Combining Adaptive and Deterministic Routing: Evaluation of a Hybrid Router

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

A novel routing scheme is proposed for virtual cut-through routing that attempts to combine the low routing delay of deterministic routing with the flexibility and low queuing delays of adaptive routing on k-ary n-cube networks. 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 traditional 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: deterministic routing, adaptive routing, virtual cut-through switching, pipelined router, router complexity.

1 Introduction

This paper reports on the implementation and evaluation of a hybrid routing scheme that combines the advantages of deterministic and adaptive routing. An expanded version of this paper can be found in [1]

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 (VCs) are included in the router to avoid deadlock [6]. 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 and the choice of path can be made dynamically in response to current network conditions. Such schemes are more flexible, minimize unnecessary waiting, and 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 [12, 14].

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 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.

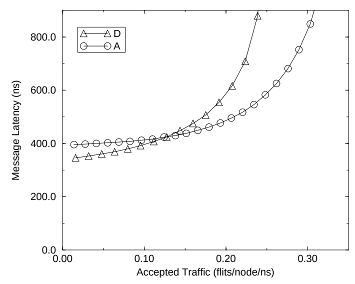


Figure 1: Message latency of deterministic (D) and adaptive (A) routing on a 10-ary 3-cube network under random uniform traffic and with message length of 8 flits.

However, the router delay for deterministic routers, and consequently their corresponding clock cycles, can be significantly lower than adaptive routers [2, 4]. This difference in router delays is due to two main reasons: number of VCs and output (OP) channel selection. Two VCs are sufficient to avoid deadlock in dimension ordered routing [6]; while adaptive routing (as described in [8, 3]) requires a minimum of three VCs in k-ary n-cube networks. In dimension-ordered routing, the OP channel selection policy only depends on information contained in the message header itself. In adaptive routing the OP channel selection policy depends also on the state of the router (i.e the occupancy of various VCs) causing increased router complexity and higher router delays.

The results reported in [2, 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 is evaluated and reported in [9] for worm-hole routing. A typical comparison of deterministic versus adaptive routing message latencies (accounting for the differences in cycle times) is shown in Figure 1: at low traffic and for short to moderate message sizes, the latency of deterministic routing is smaller [9, 15]. 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: 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. Concluding remarks are given in Section 6.

While the work described in this paper relates to a k-ary n-cube, the concepts and router architecture can easily be extended to other topologies. These results are valid for networks designed for chip or multi-machine level implementations (NOWs).

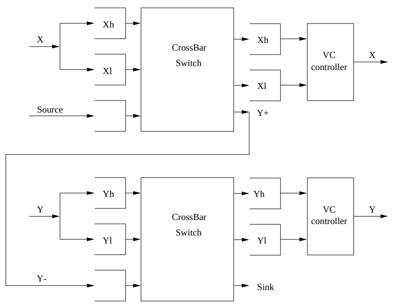
2 Deterministic and Adaptive Routing

The interconnection network model considered in this study is a k-ary n-cube using virtual cut-through switching [13]: message advancement is similar to worm-hole routing [16], 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 message lengths are equal.

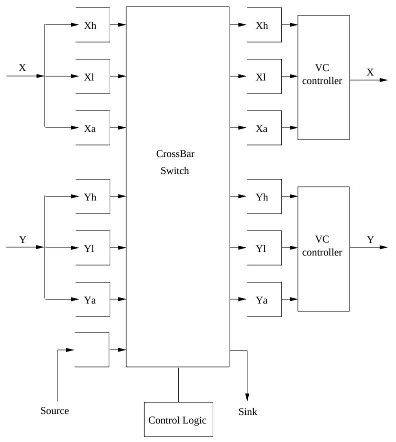
2.1 Routing Models

In the *deterministic* routing scheme (dimension-order routing) [6], 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 [8, 3]. In this algorithm, adaptive routing is obtained by using adaptive VCs 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 message size is greater than the buffer size (i.e. size of the the VC) and as long as a message's header flit is allowed 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, deadlock is prevented



(a) Deterministic Router



(b) Adaptive Router

Figure 2: Schematics of deterministic and adaptive 2D routers

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 VCs per dimension per node for each physical unidirectional channel. Therefore, the number of VCs grows linearly with network size.

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 VCs of each dimension are multiplexed onto one PC. 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 OP channels, while the adaptive router uses storage buffers associated with input (IP) channels. When using OP 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 IP 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 OP channels. The adaptive router implements a round-robin IP message selection policy which checks for messages first among all adaptive buffers and then among all deterministic buffers.

