networks [15, 14, 9, 22, 2, 5]. These algorithms vary in terms of their performance and hardware (VC) requirements. Since we are interested in a cost-effective implementation, we use a fully adaptive routing algorithm that requires the minimum number of VCs. The algorithms that meet this criteria are those in [14, 2, 9, 22] and they require 2 VCs per physical channel for deadlock-free adaptive routing in a 2-D mesh. In this paper, we use Duato's fully adaptive algorithm [9] for performance analyses, and these discussions are valid for other fully adaptive algorithms as well since they exhibit similar behavior.

3 Look Ahead Routing

3.1 Basic Look Ahead Routing

It was seen in Section 2 that table lookup and path selection cum arbitration operations are serially dependent. First, the table-lookup determines the output port of the router crossbar to be used to route a given message towards its destination. Then, upon determination of the desired output port, the message arbitrates for that port. Fig. 3 (a) illustrates this serial dependency. Also note that, only the destination address is required to be present in the message header-flit for the purposes of routing, and the same header-flit can be used by the message without modification at the next router.



(a) No Look-Ahead **(b)** Look-Ahead

Figure 3: Input header flit format, usage and new header generation for deterministic routers (a) without, and (b) with lookahead.

Now, if the crossbar output port can be known in advance, then arbitration can be performed directly by eliminating the need for table lookup for routing at the current router. Table lookup can then be performed concurrently with arbitration, for deciding the output port to be taken at the next router along the path to the destination. We refer to such a scheme as *look-ahead* routing. In particular, when such a scheme is used in a deterministic router, we refer to it as *deterministic look-ahead* routing. This scheme is illustrated in Fig. 3 (b). Note that look-ahead routing requires the crossbar output port to be used at the current router to be pre-specified in the header-flit of the message (thus increasing header size), and a partial modification of the header-flit at every router along the path. Deterministic look-ahead routing is used in the SGI SPIDER [12][1]. We extend the concept of look-ahead routing to adaptive routers.

[1] This technique is called "table-lookup pipelining" in SGI SPIDER.

3.2 Adaptive Look-Ahead Routing

Adaptive routing implies the possibility of multiple path choices being available at a given router to route a message towards its destination. Routing table entries in an adaptive router thus need to store multiple valid output ports per destination. Further, for enabling look-ahead routing, look-up tables need to contain allowed output-port information for routers along multiple candidate paths out of the current router. This further increases the storage requirement for adaptive look-ahead routing. Note that when multiple candidate paths are available, a unique path has to be picked from amongst the choices. This is called *path selection*. Path-selection and arbitration are both required before a message may be routed through the crossbar. We illustrate the working of adaptive look-ahead routing through the example of a 2-D mesh interconnect.

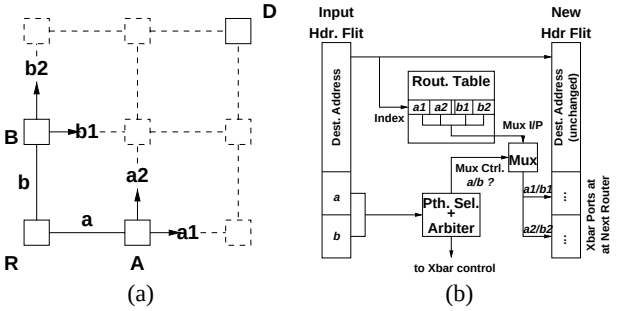


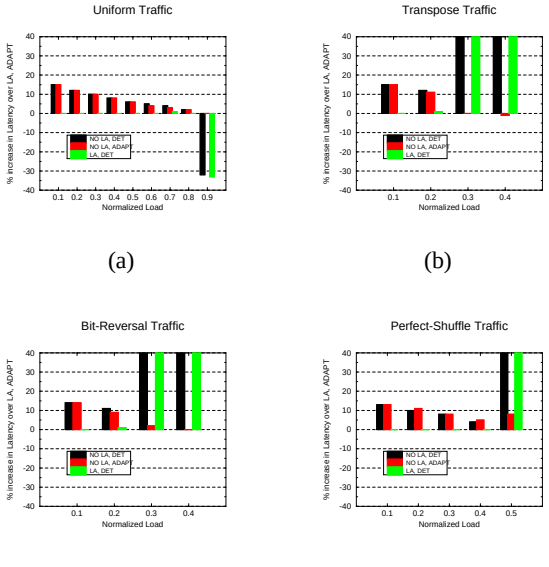
Figure 4: (a) Multiple paths to a destination node in a mesh network. (b) Input header flit format, usage and new header generation.

