# Evaluation of a Hybrid Deterministic/Adaptive Router and Its Implementations

Dianne Kumar and Walid Najjar
*Department of Computer Science*
Colorado State University
Ft. Collins, CO 80523 USA

**Abstract**

A novel routing scheme is proposed for virtual cut-through routing on k-ary n-cube networks. This scheme attempts to combine the low routing delay of deterministic routing with the flexibility and low queuing delays of adaptive routing. In this hybrid routing scheme a message is routed as soon as possible along a minimal path to its destination even though the routing choice may not be optimal. Results show that the disadvantages of making a non-optimal routing decision are offset by its speed. Two pipelined implementations of this hybrid routing mechanism are evaluated and compared to pure deterministic and adaptive implementations. The experimental evaluations show that both hybrid implementations do indeed achieve their objectives under various types of traffic patterns.

**Index Terms:** interconnection networks, router architectures, virtual cut-through switching, adaptive routing, deterministic routing.

## 1 Introduction

This paper reports on the implementation and evaluation of a hybrid routing scheme that combines the advantages of deterministic and adaptive routing [1].

[1]A preliminary version of this work was reported in [28].

In the deterministic, or dimension-ordered, routing algorithm a message is routed along decreasing dimensions with a dimension decrease occurring only when zero hops remain in all higher dimensions. Virtual channels are included in the router to avoid deadlock [8]. Deterministic routing can suffer from congestion since only a single path between source and destination can be used.

In adaptive routing, messages are not restricted to a single path when traveling from source to destination. Moreover, the choice of path can be made dynamically in response to current network conditions. Such schemes are more flexible, can minimize unnecessary waiting, and can provide fault-tolerance. Several studies have demonstrated that adaptive routing can achieve a lower latency, for the same load, than deterministic routing when measured by a constant clock cycle for both routers [19, 25].

The delay experienced by a message, at each node, can be broken down into: *router* delay and *queuing* (or waiting) delay. The former is determined primarily by the complexity of the router. The latter is determined by the congestion at each node which in turn is determined by the degrees of freedom the routing algorithm allows a message. Note that the router delay is directly related to the clock cycle time of the router. The main performance advantage of adaptive routing (besides its fault-tolerance) is that it reduces the queuing delay by providing multiple path options.

However, the router delay for deterministic routers, and consequently their corresponding clock cycles, can be significantly lower than adaptive routers as pointed out in [1, 4]. This difference in router delays is due to two main reasons:

- *Number of virtual channels.* Two virtual channels are sufficient to avoid deadlock in dimension ordered routing [8]; while adaptive routing (as described in [11, 3]) requires a minimum of three virtual channels in $k$-ary $n$-cube networks.

- *Output channel selection.* In dimension-ordered routing, the output channel selection

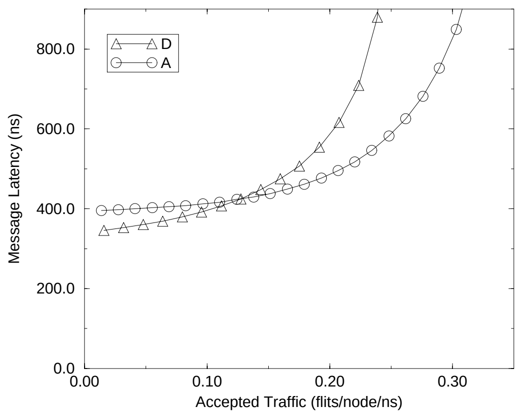


Figure 1: Message latency of deterministic (D) and adaptive (A) routing on a 10-ary 3-cube network under random uniform traffic and with message length of 8 flits.

policy is very simple: it depends only on information contained in the message header itself. In adaptive routing the output channel selection policy depends also on the state of the router (i.e the occupancy of various virtual channels) causing increased router complexity and thereby higher router delays.

The results reported in [1, 4] show that the router delays for adaptive routers are about one and a half to more than twice as long as the dimension-order router for worm-hole routing. The advantage of adaptive routing in reducing queuing delays in the nodes between source and destination is evaluated and reported in [13] for worm-hole routing. A typical comparison of deterministic versus adaptive routing message latencies (accounting for the differences in clock cycle times) is shown in Figure 1: at low traffic and for short to moderate message sizes, the latency of deterministic routing is smaller [13]. However, the flexibility of adaptive routing provides smaller queuing delays and a much higher saturation point.

In this paper we propose and evaluate a novel routing scheme for virtual cut-through switching that attempts to combine the low router delay of deterministic routing with the flexibility and low queuing delays of adaptive routing. The hybrid routing scheme is similar in concept to the hot potato algorithm and making the common case fast [2]: a message is routed as soon as possible although the choice may not be optimal, and this routing decision is fast. The results show that the disadvantages of making a non-optimal routing decision are offset by its speed. This hybrid routing mechanism relies on pipelined implementations where different paths and stages are used for different routing modes. The experimental evaluation of this router shows that it can achieve, under most conditions, the low latency of the deterministic approach as well as the high saturation point of the adaptive one.

The deterministic and adaptive routing algorithms are described in Section 2 along with the model of the routing delay for virtual cut-through switching. The hybrid routing scheme is described in Section 3. Results from the experimental evaluation comparing the hybrid router to the deterministic and adaptive ones under various traffic patterns for $k$-ary $n$-cube networks are reported in Section 4. Section 5 discusses related work and concluding remarks are given in Section 6.

## 2 Deterministic and Adaptive Routing

The interconnection network model considered in this study is a $k$-ary $n$-cube using virtual cut-through switching [22]: message advancement is similar to worm-hole routing [30], except that the body of a message can continue to progress even while the message head is blocked, and the entire message can be buffered at a single node. Note that a header flit can progress to a next node only if the whole message can fit in the destination buffer. For simplicity all messages are assumed to have the same length.

### 2.1 Routing Models

In the *deterministic* routing scheme (dimension-order routing) [6, 8], a message is routed along decreasing dimensions with a dimension decrease occurring only when zero hops remain in all higher dimensions. By assigning an order to the network dimensions, no cycle exists in the channel-dependency graph and the algorithm is deadlock-free.

The *adaptive routing* scheme considered in this work (Duato's or *-channels algorithm) is described in [12, 3]. In this algorithm, adaptive routing is obtained by using adaptive virtual channels along with dimension-order routing. A message is routed on any adaptive channel until it is blocked. Once blocked, a message is routed using dimension-order routing if possible. Note that a message may return to the adaptive channels in the following routing decisions if the adaptive channels are available. This algorithm has been proven to be deadlock-free as long as the following routing restrictions are imposed: when the message size is greater than the buffer size (i.e. size of the the virtual channel), deadlock is prevented by allowing the head flit of a message to advance to the next node only if the receiving queue at that node is empty. If the message size is less than the buffer size, then deadlock is prevented by allowing a message to advance only as long as the whole message fits in the receiving queue at that node. This algorithm requires a minimum of three virtual channels per dimension per node for each physical unidirectional channel. Therefore, the number of virtual channels grows linearly with the size of the network.

## 2.2 Switching Models

In this study, both the deterministic and adaptive routing schemes use one *unidirectional physical channel* (PC) per dimension per node. Figure 2 shows a schematic for each of the routers in 2D. In the deterministic routing case, both high and low virtual channels (VC) of each dimension are multiplexed onto one physical channel. In the adaptive routing case, the two deterministic and one adaptive VCs are multiplexed onto one PC. For both cases there is only one PC for the sink channel. Once this channel is assigned to a message, it is not released until the whole message has finished its transmission.

The deterministic router uses storage buffers associated with output channels, while the adaptive router uses storage buffers associated with input channels. When using output buffers, the routing decision must be made before buffering the message which is ideal for deterministic routing since only one choice is available for an incoming message.

When using input buffers, the routing decision must be made after buffering the message. It is suitable for adaptive routing since a message can usually be routed on several possible output channels. The adaptive router implements a round-robin input message selection policy which checks for messages first among all adaptive buffers and then among all deterministic buffers.

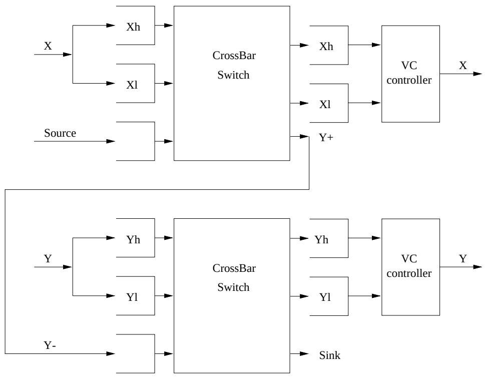
Output channel selection is performed by giving priority to those channels in the dimension with the greatest number of hops remaining for the selected message. Each dimension with decreasing number of remaining hops is tried until a free channel is found or all channels have been tried. By using this output channel selection policy, the greatest amount of adaptivity for a message is retained which reduces blocking.