OP 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 OP channel selection policy, the greatest amount of adaptivity 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, 2, 9]. 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 VCs that are multiplexed on one PC. These models were modified to account for the varying buffer space used in virtual cut-through switching. The parameters of these models are as follows:

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 these possible routes is included in the routing arbitration delay (T_{ARB}). The crossbar delay (T_{CB}) is the time for data to go through the switch's crossbar. It is usually implemented with a tree of gates. The flow control delay (T_{FC}) includes the time for flow control between routers to prevent buffer overflow. T_{SEL} is the time to select the appropriate header. Finally, the VC controller delay (T_{VC}) includes the delay for multiplexing VCs onto PCs.

For all dimension-order routers simulated here, the number of degrees of freedom (F) equals one since there exists one 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 select among all adaptive channels as well as among the appropriate deterministic channels. 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. The three main operations (delays) prevalent in all of the routers simulated here are as follows: T_r is the time to route a message, T_s is the time to transfer a flit to the corresponding OP channel, and T_c is the time to transfer a flit across a PC. The equations are:

$$T_r = T_{AD} + T_{ARB} + T_{SEL} = 4.7 + 1.2log_2F$$
$$T_s = T_{FC} + T_{CB} + T_{Latch} = 2.0 + 0.6log_2B + 0.6log_2P$$

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

Table 1: **a- Deterministic router delays ($C = 2$, $P = 3$, $F = 1$) for k-ary 3-cube networks (all values in $nsec$)**

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

Table 2: **b- Adaptive router delays ($C = 3$, $P = 10$, $F = 6$) for k-ary 3-cube networks (all values in $nsec$)**

$$T_c = 4.9 + T_{VC} = 6.14 + 0.6log_2C$$

Using the above equations, the delay values were calculated for each of the router algorithms simulated and are shown in Tables 1 and 2. It is assumed that all three operations are overlapped through pipelining as described in [9], and therefore the clock period is determined by the longest delay: $T_{ccperiod} = Max(T_r, T_s, T_c)$

From the data in Tables 1 and 2, we observe that increasing the buffer size in deterministic and adaptive routers, increases the overall router delay only when large buffer sizes are used. In deterministic routers, 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 cycle time is dominated by T_r for small and moderate buffer sizes and dominated by T_s for large buffer sizes.

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 [2] where 15% to 60% improvement is required for f-flat routers with similar number of VCs 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 (PHR) and a super-pipelined hybrid router (S-PHR).

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)[1]. 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 FDP 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 reduce adaptivity, 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 FDP (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 SDP which requires more logic and more stages than the FDP.

The AP is used to adaptively route a message and has the lowest priority. It is only used when both the FDP and SDP are unavailable. Both the SDP and AP take $H + D$ cycles for a header flit and D cycles for a data flit, where $(H + D) > (h + d)$. 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.

This routing scheme is deadlock free: for any given message, the selection of paths is always a true subset of those that could be selected by the adaptive algorithm in [8]. Since the adaptive algorithm has been proven deadlock free, the hybrid is also deadlock free.

3.2 Pipelined Implementations

The *Pipelined Hybrid Router* (PHR) 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 FDP and a 3/2 stage pipeline for the SDP and AP. Because the routing decision and switching logic of the FDP 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 SDP and AP 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 SDP and AP and all paths share the VC control logic.

The *Super-Pipelined Hybrid Router* (S-PHR) 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 PHR, the work in each stage of the PHR is divided into two stages in the S-PHR. Therefore, the 2/1 stage pipeline in the FDP of the PHR becomes a 4/2 stage pipeline in the S-PHR, while the SDP and AP paths are modified from 3/2 stage pipelines to 6/4 stage pipelines. Once again, the crossbar is physically shared between both SDP and AP and all paths share the VC control logic. Figure 5 shows the new schematic for this super-pipeline. Note that the main difference between the PHR and S-PHR is the number of stages required for each path. The flow chart in Figure 3 is used in the S-PHR where $h = d = H = 2$ and $D = 4$.

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

3.3 Clock Cycle Times

The performance of the pipelined and super-pipelined implementations of the hybrid router is compared to the corresponding 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 cycle times for both are obtained using the equations in Section 2.3. Note that the deterministic router is not implemented using a 2/1 stage pipeline as in the hybrid router. This is because to accommodate both the routing and switching delays for the Fast Deterministic Path into one stage would require the deterministic router's cycle time to be comparable to that of the adaptive router's. This greater cycle time would offset any advantage gained from having fewer number of cycles. Simulation results supporting this conclusion can be found in [1].