Consider the intermediate router R of Fig. 4 (a). If a message at R needs to be routed toward D, then ports a and b would both take the message in productive directions to routers A and B, respectively. Ports $a1, a2$ and $b1, b2$ at routers A and B, respectively, would take the message further in productive directions. The table entries at router R to route to node D for adaptive lookahead would then have entries for ports $a1$ and $a2$ of node A, which is along port a of R, and entries for ports $b1$ and $b2$ of node B, which is along port b of R. As soon as path-selection and arbitration have been performed at R, it is known which of the port a or b will be uniquely used to route towards destination D. At this point, part of the lookahead table lookup information corresponding to the unused port at R (either $a1, a2$ or $b1, b2$) can be discarded. The rest is used to construct a new header flit containing information for path-selection and arbitration at the next node along the route to D.

The header flit interpretation, table-lookup and new-header flit generation in adaptive look-ahead routers, corresponding to the above example, is shown in Fig. 4 (b). It should be observed that the new header generation based on the outcome of path selection (seen as the Mux-Control line and Mux unit in Fig. 4 (b)) is not on the critical path for the arbitration to proceed and hence does not increase the delay of the path-selection cum arbitration stage. New header generation can be performed concurrently with the header being routed through the crossbar and into the crossbar output buffer.

3.3 Performance of Adaptive Lookahead Routing

Compared to deterministic routing, performance improvement with adaptive look-ahead routers stems from two aspects of their design, viz. their ability to adaptively use multiple candidate paths and to reduce router latency by look-ahead routing. In this section, we quantify the performance improvement due to the above factors using a consistent set of delay models (PROUD and LA-PROUD) and compare the performance of four routers for a 2-D mesh interconnection network — deterministic routers with and without look-ahead, and adaptive routers with and without look-ahead. (As stated in Section 2, the routing table implements Duato's fully adaptive scheme.) Next, we evaluate the impact of message length on the latency of adaptive lookahead routers.



Load	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Uniform	69.2	74.0	80.5	87.2	97.5	111.0	130.4	168.6	432.8
Transpose	74.5	87.6	294.6	715.6					
Bit-Rev.	76.1	93.6	411.2	1155.3					
Shuffle	60.1	66.3	76.6	98.3	608.1				

Figure 5: Comparison of Router performance with or without Look-Ahead and with or without adaptive routing for four traffic patterns on a (16×16) mesh. A positive/negative bar in the figures indicates that a given scheme has higher/lower latency than the adaptive router with lookahead. The table provides the actual latency values for the adaptive router with lookahead for each of the traffic patterns.

Figure 5 shows the latency comparison of the four router architectures (NO LA DET, LA DET, NO LA ADAPT, and LA ADAPT) as a function of normalized load on the network. The results are given for a message length of 20 flits and static path selection strategy (explained in the next section) used for simulating the adaptive algorithm. The latency results are normalized with respect to the LA ADAPT router scheme. In all the four figures for the four traffic patterns, it is seen that the LA ADAPT router outperforms both the no look ahead routers (NO LA DET and NO LA ADAPT) by as much as 12-15% when the load is low. The LA DET performs almost identical as the LA ADAPT scheme for light load and hence the latency difference is negligible. For uniform traffic in Fig. 5(a), both the deterministic implementations perform better than the adaptive implementations for high load re-confirming the results that routing adaptivity does not help uniform traffic. The results for the three non-uniform traffic patterns indicate that the adaptive algorithms with or without look-ahead show significant performance improvements against deterministic schemes at high load. The benefits of look-ahead are swamped by the relatively larger benefits of adaptive routing at high loads.

Mesg. Len	Look Ahead	No Look Ahead	% Improv.
5	51.9	63.4	18.0
10	58.9	69.6	15.4
20	74.0	83.6	11.5
50	120.2	128.6	6.5

Table 3: Impact of message length (Uniform traffic, normalized network load of 0.2.)

Effectiveness of the lookahead with respect to varying message length is given in Table 3. The results are for adaptive routers with look-ahead and no look-ahead features. As expected, short messages benefit the most due to look-ahead and the relative improvement reduces with an increase in message length.