## 2.3 Modeling Router Delay

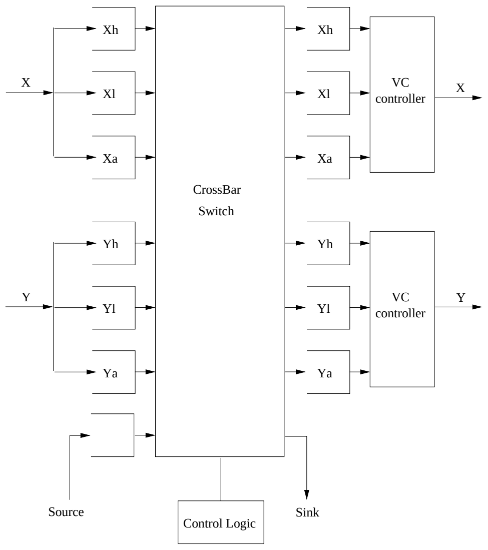
In this section we describe a router delay model for the virtual cut-though deterministic and adaptive routers. The model is based on the ones described in [4, 1, 13]. These models account for both the logic complexity of the routers as well as the size of the crossbar as determined by the number of virtual channels that are multiplexed on one physical channel. These models were modified to account for the varying buffer space used in virtual cut-through switching. The parameters of these models are:

KEY

Xh = high virtual channel in x dimension
Xl = low virtual channel in x dimension
Xa = adaptive virtual channel in x dimension
Yh = high virtual channel in y dimension
Yl = low virtual channel in y dimension
Ya = adaptive virtual channel in y dimension
X = physical channel in x dimension
Y = physical channel in y dimension



**(a)** Deterministic Router



**(b)** Adaptive Router

Figure 2: Schematics of deterministic and adaptive 2D routers

| Symbol | Variable (delay) |
|---|---|
| $T_{AD}$ | Address decoding |
| $T_{ARB}$ | Routing arbitration |
| $T_{CB}$ | Crossbar |
| $T_{FC}$ | Flow control |
| $T_{SEL}$ | Header selection |
| $T_{VC}$ | Virtual channel controller |
| $P$ | Max. no. of IP or OP ports in crossbar |
| $F$ | Degrees of freedom (OP choices of a message) |
| $C$ | No. of virtual channels |
| $B$ | Buffer size (in number of flits) |

The address decoding term ($T_{AD}$) includes the time for examining the packet header and creating new packet headers for all possible routes. The time required for selecting among all possible routes is included in the routing arbitration delay ($T_{ARB}$). The crossbar delay ($T_{CB}$) is the time necessary for data to go through the switch's crossbar and is usually implemented with a tree of gates. The flow control delay ($T_{FC}$) includes the time for flow control between routers so that buffers do not overflow. $T_{SEL}$ is the time for selecting the appropriate header. Finally, the virtual channel controller delay ($T_{VC}$) includes the time required for multiplexing virtual channels onto physical channels.

For all dimension-order routers simulated here, the number of degrees of freedom ($F$) equals one since there exists a single routing option for each message. The number of switch crossbar ports ($P$) is three because a deterministic router routes a message in either the same dimension on which the message came (on either the low or high channel) or routes it to the next dimension. For all of the adaptive routers, $F = P - 2(n-1)$ where $n$ is the number of network dimensions. This relationship holds because adaptive routing can use the adaptive channels in all the dimensions while only two virtual channels per physical channel can be used in dimension-order (to avoid deadlock). Note that this relationship includes the delivery port.

Delay equations for the routers are derived, using the above parameters. The constants in these equations were obtained in [4] using router designs along with gate-level timing estimates based on a 0.8 micron CMOS gate array process. Three main operations are used in all of the routers simulated here which contribute to the following three delays:

- $T_r$: Time required to route a message
- $T_s$: Time necessary to transfer a flit to the corresponding output channel
- $T_c$: Time required to transfer a flit across a PC

The equations are:

$$T_r = T_{AD} + T_{ARB} + T_{SEL}$$
$$T_r = 2.7 + 0.6 + 0.6 * \log_2 F + 1.4 + 0.6 * \log_2 F$$

$$T_s = T_{FC} + T_{CB} + T_{Latch}$$
$$T_s = 0.8 + 0.6 * \log_2 B + 0.4 + 0.6 * \log_2 P + 0.8$$

$$T_c = 4.9 + T_{VC}$$
$$T_c = 4.9 + 1.24 + 0.6 * \log_2 C$$

Using the above equations, the delay values were calculated for each of the router algorithms simulated and are shown in Table 1. To decrease the overall router delay, it is assumed that all three operations are overlapped through pipelining as described in [13], and therefore the clock period is determined by the longest delay:

$$T_{ccperiod} = Max(T_r, T_s, T_c)$$

From the data in Table 1, we observe that increasing the buffer size, in deterministic routers, increases the overall router delay only when large buffer sizes are used. For small and moderate buffer sizes the clock cycle is dominated by the transfer time $T_c$ while for larger ones it is dominated by the switching time $T_s$. In adaptive routers, the clock cycle

| $B$ | $T_r$ | $T_s$ | $T_c$ | CC Period |
|---|---|---|---|---|
| 8 | 4.70 | 4.75 | 6.74 | 6.74 |
| 16 | 4.70 | 5.35 | 6.74 | 6.74 |
| 24 | 4.70 | 5.70 | 6.74 | 6.74 |
| 32 | 4.70 | 5.95 | 6.74 | 6.74 |
| 48 | 4.70 | 6.30 | 6.74 | 6.74 |
| 64 | 4.70 | 6.55 | 6.74 | 6.74 |
| 96 | 4.70 | 6.90 | 6.74 | 6.90 |

**a- Deterministic router**
**($C = 2$, $P = 3$, and $F = 1$ for all)**

| $B$ | $T_r$ | $T_s$ | $T_c$ | CC Period |
|---|---|---|---|---|
| 8 | 7.80 | 5.79 | 7.09 | 7.80 |
| 16 | 7.80 | 6.39 | 7.09 | 7.80 |
| 24 | 7.80 | 6.74 | 7.09 | 7.80 |
| 32 | 7.80 | 6.99 | 7.09 | 7.80 |
| 48 | 7.80 | 7.34 | 7.09 | 7.80 |
| 64 | 7.80 | 7.59 | 7.09 | 7.80 |
| 96 | 7.80 | 7.94 | 7.09 | 7.94 |

**b- Adaptive router**
**($C = 3$ and $P = 10$ and $F = 6$ for all)**

Table 1: Deterministic and adaptive router delays (all values in $nsec$) for $k$-ary 3-cube networks

time is dominated by $T_r$. Increasing buffer size increases the overall router delay only when very large buffer sizes are used.

All of these added delays result in adaptive routers that are 15 to 16 % slower than deterministic routers. These results are similar to the results in [1] where 15% to 60% improvement is required for f-flat routers with similar number of virtual channels and under worm-hole routing.

## 3 Hybrid Routing

This section describes the mechanism of the hybrid routing scheme along with two implementations: a Pipelined Hybrid Router and a Super-Pipelined Hybrid Router.