In the more complex hybrid router, the cycle time for its 3/2 stage pipeline path is much larger than that for the 3/2 stage pipeline in the deterministic router. Therefore all the necessary logic in the FDP of the hybrid router can fit into a 2/1 stage pipeline implementation without greatly increasing the cycle time of its 3/2 stage pipeline paths (SDP and AP paths). Since the cycle time is not greatly increased by adding the 2/1 stage pipeline path (FDP), the advantage of fewer number of cycles is retained. The cycle time of the pipelined hybrid router (PHR) is simply one gate delay larger than that of the adaptive router to account for the increased critical path length due to the inclusion of the 2/1 stage pipeline path (FDP).

Super-Pipelined Router Implementation. The super-pipelined implementation for all routers consists of dividing the work in each stage of its corresponding pipelined implementation into two. This results in a 6/4 stage super-pipeline for the adaptive router and a 4/2 stage super-pipeline for the FDP of the hybrid router and a 6/4 stage super-pipeline for the SDP and AP.

The deterministic router's super-pipelined implementation is also implemented using the FDP and SDP. This is because, unlike its pipelined router implementation, having a fast path with fewer cycles does benefit the super-pipelined deterministic router. This results because channel

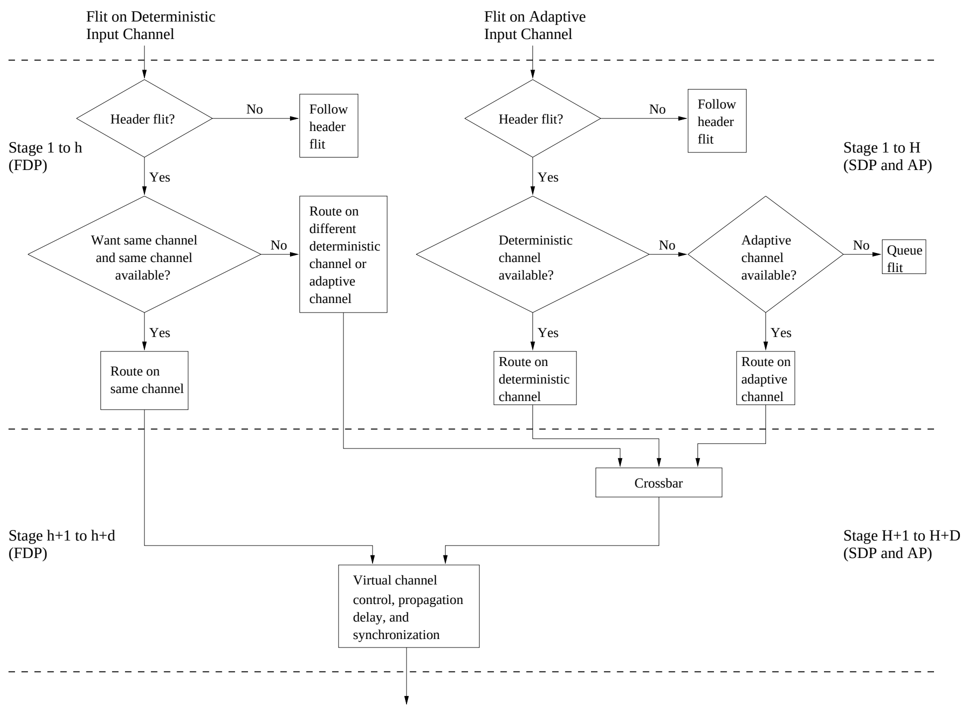

Flit on Deterministic Input Channel
Flit on Adaptive Input Channel
Stage 1 to h (FDP)
Stage 1 to H (SDP and AP)
Header flit?
No
Yes
Follow header flit
Want same channel and same channel available?
Route on different deterministic channel or adaptive channel
Deterministic channel available?
Adaptive channel available?
Queue flit
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

