The Effect of Flow Control and Routing Adaptivity on Priority-Driven Traffic in Multiprocessor Networks

Shobana Balakrishnan and Füsun Özgüner

Dept. of Electrical Engineering, The Ohio State University,
Columbus OH 43210-1272, USA

Abstract. We study the impact of two flow control schemes and routing adaptivity on the performance of priority-driven traffic produced by real-time applications. The first is wormhole routing (WR) with real-time extensions [1] and the second is preemptive pipelined circuit switching (PPCS-RT) [2]. Our simulations show that for a fixed number of virtual channels (VCs)/link, parallel lanes are more effective than adaptive routing alone, in reducing the number of messages that miss their deadlines. PPCS-RT performs better than WR due to the VC preemption protocol supported by it.

1 Introduction

Wormhole routing (WR) [3] is a flit-buffered flow control scheme increasingly used in multiprocessor point-to-point multi-hop networks, since it provides the design for low latency networks, and high link bandwidth, at low cost. The flit buffers with associated control are referred to as virtual channels (VCs). The commonly used routing algorithm in wormhole routers is the dimension order routing algorithm [3], which is an oblivious shortest path routing scheme. More recently, a number of adaptive routing algorithms have been proposed [3]. Here, the routing algorithm provides a choice of routing options and a selection function selects one of the options for routing the header. In order to prevent deadlocks, the routing algorithm uses routing classes (also called virtual networks), implemented by multiple VCs that are demand multiplexed on the link, with restrictions on the use of the classes [4]. For example, a simple minimal fully adaptive routing algorithm for meshes proposed in [4] requires two routing classes. Multiple VCs without any routing restrictions called lanes, were also proposed in [5] to improve the utilization of the physical link bandwidth. The speed advantage of WR using dimension order routing with a single lane, that requires only one VC/link, and the cost associated with multiple VCs, has resulted in the prominence of parallel systems with a modest number of VCs/link.

Real-time applications, unlike regular applications, produce tasks that must be performed before their associated deadlines. These tasks typically read sensor data and process it periodically. Multiprocessor systems are likely candidates for running real-time applications, since they offer increased computational power and fault tolerance, at good price points. Real-time applications produce messages, where a real-time message M_i is characterized by three parameters; namely period (p_i), deadline (d_i), and size (s_i). Real-time messages are classified as *hard* (also called guarantee seeking) and *soft* deadline (also called best effort) messages. For a hard deadline message, it is imperative that the communication model guarantees its timely delivery, while for a soft deadline message, timely delivery is a desirable feature, although occasional deadline misses can be tolerated. Real-time messages have an associated *global priority* that determines the importance level of the message. For example, when the message with the earliest deadline is assigned the highest priority, the assignment scheme is called Deadline Monotonic Scheduling (DMS) [6]. We shall use this priority assignment scheme in this paper.

With priority-driven real-time traffic, the priority is used to resolve the competition among messages for shared network resources. In WR, there are two types of shared resources, namely the VCs and the physical link. A WR message can compete with any of the messages that share at least one hop on the path of the message, for the use of the resources. Hence, the global priority order among all competing messages must be enforced by the communication model, for predictability. With a limited number of parallel lanes, a global priority ordering on the usage of the VCs during the course of the application is desirable. Two real-time extensions to WR have been proposed in [1], namely, priority based VC allocation and link arbitration. With priority-based VC allocation, the highest priority message amongst the competing messages *at that time* is allocated the output VC at a router. Local priority order on the usage of the link is enforced by priority based link bandwidth arbitration, instead of the conventional demand multiplexing, whereby, the physical link is in effect *locally preemptable.* A higher priority header arriving at an input VC whose routing request for an output VC is granted, can preempt a lower priority message using the link. A higher priority message, if not granted an output VC due to unavailability, cannot in any way affect lower priority messages occupying the output VCs, since a VC once allocated can only be released by the tail flit of the message. Hence, depending on the order of arrival of messages, a higher priority message may be blocked by a lower priority message using a VC, for an *unbounded* amount of time. Such a situation is referred to as *priority inversion*, which was first investigated in the context of task scheduling. The priority inversion is unbounded, because in WR, the message delivery time cannot be bounded due to the fact that the header may be blocked at each hop, depending on the contending messages. The priority inversion problem is magnified with fewer lanes. Such a WR model with v lanes, can only enforce a v level priority order, although the priority levels of the competing messages may be more fine grained. In this paper, we compare a new flit-buffered communication model that we recently proposed [2] called *Preemptive Pipelined Circuit Switching for Real-Time* (PPCS-RT) messages with WR [1]. Both models employ parallel lanes as well as deadlock-free routing. We evaluate the effectiveness of v parallel lanes under dimension order routing. We also consider the effect of routing adaptivity *i.e.*, WR using multiple VCs for adaptive routing classes as well as parallel lanes.