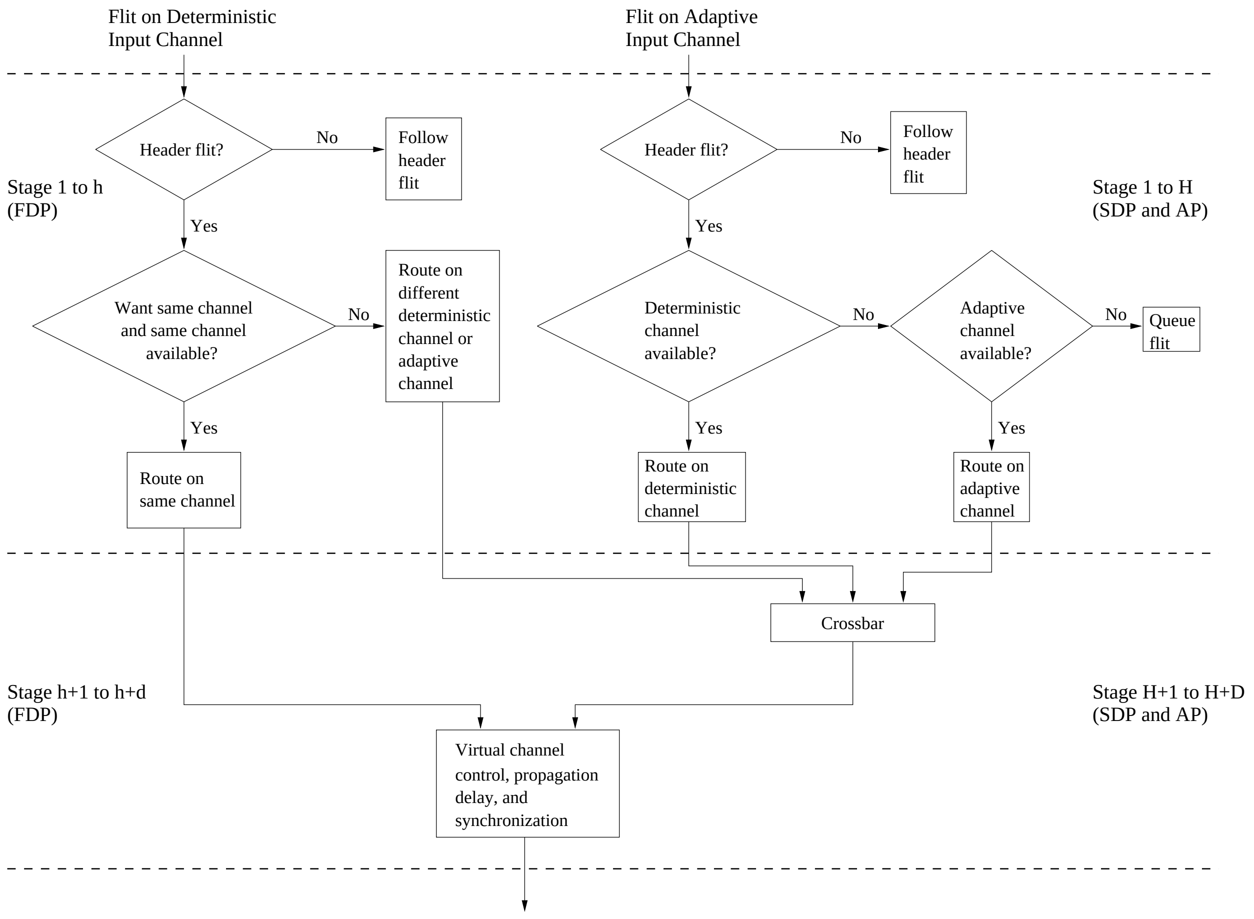

Flit on Deterministic Input Channel
Flit on Adaptive Input Channel
Header flit?
No
Follow header flit
Header flit?
No
Follow header flit
Stage 1 to h (FDP)
Stage 1 to H (SDP and AP)
Yes
Yes
Want same channel and same channel available?
No
Route on different deterministic channel or adaptive channel
Deterministic channel available?
No
Adaptive channel available?
No
Queue flit
Yes
Yes
Yes
Route on same channel
Route on deterministic channel
Route on adaptive channel
Crossbar
Stage h+1 to h+d (FDP)
Stage H+1 to H+D (SDP and AP)
Virtual channel control, propagation delay, and synchronization

Figure 3: Flow chart of hybrid routing algorithm

## 3.1 Hybrid Router Model

The hybrid router consists of three logically independent and pipelined message paths: a Fast Deterministic Path (FDP), a Slow Deterministic Path (SDP), and an Adaptive Path (AP)[2]. The routing algorithm is shown in Figure 3 while the pipeline stages of the router are shown in Figure 4 and 5. Note that the longest stage in *all* paths determines the maximum cycle time of the hybrid router.

The Fast Deterministic Path has the highest priority and is used for a message flit entering on a deterministic channel that is also able to leave on a deterministic channel of the same type (low/high) and dimension. Although the choice to route deterministically first may

[2]Some physical stages are actually shared among these logically independent paths.

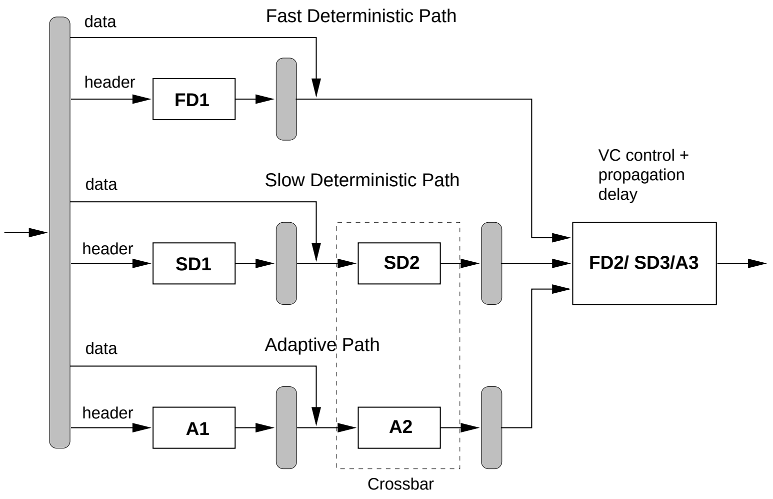


Figure 4: Logic schematic of Pipelined Hybrid Router.

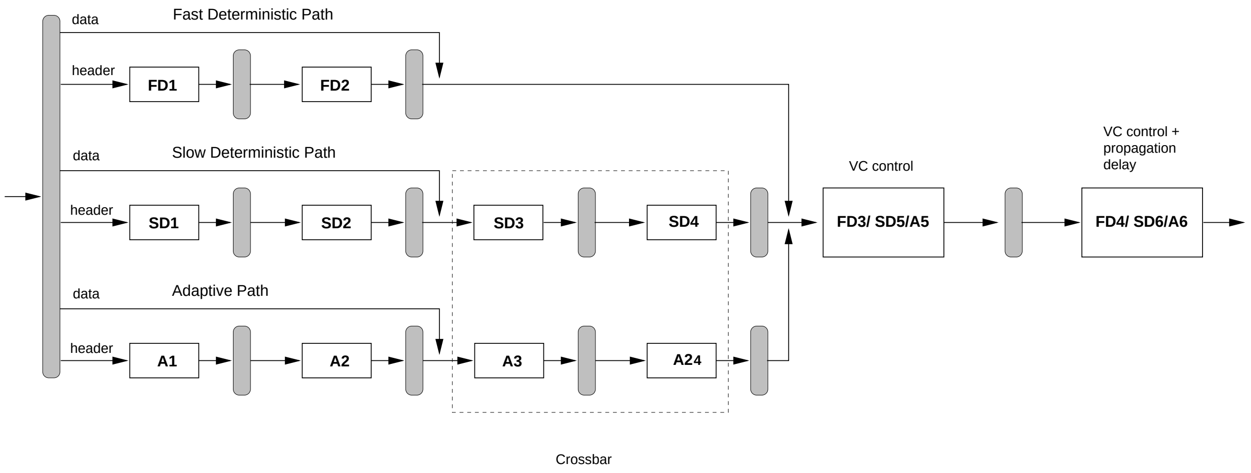


Figure 5: Logic schematic of Super-Pipelined Hybrid Router.

reduce the adaptivity of messages, the routing decision and switching logic along this fast path is simpler than the traditional deterministic and adaptive routing and requires the least number of stages: $h + d$ stages for a header flit and $d$ stages for a data flit.

If a message cannot be routed along the Fast Deterministic Path (i.e. if a deterministic channel of the same type is not available or a message is being switched to a different type or dimension), then the message is sent along the Slow Deterministic Path which requires more logic and therefore more stages than the Fast Deterministic Path.

The Adaptive Path has the lowest priority and is used when both the Fast Deterministic Path and Slow Deterministic Path are not available. In this case, a message is routed to an adaptive channel. Both the Slow Deterministic Path and Adaptive Path take $H + D$ clock cycles for a header flit and $D$ clock cycles for a data flit, where $(H + D) > (h + d)$.

Note that although a header flit requires more cycles than a data flit, a data flit must always follow a header flit. Therefore, a data flit will block if the header flit has not yet advanced through a given stage.

Giving a higher priority to the deterministic paths reduces the adaptivity (i.e. number of choices) of a message. At low traffic this loss is more than offset by the gains in router delay and translates in message latencies that are lower than those of an adaptive router. At very high traffic the probability of blocking is large and the loss of adaptivity translates into latencies that are sometimes larger than those of an adaptive router but always orders of magnitude lower than a deterministic router.