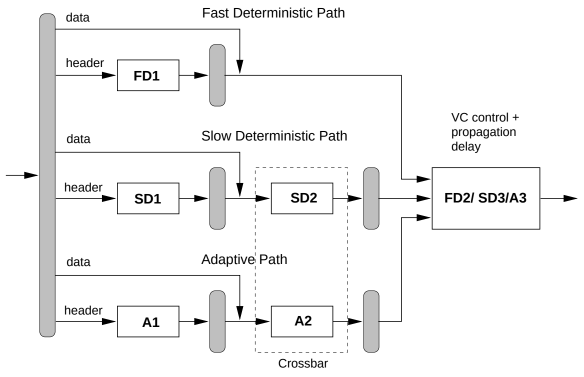


Figure 4: Logic schematic of PHR

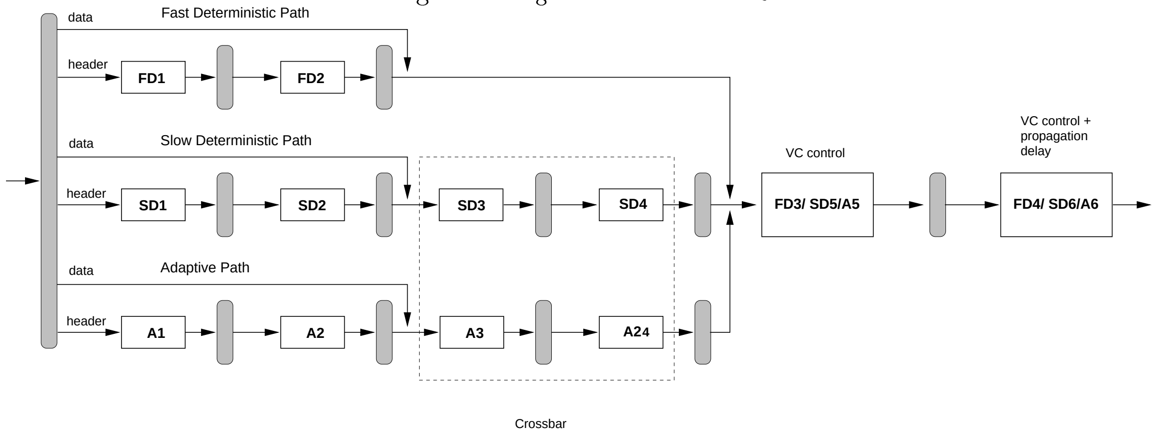


Figure 5: Logic schematic of S-PHR

	Deterministic		Adaptive		Hybrid	
B	PR	$S-PR$	PR	$S-PR$	PR	$S-PR$
8	6.74	4.90	7.80	4.90	8.40	5.00
16	6.74	4.90	7.80	4.90	8.40	5.00
32	6.74	4.90	7.80	4.90	8.40	5.00

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

delay is the bottleneck in deterministic routers. In the super-pipelined implementation, the delay required for transferring a flit across the physical wire (4.9 ns) is the longest delay among all stages and therefore equals the cycle time for the super-pipelined deterministic router. This cycle time is now large enough to accommodate all the deterministic router complexity in a 4/2 stage pipeline.

The T_r and T_s delays for the *super-pipelined* implementation of all routers were calculated using their corresponding *pipelined* delays in Equation 1.

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

The T_r and T_s delays for the super-pipelined implementation are represented as T_{S-PR} in Equation 1, while their corresponding *pipelined* router (PR) delays are represented by T_{PR}. The setup time for the latch (L) is 0.8 ns and the delay for one gate (G) is 0.6 ns.

The super-pipelined T_r and T_s delays 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.

Since the T_c delay consists of a set of gates as well as a wire, this case is considered separately. The two stages for the super-pipelined implementation of this delay consist of the VC controller delay ($T_{vc_{S-PR}} = 1.24 + 0.6log_2C$) in one stage and the propagation delay which is required for transferring a flit across the physical wire ($T_{p_{S-PR}} = 4.9$) in the other stage.

The cycle time for the super-pipelined router (CC_{S-PR}) is then determined by the longest delay among all super-pipelined stages where $T_{r_{S-PR}}$ and $T_{s_{S-PR}}$ represent the T_r and T_s super-pipelined implementation delays obtained from Equation 1:

$$CC_{S-PR} = Max(T_{r_{S-PR}}, T_{s_{S-PR}}, T_{vc_{S-PR}}, T_{p_{S-PR}}) \qquad (2)$$

The cycle times for the pipelined and super-pipelined implementations of the three routers are shown in Table 3.

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 VCs 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. The following five different traffic patterns were simulated: random uniform complement, perfect shuffle, bit-reversal, and butterfly.