In terms of alternative flow control schemes to WR, Pipelined Circuit Switching (PCS) was proposed by Gaughan and Yalamanchili [7] for achieving fault tolerance. PCS does not address the problem of real-time message scheduling. Unlike the nonblocking header in PCS, which requires an extra forward virtual channel, the header is blocking in PPCS-RT, and backtracks when *any* of the VCs acquired by it are requested by a higher priority message. A unified approach to real-time task and message scheduling has been adopted in [9] for a single hard deadline periodic input using a scheduled routing method. The SPIDER [8] network adapter supports mixed mode flow control, namely both packetized store-and-forward switching and WR, to cater to the conflicting requirements of hard and soft deadline messages, by partitioning the network into two separate virtual networks, one for hard deadline, and the other for soft deadline traffic. We believe that WR is a good choice in massively parallel systems when the application traffic does not have any deadline constraints. However, for soft and hard deadline traffic, where priorities of messages are fine grained such as on the basis of deadline, we argue that WR is inadequate. The rest of this paper is organized as follows. In Section 2 we describe the basic PPCS-RT model and the protocol for preemption. In Section 3 we compare the performance of PPCS-RT and WR with varying number of virtual channels for parallel lanes and deadlock-free routing.

2 A New Flow Control Model for Real-Time Messages

For completeness, we present in this section a brief description of the PPCS-RT flow control model proposed in [2, 10]. Additional implementation details may be found in [2]. The goal of PPCS-RT is to keep the advantages of wormhole routing such as small flit buffers (that result in single chip, fast, inexpensive routers) and prevent the unbounded priority inversion problem.

2.1 PPCS-RT

In WR, the delivery time of a message is unpredictable due to the blocking faced by the header. The basic idea of PPCS-RT is to enforce the global priority ordering on the usage of the VCs by supporting preemption during this unpredictable period when the header is in transit from the source to the destination. For this purpose, the message delivery is performed in two decoupled phases; Path Establishment (PE) and Data Delivery (DD). The header flit first establishes the path similar to circuit switching in the PE phase. If a header cannot receive the requested output VC, it blocks and holds the VCs acquired so far, as in WR. All VCs reserved by the header are preemptable in this phase, by higher priority messages. When a path is successfully established, an acknowledgment is returned to the source node, that terminates the PE phase and initiates the DD phase. Data flits are pipelined in the DD phase. In the DD phase, the VCs occupied by the data flits are not preemptable. However, the duration of the DD phase is bounded and dependent on the message size, and the link bandwidth only.

For each VC that transfers the flits in the forward direction (denoted as v_f), PPCS-RT requires a reverse VC to transmit control flits in the reverse direction (denoted as v_r). The forward and reverse VCs of adjacent nodes are multiplexed on the physical link. The header, data, and tail flits use the forward virtual channel v_f as in WR. A *success* flit is returned along the reverse path using virtual channel v_r to the source router, when the header reaches the destination successfully, which then starts the DD phase. A message whose priority is higher than the owner of a VC can request preemption, if the owner is in the PE phase. This is done by generating a non-blocking *preempt* flit at the preemption point (preempted output channel) which also travels along v_f in the forward direction, until it reaches the channel containing the header to be preempted. If the header flit is not blocked, a preempt flit may or may not be able to catch up with the header depending on the preemption point and the position of the header with respect to the destination. The preempt flit has a router cycle time that is lower than the header flit since it does not go through the routing logic at each router. If the header is close to reaching its destination and successfully completes the PE phase before the preempt flit reaches it, the success flit returning kills the preempt flit, thereby aborting the preemption request. In this case, the higher priority message must face a blocking time equal to the DD phase of the lower priority message, and contend for the VC in priority order again.