This routing scheme is deadlock free: for any given message, the choice of paths selected is always a true subset of those that could be selected by the adaptive algorithm described in [11]. Since the adaptive algorithm has been proven deadlock free, the hybrid is also deadlock free.

## 3.2 Pipelined Implementations

- The *Pipelined Hybrid Router* implementation is shown in Figure 4. It uses the flow chart in Figure 3 where $h = d = H = 1$ and $D = 2$ and corresponds to a 2/1 stage pipeline for the Fast Deterministic Path and a 3/2 stage pipeline for the Slow Deterministic Path and Adaptive Path. Because the routing decision and switching logic of the Fast Deterministic Path is simplest among all the paths, the $T_r$ and $T_s$ delays combine into one stage ($FD1$), while the $T_c$ delay is kept in a separate stage ($FD2$). The Slow Deterministic Path and Adaptive Path are more complex and require separate stages for each of the $T_r$, $T_s$, and $T_c$ delays, resulting in a 3/2 stage pipeline. Note that the crossbar is physically shared between both Slow Deterministic Path and Adaptive Path and all paths share the virtual channel control logic.

- The *Super-Pipelined Hybrid Router* relies on deep pipelines to implement the hybrid router. Using deep pipelines can increase overall throughput at the cost of additional latch delays. Also the clock skew becomes more prominent: if the clock cycle becomes as small as the sum of the clock skew and latch overhead, further pipelining is no longer useful. An important factor to consider is an efficient use of the pipeline stages. Since the stages in all three paths are efficiently used in the Pipelined Hybrid Router, the work in each stage of the Pipelined Hybrid Router is divided into two stages in the Super-Pipelined Hybrid Router. Therefore, the 2/1 stage pipeline in the Fast Deterministic Path of the Pipelined Hybrid Router becomes a 4/2 stage pipeline in the Super-Pipelined Hybrid Router, while the Slow Deterministic Path and Adaptive Path paths are modified from 3/2 stage pipelines to 6/4 stage pipelines. Once again, the crossbar is physically shared between both Slow Deterministic Path and Adaptive Path and all paths share the virtual channel control logic. Figure 5 shows the new schematic for this super pipeline. Note that the main difference between the Pipelined Hybrid Router and the Super-Pipelined Hybrid Router is the number of stages required for each path. The flow chart in Figure 3 is used in the Super-Pipelined Hybrid Router where $h = d = H = 2$ and $D = 4$.

## 3.3 Clock Cycle Times

The performance of the pipelined and super-pipelined implementations of the hybrid router is compared to pipelined and super-pipelined implementations of both the deterministic and adaptive routers.

**Pipelined Router Implementation.** Both the deterministic and adaptive routers are implemented as a 3/2 stage pipeline, where 3 stages are required for a header flit and 2 stages are required for a data flit. The clock cycle times for both are obtained using the equations in Section 2.3.

The clock cycle time of the Pipelined Hybrid Router is one gate delay larger than that of the adaptive router to account for the increased critical path length.

Note that the Fast Deterministic Path stage (2/1 stage pipeline) used in the hybrid router is not part of the deterministic router. Including it would increase the clock cycle time of the deterministic router which would offset any advantage gained from having fewer number of cycles.

**Super-Pipelined Router Implementation.** Since the stages are efficiently used in all the pipelined routers, the super-pipelined implementation for all routers consists of dividing the work found in each stage of its corresponding pipelined implementation into two. This results in a 6/4 stage super-pipeline for the deterministic and adaptive routers and a 4/2 stage super-pipeline for the Fast Deterministic Path of the hybrid router and a 6/4 stage

|  | Deterministic |  | Adaptive |  | Hybrid |  |
|---|---|---|---|---|---|---|
| $B$ | $PR$ | $S-PR$ | $PR$ | $S-PR$ | $PR$ | $S-PR$ |
| 8 | 6.74 | 3.80 | 7.80 | 4.40 | 8.40 | 5.00 |
| 16 | 6.74 | 3.80 | 7.80 | 4.40 | 8.40 | 5.00 |
| 32 | 6.74 | 3.80 | 7.80 | 4.40 | 8.40 | 5.00 |

Table 2: Clock cycle times for all three routers (in $nsec$) for $k$-ary 3-cube networks

super-pipeline for the Slow Deterministic Path and Adaptive Path.

The clock cycle times for the *super-pipelined* implementation of all routers were calculated using their corresponding *pipelined* clock cycle times in Equation 1.

$$CC_{S-PR} = \left\lceil \frac{(CC_{PR} - L)}{2.0 * G} \right\rceil * G + L \tag{1}$$

- $CC_{S-PR}$: CC of longest stage in *super-pipelined* router (S-PR) implementation
- $CC_{PR}$: CC of longest stage in *pipelined* router (PR) implementation
- $L$: Setup time for latch (0.8 ns)
- $G$: Delay for one gate (0.6 ns)

The super-pipelined cycle time for all the routers involves subtracting the latch setup delay to obtain the combinational logic delay which is then split in two in the super-pipelined router. The number of integer gate delays is then calculated and the latch setup time is added back. The cycle times for the pipelined and super-pipelined implementations of the three routers are shown in Table 2.

## 4 Experimental Evaluation

Simulations of the deterministic, adaptive and hybrid routing implementations were performed using a discrete-time simulator on an 8-ary 3-cube network. The simulations use a stabilization threshold of a 0.005 difference between traffic 1000 clock cycles apart to determine steady state. Message sizes varied from 8 to 32 flits and traffic from 0.1 until saturation was reached in 0.1 increments. The buffer sizes used in the simulation are all equal to a single message length. The adaptive router and the adaptive path in the hybrid router use three virtual channels per dimension. The deterministic router and the deterministic path in the hybrid router use two. The simulator implements a back-pressure mechanism which results in a negative slope of the latency versus accepted traffic plots at higher loads.

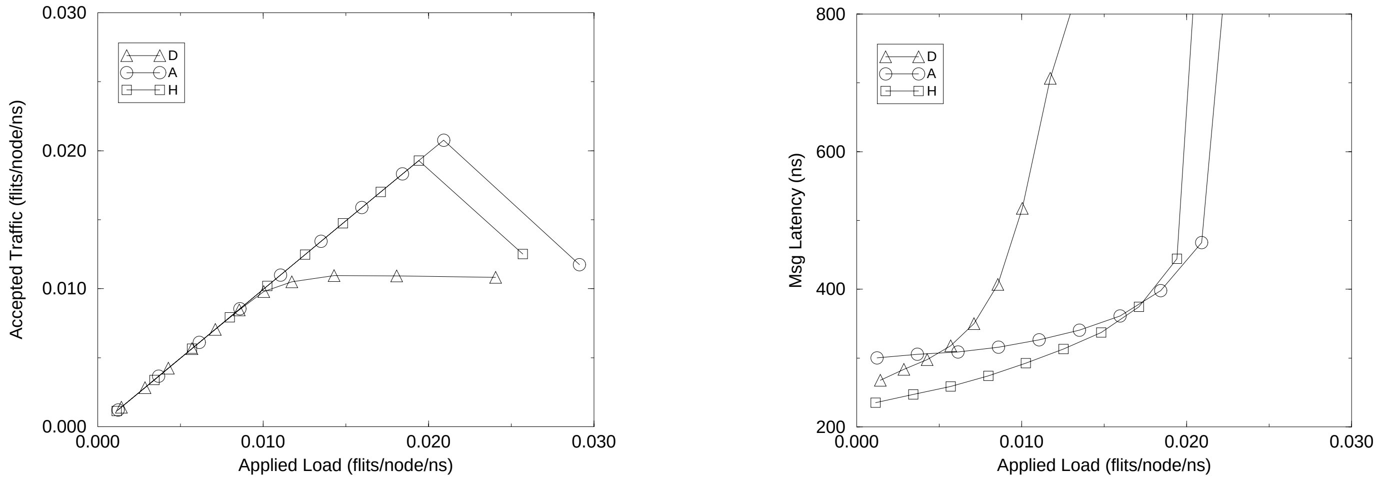
Five different traffic patterns were simulated. They are defined as follows:

- Random Uniform: Source and destination nodes are uniformly distributed.
- Complement: Node $a_{n-1}a_{n-2}...a_1a_0$ communicates with node $\overline{a_{n-1}a_{n-2}...a_1a_0}$
- Perfect Shuffle: Node $a_{n-1}a_{n-2}...a_1a_0$ communicates with node $a_{n-2}a_{n-3}...a_0a_{n-1}$
- Bit-Reversal: Node $a_{n-1}a_{n-2}...a_1a_0$ communicates with node $a_0, a_1, ...a_{n-2}, a_{n-1}$
- Butterfly: Node $a_{n-1}a_{n-2}...a_1a_0$ communicates with node $a_0a_{n-2}...a_1a_{n-1}$