In summary, look-ahead helps in reducing message latency at low load while adaptivity takes over for reducing message latency at high load. Thus, a look-ahead, adaptive router is the best choice since it can help over the entire range of workload. In addition, short messages see the maximum reduction in latency by saving one pipe line stage in the router architecture.

4 Path Selection Heuristics

4.1 Path Selection

An adaptive router when presented with multiple path choices to route to a given destination, must select a unique and currently available path from amongst the multiple candidates. The criterion for selection is called the path selection function [10]. Researchers and router designers have commonly used dimension-order selection [10, 2], random selection [17], and first-available-free-path selection [13] for their simplicity. These criteria are static in the sense that they do not make use of current network conditions to select a path, which is likely to experience the lowest contention. In this paper, the dimension-order selection is referred to as STATIC-XY since it prefers the X-dimension first (in the case of a 2D-mesh).

By using port usage history, dynamic or traffic sensitive selection criteria can be used with an attempt to minimize path contention in the network, thereby improving routing performance. We propose three dynamic path selection heuristics (PSHs), called LRU, LFU and MAX-CREDIT, and compare them against STATIC-XY and another dynamic scheme referred to as MIN-MUX in this paper. MIN-MUX uses the physical channel with the minimum degree of VC-multiplexing (For details about the MIN-MUX PSH, see [9]).

Least Frequently Used (LFU) PSH: This PSH selects the output port (among the candidates) with the lowest usage count until that time. It works on the premise that if link utilizations are balanced, it will result in improved network performance. Implementing this PSH would require maintaining a counter for each crossbar output port, incrementing it whenever the corresponding port is used and selecting the port with the lowest counter value.

Least Recently Used (LRU) PSH: Typically, recent history is a better indicator of congestion information than cumulative history. LRU tries to route a message through a candidate crossbar output port that was used farthest in the past. One way to implement this scheme would be to use counters for each output port like in LFU. However, these "age" counters would get incremented every time the output port remains unused and reset when the corresponding crossbar port is used. The oldest port amongst the candidates would be selected.

MAX-CREDIT PSH: The LFU PSH keeps track of all past usage of a channel. However, it does not accurately reflect the current usage of the channel. The MAX-CREDIT PSH tries to remedy this situation.

A majority of wormhole routers use credit-based flow control, where routers credit their neighboring routers with the amount of free buffer space available for that channel with them. These credits are decremented by the upstream router as flits are sent along the link and incremented as acknowledgments are received. Thus, a large credit value for a link is indicative of possibly low congestion at the downstream router. The MAX-CREDIT PSH hence picks the channel with maximum available credits from amongst the available candidate channels. Implementation details of these schemes can be found in [23].

4.2 Performance of Path Selection Heuristics

Figure 6 depicts the average message latency as a function of network load with five PSHs (static-XY, MIN-MUX, LFU, LRU and MAX-CREDIT). The results are for a (16×16) network with four different traffic patterns.

As expected, the static path selection performs the best for uniform traffic, although MIN-MUX, LRU and MAX-CREDIT heuristics are comparable except at very high load. For the rest of the three traffic patterns, the four load sensitive selection schemes perform much better than the static path selection. In particular, the

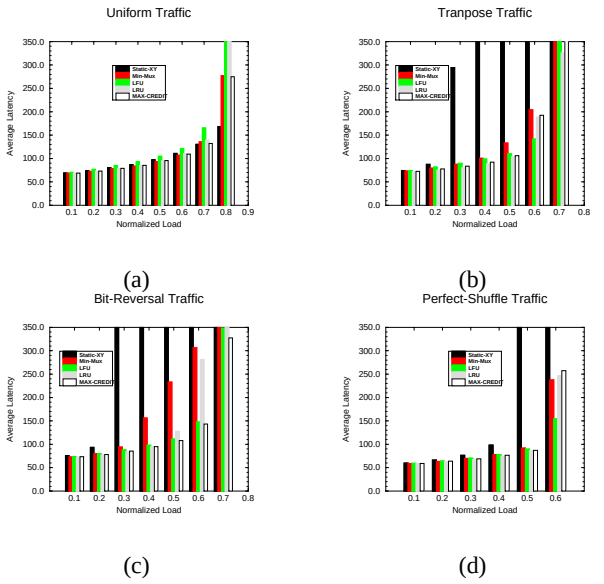


Figure 6: Performance of path-selection heuristics

LRU, LFU and the MAX-CREDIT schemes are the best performers under medium to high load before saturation. The LFU policy has the minimum latency for bit-reversal traffic. In most cases, MAX-CREDIT performance lies in between LFU and LRU. While all these three policies exhibit similar performance, the LRU and MAX-CREDIT implementations need smaller counters.