When a header is preempted, a *free* flit is produced which backtracks along a reverse path using v_r and frees all forward VCs held by the preempted message until the point of preemption. Note that all VCs on the path up to the preemption point are still held by the preempted message. The header, preempt, free, and success flits will be referred to as control flits. It was shown that PCS [7] suffers from larger no-blocking message latency (minimum latency) due to the overhead of path setup, compared to WR, which is also the case with PPCS-RT. The experiments conducted in [7] assumed 16 VCs/link when the effect of blocking due to VC unavailability is negligible. With fewer VCs, by resolving VC contention based on global priority rather than local priority as in WR, we will show in the next section, that PPCS-RT results in improved mean latency over WR for high priority traffic, despite the additional overhead of the PE phase.

2.2 Control Flit Format

Figure 1 illustrates the fields in a control flit. Two bits are used to encode the control flit function. Note that the tail flit is not considered here since its function is identical to WR. The header flit carries the address of the destination. The preempt flit carries the address of the node at which the preemption occurs. This information is recorded in a writable register associated with each channel along which the preempt flit is propagated downstream. The free flit uses this information to detect when to stop its reverse path. The free flit carries the address of the destination node, which is used by the control logic to generate a new header when the free flit reaches the preemption point, so that the preempted message can resume from the preemption point. Note that the header and free flits differ only in the first two bits. The success flit does not carry any address, and simply follows the reverse path up to the source node. The header and free flits also carry the priority of the message used to determine which channels are eligible for preemption by the routing logic.

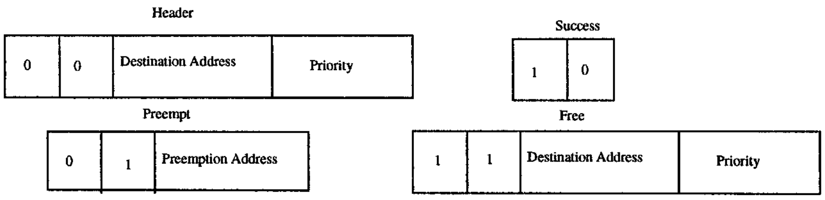


Fig. 1. Control flit format

2.3 Protocol for Virtual Channel Preemption

In order to support VC preemption, the state of the output virtual channels, and the priority of the owner must be maintained for each VC. A VC is in the PE state when the message owning it is in the PE phase. Similarly, a VC is in the DD state when the message owning it is in the DD phase. A VC is in the preempt state (PT) when a preemption request has been propagated on that VC. A VC, if available, is in the AV state. All examples are based on a 2D mesh, and one VC/link is assumed for simplicity. We first illustrate the ***header-success*** paths for a message originating at node A and destined to node D in Figure 2 (a), using the bidirectional links at each router. The actual flow of header and success flits is shown in Figures 2(b) and (c) respectively. At each node we show the input virtual channel, the state of the output virtual channel, and the crossbar connection made. Details of the implementation of the forward and reverse paths are discussed in [2]. The crossbar is a VC to VC crossbar. In this example, the crossbar is a 5x5 crossbar with inputs numbered xi and outputs numbered xo, where x denotes the directions $0 - 3$ or the node N.

The header flit shown in Figure 2(b) is responsible for switching the output virtual channel state to PE. The success flit generated at node D returns on the reverse VC path until it reaches the source node A as shown in Figure 2(c). A success flit arriving on an input channel j switches the state of output channel j to DD. At node A, the success flit is propagated to the reverse node channel No where it terminates, and the DD phase is started thereafter. Although the header and success flits are shown to exist in several VC FIFOs in the figure, only a single flit exists in any FIFO of a path, at any given time.