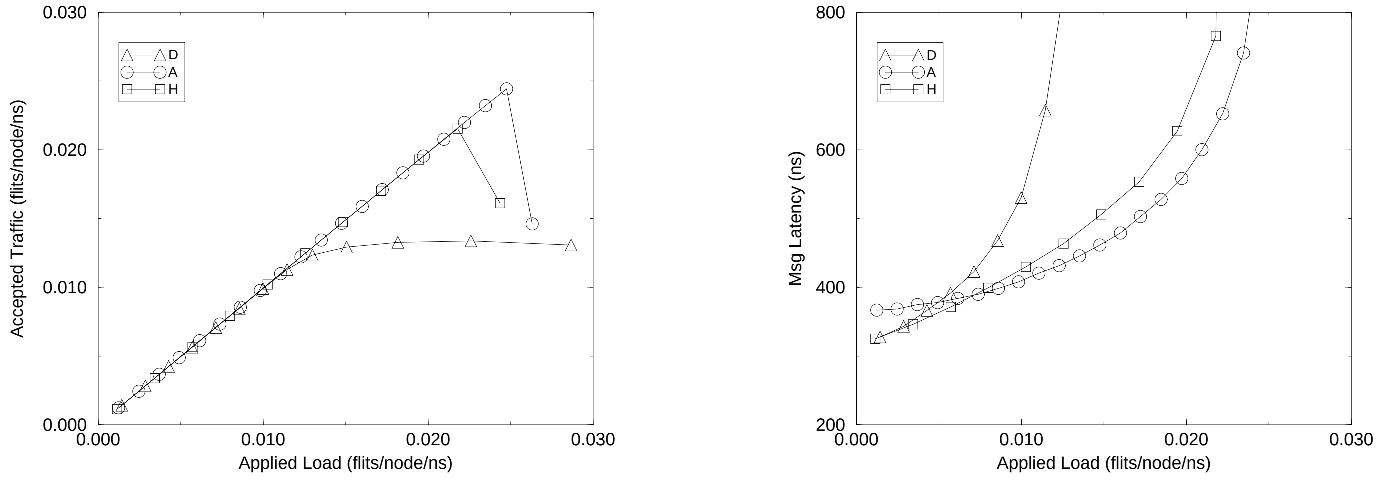
## 4.1 Performance of Hybrid Routing

Figure 6 shows the message latencies as well as the accepted load versus offered load plots of the deterministic, adaptive and hybrid pipelined implementations under random uniform traffic and message lengths of 8, 16, and 32 flits. Figure 7 shows similar plots for the complement and perfect shuffle traffic patterns. These two traffic patterns were selected because the behavior of the complement traffic is representative of all the others except the perfect shuffle.

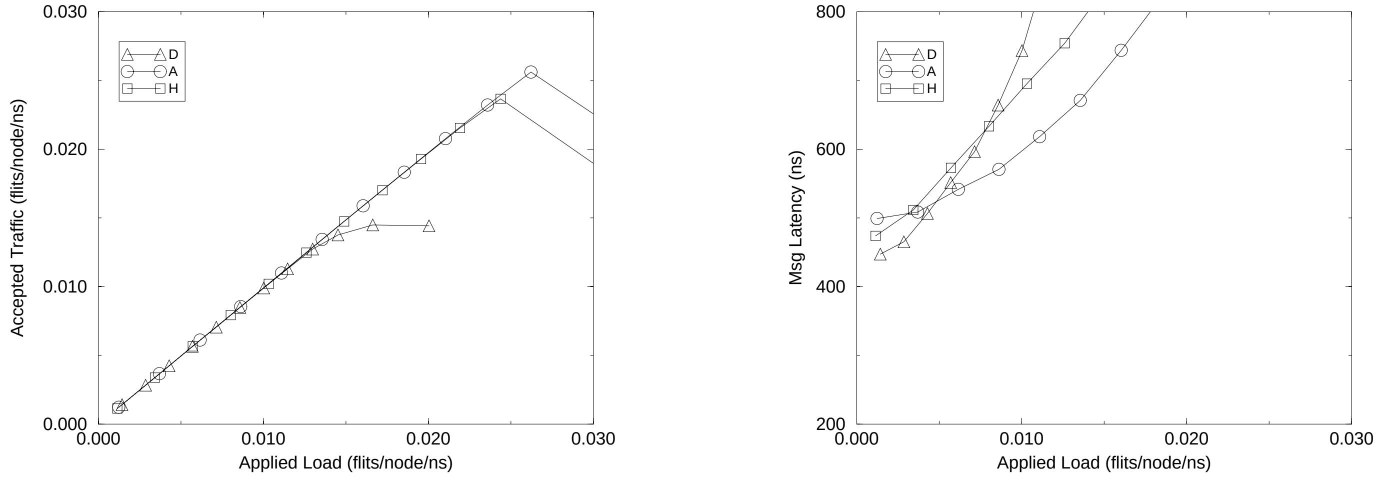
**Message Latency.** Under random uniform traffic, for small messages (8 flits) the latency of the Pipelined Hybrid Router is not only lower than the pipelined adaptive one but is also



**Message Length = 8**



**Message Length = 16**



**Message Length = 32**

Figure 6: Message latencies of deterministic, adaptive and hybrid pipelined implementation routers in an 8-ary 3-cube under random uniform traffic

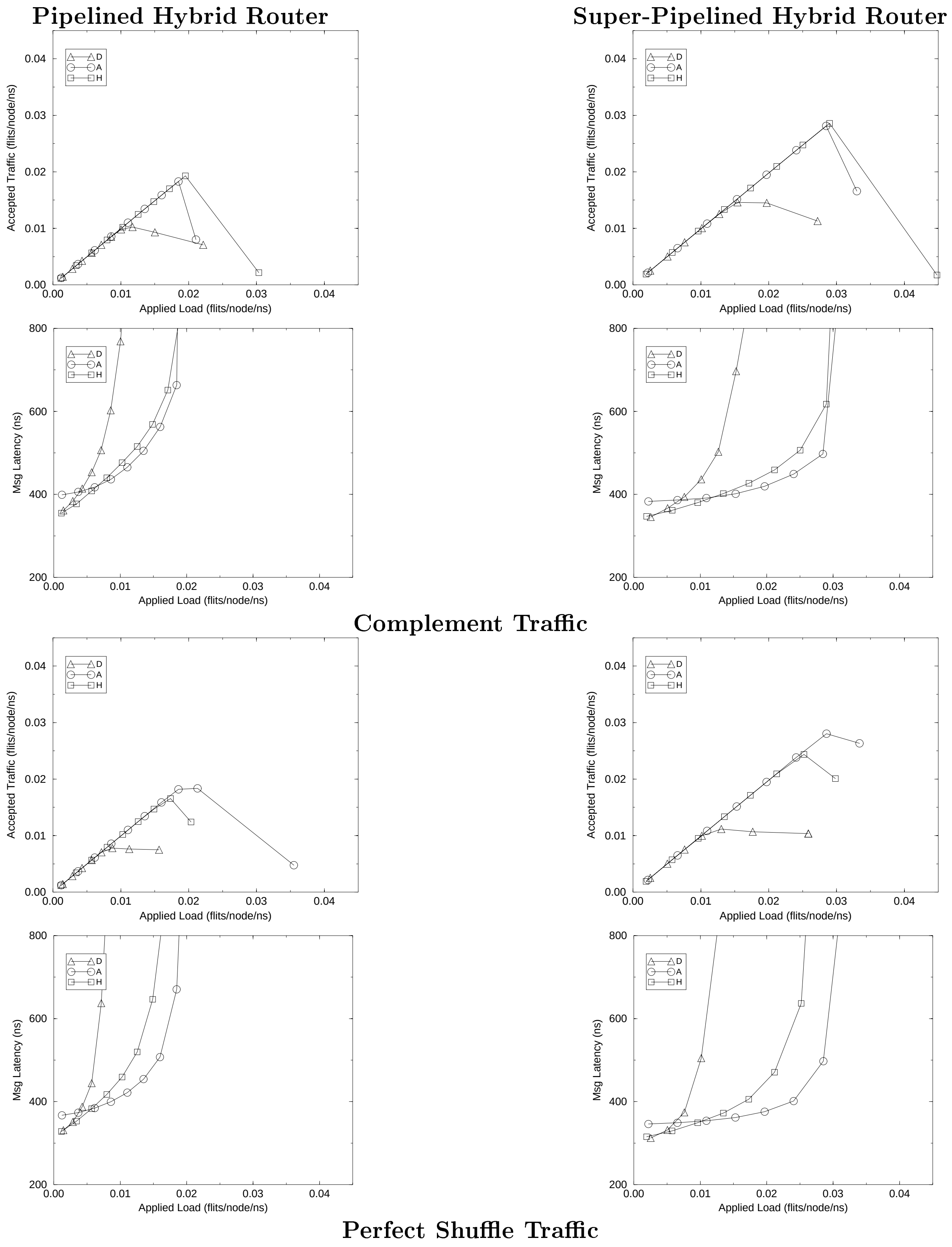


Figure 7: Comparison of pipelined and super-pipelined implementations of deterministic, adaptive and hybrid for 8-ary 3-cube (L=16)