These results indicate that adaptive router designs should include traffic-sensitive path selection heuristics to utilize the links more efficiently. It is possible to enhance the performance further by implementing the same adaptive algorithm with better, but not necessarily expensive hardware support.

5 Reducing Table Storage Overhead

Table-based routing schemes have become popular due to factors such as the ability to reprogram routers for changes in topology, improving performance (by using more efficient routing algorithms), small and constant table lookup cost (independent of routing algorithm complexity) and fault-tolerance capability. The preferred method of implementing routing tables has been to use complete routing tables, where a distinct routing table entry is available for every destination node in the network (up to the maximum number of nodes to be supported). Such an implementation scheme is referred to as *full-table routing*.

Full-table routing has the flexibility of supporting arbitrary network topologies limited only by the number of ports in the router. This approach is used in the Cray T3D and T3E routers [20, 21], as well as the Sun S3.mp router [19]. However, full-table routing by its nature has storage overhead proportional to the maximum number of nodes in the network and thus limits its scalability to large network sizes. A large table RAM size may also lead to slower lookup times.

Support for adaptive routing require multiple entries per table index, increasing storage requirements over deterministic routers. Lookahead routing further increases these requirements. This motivates the need for economical storage solutions for scalable high performance routers. In this section, we present two-known schemes for reducing routing table sizes, discuss their applicability for adaptive routing and propose an innovative approach for drastically reducing storage requirements for n-dimensional mesh-like networks.

5.1 Earlier Schemes for Reducing Routing-Table Size

5.1.1 Hierarchical or Meta-table Routing

In full-table routing, destination addresses are used to index into a flat routing table-structure. Conceptually, hierarchical or meta-table routing differs by maintaining two or more levels of routing tables. Interconnection network nodes are partitioned logically into clusters such that all nodes within a cluster have the same *cluster id* and distinct *sub-cluster ids*. Routing to nodes within the same cluster is performed by means of a full mapped table. Routing information for nodes which are outside the local cluster is restricted to a single entry per cluster maintained in a cluster table. This cluster table could have a flat structure or further hierarchies. This type of storage savings are used in the SGI-SPIDER (2-level) [12] and the Servernet-II router [13] (3-levels). For adaptive routing, the sub-cluster routing table as well as the cluster routing table(s) need to support multiple entries per index.

5.1.2 Interval/Universal Routing

Interval routing [25] reduces table-size on a router to the smallest possible size equal to the number of router ports, thus making the table-size independent of the number of nodes in the interconnection network. This is achieved by a node-labeling scheme, wherein nodes with contiguous labels within a specified interval can be routed to using the same router exit port. The number of such non-overlapping intervals is equal to the number of router exit ports. It has been shown that interval labeling schemes can be derived for any connected network, hence also the name *universal routing*. The Transputer/C-104 switch [18] uses interval routing.

This scheme has significant advantages in terms of the small table size, excellent scalability and applicability to arbitrary topologies. However, in general, it cannot guarantee minimal paths for routing and requires specific labeling schemes, deadlock freedom for routing algorithms is not simple to specify, and this scheme is not readily receptive to adaptive routing.

5.2 Economical Storage for Mesh-like networks

The three schemes, full-table routing, meta-table routing and interval routing schemes do not use topology specific information to optimize routing table size. However, most often such routers are used in fairly regular topologies such as hypercubes, meshes and tori. Here, we propose a scheme, which uses topology-specific optimizations for n-dimensional mesh-like networks, to reduce table size for fully-adaptive routing, while retaining the advantages of programmable routing tables. We call this scheme *economical storage* (ES).

Consider a 2-D mesh network with node labels specified in (X, Y) Cartesian coordinates. Each router in such a network has five exit ports — four in the 4 coordinate directions $+X$, $+Y$, $-X$, $-Y$ and one port 0 to exit the interconnection network if the current node is the destination. From any given node in this network, at most two output port choices exist for routing to any other node using minimal paths. Without loss of generality, let this source node be at the origin. Now, all destination nodes in any one of the four quadrants, say the quadrant $(X > 0, Y > 0)$, can be routed to using one of the two ports choices, $+X$ or $+Y$ in this case. Any destinations on the four axes, say the positive X-axis $(X > 0, Y = 0)$, can be routed to using only one port, which in this case is $+X$ port. Finally, the last case is if the destination port is the current node itself (origin $(0, 0)$), which can be routed to using port 0. Thus, for any arbitrary sized 2-D mesh network, only 9 table entries, each with up to two output-port choices, are required in a router to implement minimal path fully-adaptive routing.