In Figure 3(a) we illustrate the same message from A to D, whose header is blocked at node C due to the fact that the output channel $3o$ is in DD state. While the header is blocked, a higher priority message, occupying channel $0i$ at

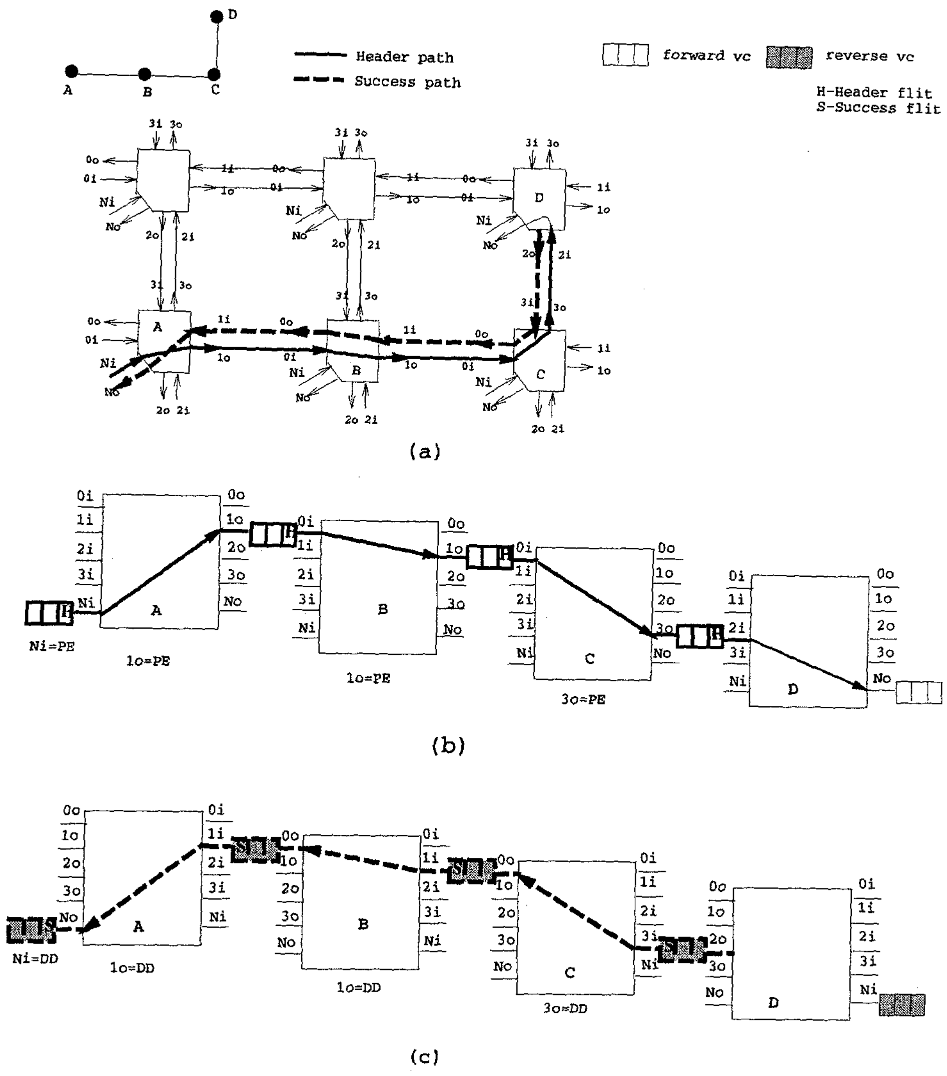


Fig. 2. PE phase (a) Header-Success paths (b) Header flit path (c) Success flit path

node A (Figure 3(b)) generates a preemption request for output channel $1o$. A preemption request is generated if the output channel to be preempted is in th PE state, and the owner priority is lower than the requesting message priorit The preempt flit propagated on an output channel j, changes the state of th channel to PT. The node address at which the preemption request is generatec must be recorded at each preempted channel for future preemption requests, a well as for the returning free flit to determine if it has reached the preemptio point. For this purpose, a writable register *Paddr[j]* associated with each outpı