lower than the pipelined deterministic one at low traffic. This is due to the fact that the Pipelined Hybrid Router has a 2/1 stage pipeline for header/data flits, while the deterministic router has a 3/2 stage pipeline. Even though the delay per stage in the deterministic router is shorter than the Pipelined Hybrid Router's, the greater number of stages dominates. For medium messages (16 flits) the latency of the Pipelined Hybrid Router is very close to that of the deterministic one at low traffic and follows the adaptive one at higher traffic. For larger messages (32 flits) the Pipelined Hybrid Router latency is less than the adaptive one at low traffic and greater than the adaptive one at high traffic. In general, the latency of the Pipelined Hybrid Router follows the deterministic one at low traffic and the adaptive one at high traffic.

**Effects of Message Length.** Note that as message size increases under random uniform traffic, the performance advantage of the Pipelined Hybrid Router decreases compared to the deterministic and adaptive pipelined routers. This is due to the facts that more messages, and therefore headers, are needed to achieve the same utilization with short message length and the Pipelined Hybrid Router has a performance advantage for header flits, especially at low utilization. While the deterministic router has a 3-stage header flit pipeline with a low clock cycle time, the Pipelined Hybrid Router has a 2-stage deterministic header flit pipeline with a higher clock cycle time. Since the number of pipeline stages dominates performance (and not the clock cycle time), the performance difference between the routers is greater for small message sizes than for large message sizes. This difference also exists at high traffic, although it's much smaller due to the fact that more message blocking occurs covering up differences in header flit time. This difference is exaggerated in larger sized networks because the average number of hops per message increases, thereby increasing the header flit contribution.

**Effects of Traffic Patterns.** The performance of the Pipelined Hybrid Router under all non-random traffic patterns is similar to that for random uniform traffic. Once again, the Pipelined Hybrid Router performs best at low traffic, while the adaptive router performs slightly better at high traffic. This is due to the higher priority given to the deterministic paths in the Pipelined Hybrid Router: less choices are available as a message is routed through the network on deterministic channels.

**Saturation Point.** The saturation points for all pipelined router implementations are shown in the accepted load versus offered load graphs in Figure 6 for varied message lengths. Under random uniform traffic, the saturation point of the Pipelined Hybrid Router is, in all cases, much higher than that of the pipelined deterministic router and is very close to the adaptive one. One reason for the slight decrease in saturation point for the Pipelined Hybrid Router with respect to the adaptive router, is that messages are routed onto the deterministic channels first, reducing the number of options available to a message later on. As traffic increases, this effect causes more blocking and slightly smaller saturation points. Under all non-random traffic (Figure 7) the Pipelined Hybrid Router's saturation point is once again much higher than that of the pipelined deterministic router and is very close to the adaptive one.

## 4.2 Effects of Super-Pipelining

The effects of super-pipelining on message latency are shown in Figure 7. Under all traffic patterns, the super-pipelined implementations for all routers achieve better overall performance gain than the pipelined implementations. This is due to the higher throughput that is achieved by deeper pipelines. However, the improvement is minimal with deterministic routers: deeper pipelines do not alleviate the congestion that results from deterministic routing. Because of the higher throughput, all super-pipelined routers achieve higher saturation points than the pipelined implementations.

## 4.3 Effect of Clock Cycle Time

For some of the hybrid router's cycle times, the ceiling function of the super-pipelined delay equation (Equation 1) results in very conservative estimates. In particular, the Super-Pipelined Hybrid Router's cycle time for message length equal to 16 flits results in a time that is equivalent to adding two gate delays to the adaptive router's cycle time instead of one. Since the hybrid router paths are similar in complexity to that of the adaptive router, this results in a very conservative estimate.

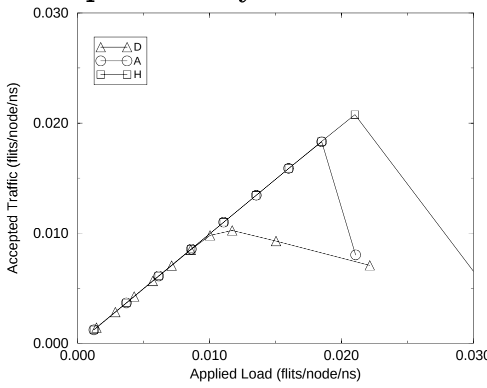
To obtain more accurate simulation results, a cycle time for the hybrid router equal to that of the adaptive router for a 16 flit message length was simulated. Once again only two of the most representative results of all traffic patterns are shown, with most results similar to those obtained under complement traffic. The worst performance was once again obtained under perfect shuffle traffic. The results are shown in Figure 8. In all cases, performance is either very close to or better than adaptive routing for all types of traffic simulated here.

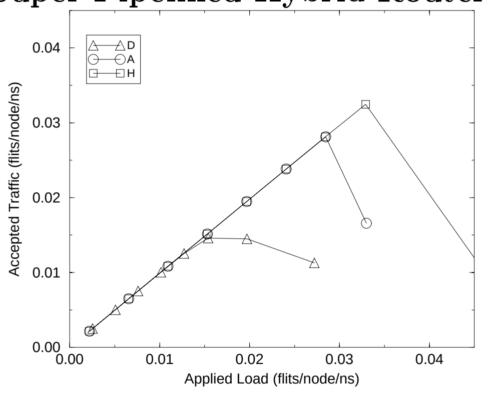
## 4.4 Effects of Path Priorities

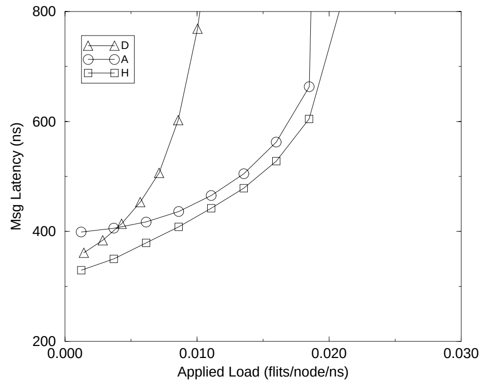
The hybrid router's implementation for all the previous results includes first routing on the Fast Deterministic Path, then on the Slow Deterministic Path, and finally on the Adaptive Path. This scenario is referred to as the Slow Deterministic Path scenario. However, by routing the Adaptive Path last, adaptivity that could be utilized at high loads may be lost. Therefore, simulations were performed to see if switching the priorities of the Adaptive Path and Slow Deterministic Path would improve performance near saturation. In this scenario, the Fast Deterministic Path is still given highest priority. However, the Adaptive Path is given the next highest priority, followed by the Slow Deterministic Path. This scenario is called the Adaptive Path scenario.