5.2.1 Implementing Economical-Storage Routing Tables

Assume that nodes in the 2-D mesh are labeled with Cartesian coordinates. Let $D = (dx, dy)$ be a destination node specified in the header of a message arriving at an intermediate router $I = (ix, iy)$. The router computes the sign of the relative coordinates of the destination, by computing

$$sx = sign(dx - ix), \text{ and}$$

$$sy = sign(dy - iy),$$

where, $sx, sy \in \{+, -, 0\}$. The signs sx and sy together are used to index into the 9 entry routing table to determine the output port(s) to be taken to route to destination D.

The actual hardware requirements apart form the routing table are a *node-id* register on the router and two comparators to find sx and sy used to index into the routing table.

Fig. 7 shows an example of a 9-node 2-D mesh, and shows how the router at an intermediate node $4 = (1, 1)$ would be programmed for North-Last partially adaptive routing (based on the Turn Model [15]). It should be noted from this example that although 2 output ports may be available to route to some destination, specific routing algorithms could deny them for guaranteeing deadlock freedom.

The 9 entry table for 2-D meshes stems from the 3 choices $\{+, -, 0\}$ each for sx and sy. Extending the economical storage scheme for 3-D mesh routers requires a 27 entry routing table. In general, for n-dimensional k^n-node meshes, a 3^n size table would suffice while full-table routing would require a k^n node routing table. Implementation concerns usually restrict mesh interconnects to small n (typically 2 or 3) and large k (typically 8 to 12). For example, the 2048 node 3-D interconnect in Cray T3D uses a 2048 entry routing table, which could be reduced to a 27 entry table using the economical storage scheme.

In the interest of brevity, we have only presented a basic implementation for the ES scheme here. It is, however, possible to implement ES with lookahead, provide minimal path routing n-dimensional tori, and support irregular topologies. (See [23].)

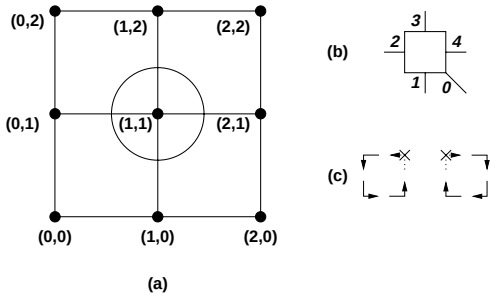
5.2.2 Adaptive Routing Performance Comparison with Various Table Storage Schemes

Table storage optimizations come with an associated tradeoff — that of decreased routing flexibility. It is thus, important to study the performance impact of lowering table storage requirements. We analyze this impact on the performance of adaptive routing in two-dimensional meshes. We compare the performance of full-table routing, meta-table routing (with a 2-level hierarchy) and economical storage routing.

Full-table routing offers complete flexibility in routing, where routing paths from each source to every destination can be individually configured. This flexibility is, however, rarely useful. For example, perhaps all the popular adaptive mesh routing algorithms use network symmetry and source-relative directions for routing to the destination for simplicity of the algorithm and proof of deadlock freedom.

In the case of meta-table routing, there is some loss of flexibility. In a (two-level) meta-table implementation an n-bit node-id is partitioned (into 2) to derive a cluster id and a sub-cluster id. For routing to any node in a distinct cluster, the same set of output ports have to be used. This implies that node-ids must be appropriately assigned to provide maximal flexibility in routing. For the sake of performance comparisons, we have used 2 different node-labeling schemes for studying meta-table routing in meshes. These mappings are presented in Fig. 8. The first mapping, Fig. 8 (a) permits minimal flexibility. This is because all nodes within a cluster are in a single row, which implies no-flexibility in routing within the sub-cluster. Similarly, no flexibility exists in routing to other clusters, because clusters are arranged in a single column. This map forces routing to be equivalent to deterministic XY routing. The second mapping, Fig. 8 (b), permits maximal flexibility. This is because, each sub-cluster is a square (4×4) mesh, permitting maximal adaptivity within the cluster. Clusters are also arranged in a (4×4) configuration which permits flexibility in routing to other clusters.