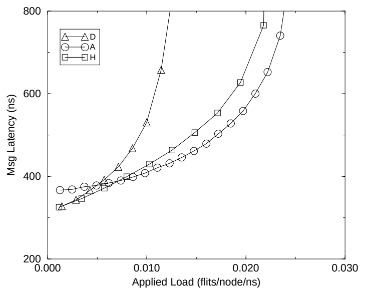


Figure 6: Message latencies of deterministic, adaptive and hybrid pipelined implementation routers in an 8-ary 3-cube under random uniform traffic with L=16

4.1 Performance of Hybrid Routing

Figure 6 shows the message latency versus offered load plots of the deterministic, adaptive and hybrid pipelined implementations under random uniform traffic with message length of 16 flits. Figure 7 shows similar plots for complement traffic which has representative behavior of most traffic patterns simulated. An expanded version of this paper with complete results can be found in [1].

Message Latency. Under random uniform traffic, for small messages (8 flits) the latency of the PHR is not only lower than the pipelined adaptive one but is also lower than the pipelined deterministic one at low traffic. This is due to the fact that the PHR 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 PHR's, the greater number of stages dominates. For medium messages (16 flits) the latency of the PHR 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 PHR 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 PHR follows the deterministic one at low traffic and the adaptive one at high traffic.

Effects of Message Length. As message size increases under random uniform traffic, the performance advantage of the PHR 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 PHR 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 cycle time, the PHR has a 2-stage deterministic header flit pipeline with a higher cycle time. Since the number of pipeline stages dominates performance (and not the 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.

Effects of Traffic Patterns. The performance of the PHR under all non-random traffic patterns is similar to that for random uniform traffic. Once again, the PHR 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 PHR: less choices are available as a message is routed through the network on deterministic channels.

Saturation Point. Under random uniform traffic, the saturation point of the PHR 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 PHR 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 the PHR'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. Because of the higher throughput, all super-pipelined routers achieve higher saturation points than the pipelined implementations.

4.3 Effects of Path Priorities

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

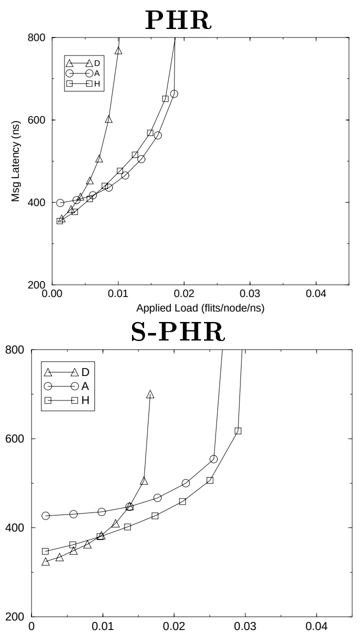


Figure 7: Comparison of pipelined and super-pipelined implementations of deterministic, adaptive and hybrid for 8-ary 3-cube (L=16) under complement traffic

However, the AP is given the next highest priority, followed by the SDP. This scenario is called the AP scenario.

The simulated results between these two scenarios for all traffic patterns are so close that the resulting graphs will not be shown here. However, such close results demonstrate that although the AP scenario may allow more routing choices as load increases, the SDP scenario performs equally well because of the high priority and low cycle time of the FDP. Since the FDP has the highest priority, the benefit of retaining messages on deterministic channels allows the FDP path to be utilized more often and offsets any adaptivity loss.

5 Related Work

Some of the earliest work in understanding the effects of router complexity on cycle time involved deterministic routers [7, 5, 10]. Adaptive and deterministic router implementations were then compared for worm-hole routing [2, 4, 9]. However, the comparison in [2, 4] does not account for the reduced queuing delay in adaptive routing. In [9] 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 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 [11]. The Cray T3E router is also a hybrid router. Messages can be routed deterministically or adaptively by simply setting a bit in the header [17]. The router supports a shortcut for messages that continue traveling in the same dimension and uses directionorder routing for its deterministic routing algorithm. It also implements a routing function that bases the VC selection on the current VC and destination and implements a VC optimization scheme for VC balancing.

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 (PHR and S-PHR) 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.

References

[1] `http://www.colostate.edu/ ~najjar/papers/hybrid.pdf`.

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

[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] 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.

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

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

[8] 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.

[9] 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.

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

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

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

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

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

[15] D. Miller and W. Najjar. Empirical evaluation of deterministic and adaptive routing with constant-area routers. In *Parallel Architecture and Compiler Techniques*, 1997.

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

[17] 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.