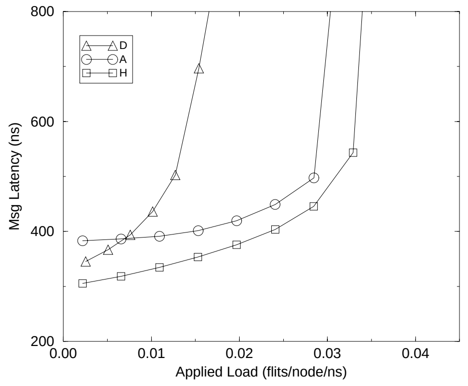
## Pipelined Hybrid Router

## Super-Pipelined Hybrid Router

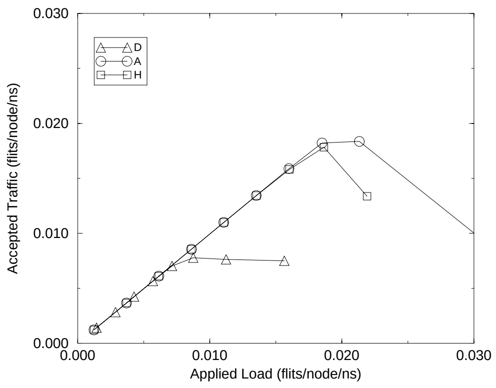


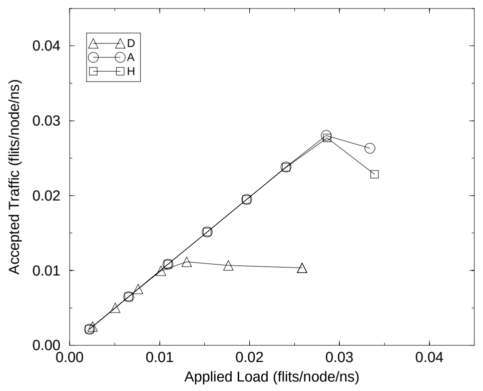


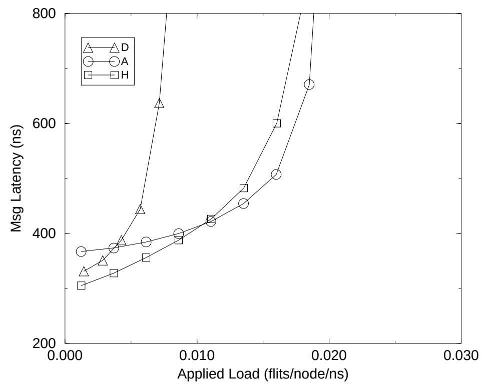


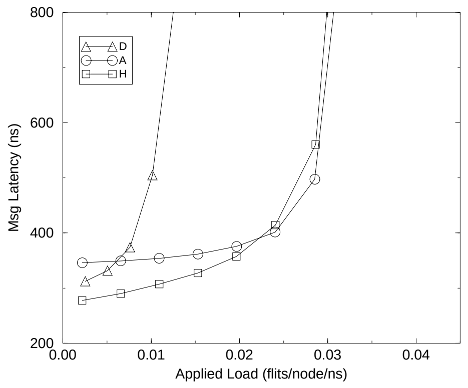


**Complement Traffic**









**Perfect Shuffle Traffic**

Figure 8: Comparison of pipelined and super-pipelined implementations of deterministic, adaptive and hybrid for 8-ary 3-cube (L=16) with *minimal* cycle times

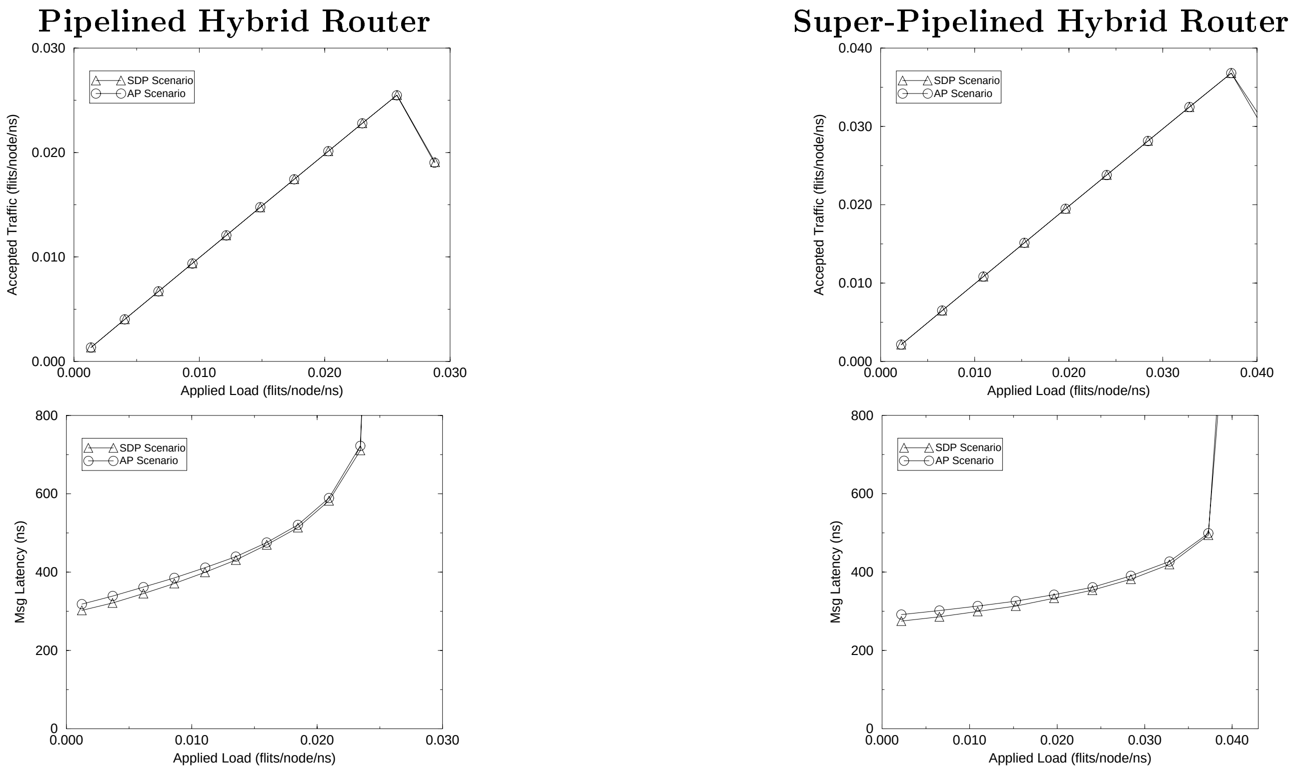


Figure 9: Comparison of pipelined and super-pipelined implementations of two different path options for hybrid routing on an 8-ary 3-cube (L=16) for random uniform traffic

The results are shown in Figure 9 only for random uniform traffic since all other traffic patterns result in similar performance. For all traffic patterns, the difference between the Slow Deterministic Path and Adaptive Path scenarios is quite small. However, the Slow Deterministic Path scenario does perform slightly better than the Adaptive Path scenario because of the priority given to the Fast Deterministic Path. Since the Fast Deterministic Path has the highest priority, it is more beneficial to keep messages on the deterministic channels so that the Fast Deterministic Path path can be utilized more often. Since the Slow Deterministic Path scenario does just this, it outweighs the advantage of retaining adaptivity in the Adaptive Path scenario.

## 5 Related Work

Deterministic routing algorithms are very common in commercial multicomputers, such as the Intel Paragon [21], Cray T3D [23], and NCUBE-2/3 [29]. This type of routing is also prevalent in many experimental multicomputers, such as the Stanford DASH [26] and MIT J-Machine [31]. This routing type is so popular because of its simplicity both in concept and in implementation. Under deterministic routing a message always uses the same path between each source and destination node and this path is determined as a function of the destination address.

One of the simplest and most common types of deterministic routing is called dimension-order routing. In this type of routing a message is routed along decreasing dimensions with a dimension decrease occurring only when zero hops remain in all higher dimensions. Virtual channels are included in the router to avoid deadlock [8].

Besides deterministic routing, partially adaptive routing algorithms also exist. In this type of routing, the adaptivity of a message is increased with little resource addition. An example of this routing type is called planar-adaptive routing and is proposed for $n$-dimensional meshes and hypercubes [5]. In this type of routing, a packet is only routed in two dimensions at a time. In order to prevent deadlock, an ordering is imposed. For example, a message is first routed in plane $A_0$, then plane $A_1$, and so on. Since at each clock cycle, adaptivity is limited to two dimensions, resources are minimized while gaining adaptivity.

The Turn Model proposed by Chien and Ni [19] is another example of a partially adaptive routing algorithm. In this algorithm deadlock freedom is accomplished by preventing turns and does not require virtual channels. Once again adaptivity is gained with a minimal increase in resources.

Deterministic and partially adaptive algorithms are relatively simple to implement. Unfortunately, these routing algorithms can suffer from congestion since only a subset of all possible paths between a source and destination are used. Deterministic algorithms are particularly vulnerable to poor performance at high traffic since no network information is used in the routing decision. Adaptive routing, on the other hand, has many alternative choices for a message as well as uses information about the status of the network.

In adaptive routing, messages are not restricted to a single path when traveling from source to destination. Moreover, the choice of path can be made dynamically in response to current network conditions. Such schemes are more flexible, can minimize unnecessary waiting, and can provide fault-tolerance. Several studies have demonstrated that adaptive routing can achieve a lower latency, for the same load, than deterministic routing when measured by a constant clock cycle for both routers [25].