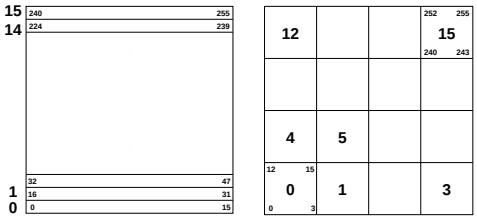
In economical storage routing, there is no real loss of flexibility as compared to full-table routing. This is because, all the known



Dest node	Sign sx	Sign sy	Candidate Router Ports	Tbl-Entries North-Last Rout.
(0,0)	-	-	2, 1	2, 1
(1,0)	0	-	1	1
(2,0)	+	-	4, 1	4, 1
(0,1)	-	0	2	2
(1,1)	0	0	0	0
(2,1)	+	0	4	4
(0,2)	-	+	2, 3	2
(1,2)	0	+	3	3
(2,2)	+	+	4, 3	4

(d)

Figure 7: Table programming for a 2-D mesh network router using economical storage. (a) 9-node mesh with node labels in (X,Y) coordinates. Node (1,1) is the source router under consideration. (b) Output port labels for 5 port 2-D mesh router. (c) Permitted turns in North-Last routing. Turns with dotted lines in the figure are disallowed for guaranteeing deadlock. (d) Economical Storage Table programming for North-Last routing.



(a) (b)

Figure 8: Meta-table mapping for a 256 node 2-D mesh for (a) minimal, and (b) maximal adaptivity in routing. In each of the figures the labels in small-type indicate the complete address of a node in the mesh, and the labels in large type indicate cluster labels.

mesh routing algorithms such as those in [15, 9, 22, 2] can be implemented with the economical storage. Hence, performance of full-table routing and economical storage routing are identical.

The results comparing the performance of the two meta-table mapping schemes and the full-table and ES tables are presented in Table 4. As the node labels in the meta-table map for maximal adaptivity differ from the conventional mesh node labeling scheme, we have ensured that when considering the various traffic patterns used for evaluation, source and destination locations (and not labels) are preserved. This makes the performance comparisons meaningful.

Traffic	Load	Meta-Tbl Adp.	Meta-Tbl Det.	Full-Tbl-Adp./ Econ. Storage
Unif.	0.1	71.5	69.2	69.2
	0.2	82.3	74.0	74.0
	0.3	294.1	80.6	80.5
	0.4	Sat.	87.4	87.2
	0.5	Sat.	97.8	97.5
	0.6	Sat.	111.5	111.0
	0.7	Sat.	132.2	130.4
	0.8	Sat.	169.3	168.6
	0.9	Sat.	289.1	432.8
Trans.	0.1	1024.1	74.6	74.5
	0.2	1632.7	88.5	87.6
	0.3	Sat.	746.6	294.6
	0.4	Sat.	1485.0	715.6
	0.5	Sat.	Sat.	853.5
Bit-Rev.	0.1	77.5	76.3	76.1
	0.2	103.3	95.0	93.6
	0.3	1164.8	1033.2	411.2
	0.4	Sat.	Sat.	1155.3

Table 4: Performance Comparison of Table-Storage Schemes. (Sat. indicates that network saturation has occured.)

A counter intuitive result that is seen is that the performance of meta-table routing with the minimal flexibility mapping (Meta-Tbl-Det) performs better than the mapping for maximal flexibility (Meta-Tbl). This behavior occurs because of large link contention at the links at cluster boundaries in the latter mapping. To understand why this behavior is exhibited consider a message being sent from any node in cluster 0 of Fig. 8 (b) to any node in cluster 5. This message could route adaptively until it cross over into cluster 4 at the north boundary or into cluster 1 at the east boundary. When the message reaches either cluster 4 or cluster 1, it can no longer route adaptively until it crosses over into cluster 5. Although, it can once again route adaptively within cluster 5 to reach its destination, the loss of adaptivity at the boundary nodes of cluster 4 or 1 causes unbalanced congestion at these links resulting in high latencies and premature saturation of the network. As a result, despite adaptive routing capabilities, this scheme performs worse than even deterministic routing.