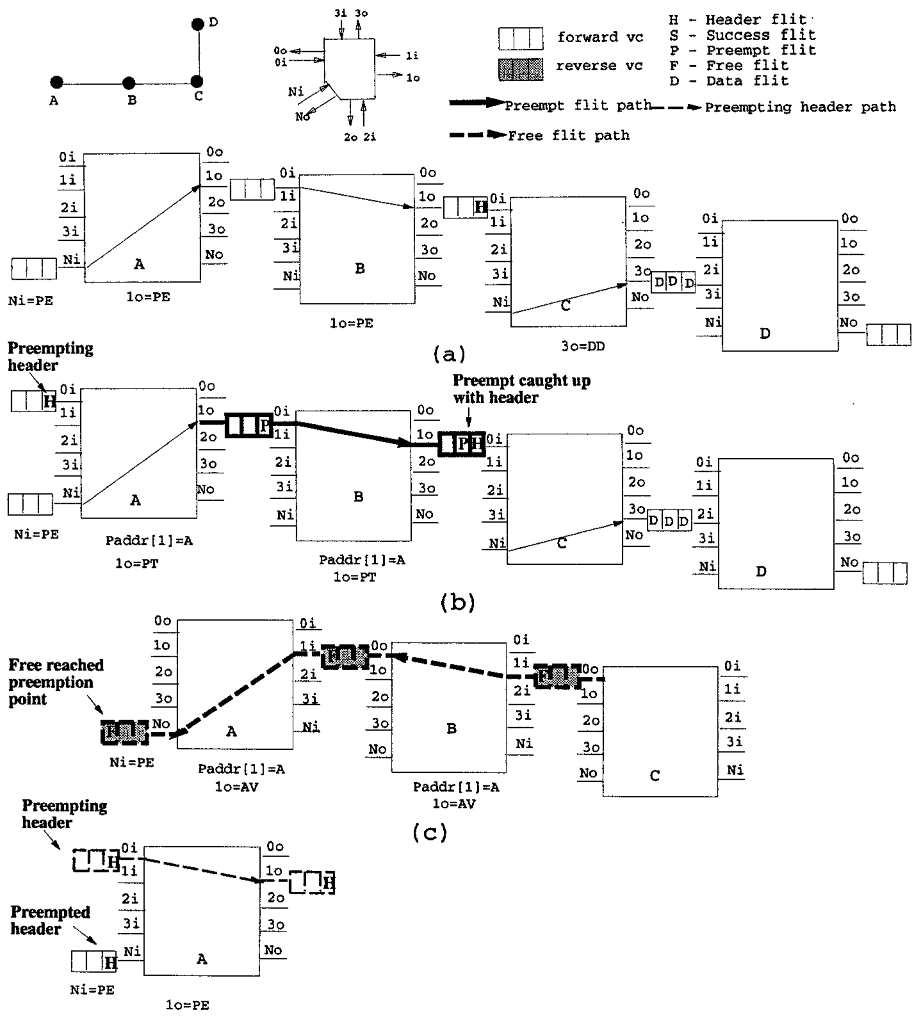


Fig. 3. Preempt-free flit paths (a) Header is blocked at node C (b) Preempt flit path (c) Free flit path (d) Preempting header path

VC j is required. A preempt flit that is propagated on an output VC j, updates the *Paddr[j]* register to the preemption address carried by it. In this example, at nodes A and B, *Paddr[1]* is set to the address of node A.

A free flit is generated when the preempt flit reaches node C, where the preempted header is blocked. The free flit shown in Figure 3(c), traverses the reverse path from node C to node A. At each hop, the free flit is propagated upstream if the preemption point is not reached. The free flit arriving at input channel 1 at node A terminates. The free flit arriving on an input channel j, changes the state of the output channel j to AV. At the preemption point, if the channel preempted is not an injection channel, a new header is generated by converting the free flit to a header flit, which occupies the channel given by the forward crossbar mapping, which is the injection channel Ni in this example. The header that preempted the lower priority message can then propagate as shown in Figure 3(d). The preempted header must once again go through the routing logic and obtain an outgoing virtual channel at the preemption point. In this example, the message is preempted at the first VC along its path ($1o$), although that need not be the case. A control unit that handles control flit arrivals is required for each VC. A discussion of the precedence conflicts and how they are handled by PPCS-RT is presented in [2].