The Chaos router [24] is an example of an adaptive router. It has buffers associated with each input and output port of a node, as well as centralized buffers which are used for deadlock prevention. This router performed particularly well when hot-spots were present due to its ability to retain adaptivity as well as to efficiently use network status information.

A class of adaptive routing algorithms is know as the hop algorithm class [20]. In this type of algorithm, deadlock is avoided by splitting buffers into several classes. The packets can then only be moved from one buffer to another in such a way that buffer class is never decremented. This algorithm can be used for any topology. However, the number of buffers required is very high and depends on the network size.

Many researchers also tried avoiding deadlock in adaptive routers using virtual channels. Unfortunately, many of the early models required a large number of virtual channels. In [27], the concept of virtual channels was extended to that of virtual networks. In this strategy, a single network was logically split into several layers of networks, and messages moved through succeeding layers as they traversed the network. Adaptivity was allowed within a layer, and thus a message could choose among a variety of paths. The drawback of this system, however, is that the number of virtual channels required by the model grows exponentially with the size of the network. Even if these are not separate physical channels, the buffer space and control logic required to manage such a system quickly makes the model infeasible.

Because of the effect router complexity has on clock cycle time, issues concerning this topic needed to be researched. Some of the earliest work in this area involved the torus routing chip [9]. In this chip deadlock-free worm-hole routing techniques were used along with an implementation based on a full crossbar architecture. Later implementations greatly increased router performance by taking advantage of dimension-order routing's uniformity. This was accomplished by subdividing the full crossbar into a set of smaller crossbars, as well as subdividing all other regular logic [7, 17, 34]. However, all of these have only focused on dimension-order routing.

A number of commercial worm-hole routers have been implemented for the Symult 2010 [33] and the Intel Paragon [21, 16]. The Paragon router also uses the 0.8 micron CMOS gate array technology. It gives comparable performance to those routers found in [4] with a delay of 40 nanoseconds and channel bandwidth of 200 megabytes/second.

The Triplex routing algorithm is an example of a multi-class routing algorithm in which the dynamic selection of oblivious, minimal fully adaptive, and non-minimal fully adaptive routing is possible [18]. The Cray T3E router is also a hybrid router: Messages can be routed deterministically or adaptively simply by setting a bit in the header [32]. The router also supports a shortcut for messages that continue traveling in the same dimension and uses direction-order routing for its deterministic routing algorithm.

Comparisons of adaptive and deterministic router implementations, for worm-hole routing, are described in [1, 4] and [13]. However, the comparison in [1, 4] does not account for the reduced queuing delay in adaptive routing. In [13] the reduction in queuing delay for worm-hole routing is taken into account and the comparison is based on a constant total buffer area.

The architectural support for the reduction of communication overhead is described in [10]. This scheme exploits the communication locality in message passing programs to distinguish between cacheable and non-cacheable virtual channels. Cacheable virtual channels are retained for multiple messages thereby allowing an overlap of communication and computation and eliminating the overhead of multiple message set-up. This mechanism is a hybrid scheme combining circuit and worm-hole switching. The implementation of a router supporting this scheme is described in [14]. Its routing properties are discussed in [15].

## 6 Conclusions

This paper reports on the empirical evaluation of a hybrid routing scheme which combines the low router delay of deterministic routing with the flexibility and low queuing delays of adaptive routing. This hybrid routing mechanism is realized using two different implementations (Pipelined Hybrid Router and Super-Pipelined Hybrid Router) in which different paths and stages of the router are used for different routing modes. The scheme also relies on making the "common case fast" and is similar in concept to the hot potato algorithm.

The results from the simulation evaluation of this scheme show that both implementations of the hybrid router do achieve their objectives: a message latency comparable to that of the deterministic router at low traffic and a saturation point close to that of the adaptive router at high traffic. In addition, deeper pipelines achieve better overall performance gain than the pipelined implementations, although the improvement is minimal with deterministic routers.

## References

[1] K. Aoyama and A. Chien. The cost of adaptivity and virtual lanes in wormhole router. *J. of VLSI Design*, 2(4), 1995.

[2] P. Baran. On distributed communication networks. *IEEE Trans. on Commun. Systems*, CS-12:1–9, March 1964.

[3] P. Berman, L. Gravano, G. Pifarre, and J. Sanz. Adaptive deadlock and livelock free routing with all minimal paths in torus networks. In *Proc. of the Symp. on Parallel Algorithms and Architectures*, pages 3–12, 1992.

[4] A. Chien. A cost and speed model for $k$-ary $n$-cube wormhole routers. In *IEEE Proc. of Hot Interconnects*, Aug. 1993.

[5] A. A. Chien and J. H. Kim. Planar-adaptive routing: low cost adaptive networks for multiprocessors. *Int. Symp. on Computer Architecture*, pages 268–277, May 1992.

[6] W. Dally and et al A. Chien. The J-Machine: a fine-grain concurrent computer. In *Proc. of the IFIP Congress*, pages 1147–1153, Aug. 1989.

[7] W. Dally and P. Song. Design of a self-timed VLSI multicomputer communicaton controller. In *Proc. of the Int. Conf. on Computer Design*, pages 230–40, 1987.

[8] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Computers*, 3(2):194–205, March 1992.

[9] W. J. Dally and C. L. Seitz. The torus routing chip. *J. Dist. Computing*, 1(3):187–196, 1986.

[10] B. Dao, S. Yalamanchili, and J. Duato. Architectural support for reducing communication overhead in multiprocessor interconnection networks. In *High Performance Computer Architecture*, pages 343–52, 1997.

[11] 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.

[12] J. Duato. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Trans. on Parallel and Distributed Systems*, pages 841–854, 1996.

[13] J. Duato and P. Lopez. Performance evaluation of adaptive routing algorithms for k-ary n-cubes. In *Parallel Computer Routing and Communication*, pages 45–59, 1994.

[14] J. Duato, P. Lopez, F. Silla, and S. Yalamanchili. A high performance router architecture for interconnection networks. In *Int. Conf. on Parallel Processing*, August 1996.

[15] J. Duato, P. Lopez, and S. Yalamanchili. Deadlock- and livelock-free routing protocols for wave switching. In *Int. Parallel Processing Symp.*, April 1997.

[16] D. Dunning. The intel paragon router. Master's thesis, Massachusetts Institute of Tehnology, 1992.

[17] C. Flaig. VLSI mesh routing systems. Master's thesis, California Institute of Tehnology, May 1987.

[18] M. Fulgham and L. Snyder. Integrated multi-class routing. In *Proceedings of the Workshop on Parallel Computer Routing and Communication*, 1997.

[19] C. L. Glass and L. M. Ni. The turn model for adaptive routing. In *Int. Symp. on Computer Architecture*, pages 278–287, May 1992.

[20] I. Gopal. Prevention of store-and-forward deadlock in computer networks. *ieeetc*, 33(12):1258–64, December 1985.

[21] Intel Corp. *Paragon XP/S Product Overview*, 1991.

[22] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267 – 286, 1979.

[23] R. Kessler and J. Schwarzmeier. CRAY T3D: a new dimension for cray research. *Compcon*, pages 176–82, 1993.

[24] S. Konstantinidou and L. Snyder. Chaos router: architecture and performance. *Int. Symp. on Computer Architecture*, pages 212–221, 1991.

[25] Annette Lagman. *Modelling, Analysis and Evaluation of Adaptive Routing Strategies.* PhD thesis, Colorado State University, Computer Science Department, November 1994.

[26] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam. The Stanford DASH multiprocessor. *IEEE Computer*, 25(3):63–79, March 1992.

[27] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. on Computers*, 40(1):2–12, January 1991.

[28] D. Miller and W. Najjar. Preliminary evaluation of a hybrid deterministic/adaptive router. In *Parallel Computer Routing and Communication*, 1997.

[29] NCUBE Company. *NCUBE 6400 Processor Manual*, 1990.

[30] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, pages 62–76, 1993.

[31] M. Noakes, D. Wallach, and W. Dally. The J-Machine multicomputer: An architectural evaluation. In *20th International Symposium on Computer Architecture*, May 1993.

[32] S. Scott and G. Thorson. The Cray T3E networks: adaptive routing in a high performance 3d torus. In *Proceedings of Hot Interconnects IV*, August 1996.

[33] C. Seitz, W. Athas, C. Flaig, A. Martin, J. Seizovic, C. Steele, and W. Su. The architecture and programming of the ametek series 2010 multicomputer. In *Third Conference on Hypercube Computers*, pages 33–6, 1988.

[34] C. Seitz and W. Su. A family of routing and communication chips based on the mosaic. In *Proc. of the University of Washington Symp. on Integrated Systems*, 1993.