Router Property	Full-Table	m-Level Meta Table	Interval	Economical Storage
Table Size	2^N	$m.2^{N/m}$ (optimal)	#-ports (Indep. of Net. Size)	9 (2-dim), 27 (3-dim)
Scalability	Poor	Better	Great	Great
Adaptivity Possible ?	Yes	Yes (limit.)	Not-direct	Yes
Topology	Arbitrary	Fairly Arbit.	Arbitrary	Meshes, Tori, Irregular
Lookup Time ($\propto$ tbl-size)	Possibly High	Low	Small	Small
Commercial Routers	T3D, T3E, S3.mp	SPIDER ($m = 2$), Servernet-II ($m = 3$)	C-104 Transputer	None (Proposed here)

Table 5: A summary of the relation between table storage optimizations and router properties considering a 2^N node network. Commercial implementations of various table storage optimizations are also summarized.

Although meta-table based adaptive routing holds the promise of reducing table-storage, it results in poor adaptive routing performance, at least in the case of mesh networks. It is possible, however, that adaptive routing with meta-table implementation may demonstrate good performance for topologies where intra-cluster messages do not interfere with inter-cluster message (due to distinct links being used), such as in the case of hypercubes. On the other hand, we find that the novel economical storage scheme offers all the benefits of table-lookup routing at a small storage cost without affecting performance of adaptive routing. A summary of schemes to reduce table storage in presented in Table 5.

6 Concluding Remarks

With the increasing use of multiprocessor networks in more demanding as well as general purpose application environments, where the workload could be high, fluctuating and may require other service guarantees, latency reduction becomes even more meaningful. While prior research has shown that adaptive routing can help in this regard, very little attention has been paid to their practicality of implementation. Thus, the idea has not been well received in commercial routers. This paper, therefore, considers the feasibility of supporting adaptivity in the context of current wormhole switched, pipelined-router designs and proposes three enhancements, together called as the LAPSES approach, which can supplement each other in providing a cost-effective solution to adaptive routing.

The first solution, known as *look-ahead routing* decouples the table lookup and arbitration stages of the pipelined design and uses the current routing table entry for the next routing step. The second solution proposes three new path selection heuristics, known as *least recently used* (LRU), *least frequently used* (LFU) and *maximum credit* (MAX-CREDIT) for selecting one of the available alternate paths due to routing adaptivity. The third solution attempts to reduce storage requirement for implementing adaptive routing via routing tables. We first show how one can use meta-table routing that can reduce the memory requirement. Next, we propose an economic table-based implementation of adaptive routing that needs only 3^n table entries (9 or 27 entries for a 2-D or 3-D topology) for an n-dimensional mesh. This implementation reduces the storage requirement drastically, can implement all proposed mesh/ torus routing algorithms and has identical performance to that of the full-table scheme. The lookahead and path selection policies help network performance while the economic table implementation addresses the storage cost concern.

We analyze the performance implications of these designs via simulation on a (16×16) mesh network that uses this router architecture and several traffic patterns. The lookahead mechanism is shown to benefit the latency at low load while the advantage of adaptivity kicks in at high load. Thus, the combined lookahead adaptive router seems to a good choice for the entire workload. The results further suggest that the look-ahead feature should be more attractive for short message transfer typically encountered in shared memory systems. It is shown that the new path selection strategies can utilize the available paths more prudently than the static-XY scheme and the MIN-MUX scheme [9] and hence can contribute to low message latency. The network latency thus reduces significantly for non-uniform traffic patterns. The two-level, meta-table implementation of adaptive routing algorithm severely affects the performance in a 2-D mesh due to traffic congestion at the boundary nodes between clusters. This suggest that unless the inter-cluster and intra-cluster communications use separate links, meta-table routing is not a good choice. Separate link traversal is feasible in networks such as hypercubes, but not in flat mesh topologies. We plan to evaluate these designs with various application workloads and other service requirements for quantifying the performance improvements more accurately.

Acknowledgments

Discussion on several of the issues regarding implementation of routers and performance enhancement schemes in this paper were possible due to the help and suggestions provided by Mike Galles of SGI, Steve Scott of SGI/Cray, Jose Duato of Universidad Politéchnica de Valencia, Wolf-Dietrich Weber of HAL, Andreas Nowatzyk of Compaq Western Research Labs and Dave Garcia of Compaq/Tandem. The authors gratefully acknowledge the help from all.

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, February 1995.

[2] Y. M. Boura and C. R. Das. Efficient Fully Adaptive Wormhole Routing in n-dimensional Meshes. In *Proc. Intl. Conf. on Distributed Computing Systems*, pages 589–596, June 1994.

[3] J. Carbonaro and F. Verhoorn. Cavallino: The Teraflops Router and NIC. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 4)*, pages 157–160, August 1996.

[4] A. A. Chien. A Cost and Speed Model for k-ary n-cube Wormhole Routers. In *Proc. Symp. High Performance Interconnects (Hot Interconnects)*, August 1993.

[5] A. A. Chien and J. H. Kim. Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors. *Journal of the ACM*, 40(1):91–123, January 1995.

[6] W. J. Dally. Virtual-Channel Flow Control. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):194–205, May 1992.

[7] W. J. Dally and H. Aoki. Deadlock-Free Adaptive Routing in Multicomputer Network using Virtual Channels. *IEEE Trans. on Parallel and Distributed Systems*, 4:466–475, April 1993.

[8] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. Arhitecture and Implementation of the Reliable Router. In *Proc. of Hot Interconnects II*, Stanford University, Palo Alto, CA, August 1994.

[9] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Trans. on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

[10] J. Duato, S. Yalamanchili, and L. M. Ni. *Interconnection Networks: An Engineering Approach*. IEEE CS Press, 1997.

[11] M. L. Fulgham and L. Snyder. Performance of Chaos and Oblivous Routers under Non-Uniform Traffic. Technical Report UW–CSE–93–06–01, Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195, July 1994.

[12] M. Galles. Scalable Pipelined Interconnect for Distributed Endpoint Routing : The SGI SPIDER Chip. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 4)*, pages 141–146, August 1996.

[13] D. Garcia and W. Watson. Servernet II. In *Proc. of the 1997 Par. Computing, Routing, and Comm. Workshop (PCRCW'97)*, June 1997.

[14] C. J. Glass and L. M. Ni. Maximally Fully Adaptive Routing in 2D Meshes. In *Proc. Intl Conf. on Parallel Processing*, August 1992.

[15] C. J. Glass and L. M. Ni. A Turn Model for Adaptive Routing. In *Proc. Intl. Symp. on Computer Architecture*, pages 278–287, May 1992.

[16] D. Jadav and A. Choudhary. Designing and Implementing High-Performance Media-on-Demand Servers. *IEEE Parallel & Distributed Technology*, pages 29–39, Summer 1995.

[17] S. Konstantinidou and L. Snyder. The Chaos Router. *IEEE Trans. on Computers*, 43(12):1386–1397, December 1994.

[18] M. D. May. The Next Generation Transputers and Beyond. In *Proc. 2nd European Distrinuted Memory Conf.*, pages 7–22, April 1991.

[19] A. G. Nowatzyk, M. C. Browne, E. J. Kelly, and M. Parkin. S-Connect: from Network of Workstations to Supercomputer Performance. In *Proc. of the 22nd Annual International Symposium on Computer Architecture*, pages 71–82, June 1995.

[20] S. L. Scott and G. M. Thorson. Optimized Routing in the Cray T3D. In *Proc. Parallel Computer Routing and Communications Workshop (PCRCW)*, pages 281–294. Springer Verlag Lecture Notes in Computer Science, May 1994.

[21] S. L. Scott and G. M. Thorson. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 4)*, pages 147–156, August 1996.

[22] C. Su and K. G. Shin. Adaptive Deadlock-Free Routing in Multicomputers Using Only One Extra Channel. In *Proc. Intl. Conf. on Parallel Processing*, volume I, pages 227–231, August 1993.

[23] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das. LAPSES: A Recpie for Adaptive Router Design. Technical Report CSE–98–010, Department of Computer Science and Engineering, The Pennsylvania State University, 220 Pond Lab, University Park, PA, 1997.

[24] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das. The PROUD Pipelined Router Architectures for High Performance Networks. Technical Report CSE–97–007, Department of Computer Science and Engineering, The Pennsylvania State University, 220 Pond Lab, University Park, PA, 1997.

[25] J. van Leeuwen and R. B. Tan. Interval Routing. *The Computer Journal*, 30(4):298–307, 1987.

[26] W.-D. Weber, S. Gold, P. Helland, T. Shimizu, T. Wicki, and W. Wilcke. The Mercury Interconnect Architecture: A Cost-effective Infrastructure for High-Performance Servers. In *Proc. International Symposium on Computer Architecture*, pages 98–107. ACM, 1997.