3 Comparison of WR and PPCS-RT

3.1 Simulation Environment

The simulator is based on the CSIM [11] process oriented language. A flit-buffered network connected in a k-ary n-cube/mesh topology is simulated. The network configuration used for these experiments is an 8x8 mesh with 4 injection and consumption channels. This is the all-port model, where a node can inject/consume a message on all its ports simultaneously. For comparing the two models, we considered the dimension order routing algorithm [3]. We also considered the effect of adaptive routing with WR, with a minimal fully adaptive routing algorithm that requires two routing classes ($r = 2$) [4], each routing class consisting of v lanes. A header can be routed on any VC along any dimension in the first routing class, or along the highest dimension in the second routing class, called the dimension order routing class. Hence, rv VCs/link are required, where $r = 1$ for dimension order routing, and $r = 2$ for fully adaptive routing. Two approaches for implementing the fully adaptive algorithm with v lanes were considered. In the first (A1), if a header cannot obtain an available VC, the header waits for a VC from the dimension order class to become available, and does not contend for VCs in the first routing class. In the second (A2), if a header cannot obtain an available VC, it contends in round robin fashion until a free VC becomes available in either routing class. Approach A2 requires a centralized round robin arbitration scheme.

The timing parameters for the simulated PPCS-RT and WR models are as follows: Both models assume a physical link bandwidth (B) of 10 MB/s, *i.e.*, it takes $0.4\mu s$ to transfer a data flit across a link where a flit is assumed to be 4 bytes. The header flit in WR takes $1.2\mu s$ to propagate one hop. This includes the routing decision and the flit transfer time. The header flit in PPCS-RT takes $1.6\mu s$ to propagate one hop (t_r), due to the added complexity of the PPCS-RT router. The success flit has a cycle time per hop of $0.4\mu s$. We assume that the preempt, and free flits propagate without delay. This is a reasonable assumption since the overhead of preemption is very much smaller than the PE and DD phases, due to the fact that both the preempt and free flits are non-blocking.

The reason for this assumption, is that we used events to simulate preempt and free conditions thereby simplifying the PPCS-RT programming model. The ideal PPCS-RT latency C_i is given by:

$$C_i = h_i t_r + h_i/B + (h_i + s_i - 1)/B \quad (1)$$

where s_i is the size of message M_i, h_i is the number of hops traversed by M_i, B is the bandwidth of the link, and t_r is the header propagation time per hop. The first term in Eq. 1 is the time for the header to reach the destination, the second term is the time for the success flit to return, and the last term is the time taken by the DD phase.

A message generation program generates messages that are periodic with random source destination pairs. The message size is randomly chosen to be one of four message sizes (16, 128, 1024, 4096 flits). Depending on the message size, the period is randomly chosen from 16 values within a range of $2 - 100$ ms so that larger messages are restricted to larger periods. For example, a message of size 16 flits has a period in the range of $2 - 10$ ms while a message of size 4096 flits has a period in the range $20 - 100$ ms. This is done so as to represent real-time traffic more realistically, where smaller messages (often critical messages) occur more frequently than larger messages. Furthermore, we do not want very large messages to occur with a high frequency thereby saturating the network very quickly. The deadline of the message is chosen to be less than the period. Messages are generated until a predefined average link utilization (LU) is reached ranging from $0.1 - 0.5$. The link utilization at a link with k messages, is defined as the fraction of the link bandwidth utilized by the messages using the link, *i.e.*, $\sum_{i=1}^{k} C_i/p_i$, where C_i is given by Equation 1. Due to the periodic nature of messages, at a link utilization of 0.5, the number of messages existing in the network is very large (over 15,000) resulting in long simulation times. The simulator injects each generated message in the message set into the network initially at time 0. Thereafter, new instances are injected at their respective periods. Each message instance is a process which creates a header process. The header process in the case of WR spawns a data process at each node which is responsible for transferring the flits. In the case of PPCS-RT, the header process simulates the PE phase and backtracking if preempted.

3.2 Results

In Figure 4(a) we compare the performance in terms of Overall Miss Ratio ($OMR\%$) defined as the ratio of missed messages to total number of messages delivered, using PPCS-RT and WR with $v = 1$ and $v = 2$ under dimension order routing. In general, PPCS-RT significantly outperforms WR for the same v, with the difference in performance being more significant at increased link utilizations. The performance difference between PPCS-RT and WR is greatest for $v = 1$. PPCS-RT($v = 1$) outperforms WR($v = 2$) at high link utilization in terms of $OMR\%$ (Figure 4(a)). Hence, at increased link utilization, a large number of messages suffer from priority inversion in WR resulting in more deadline misses than PPCS-RT even though WR uses twice the number of lanes used by PPCS-RT.

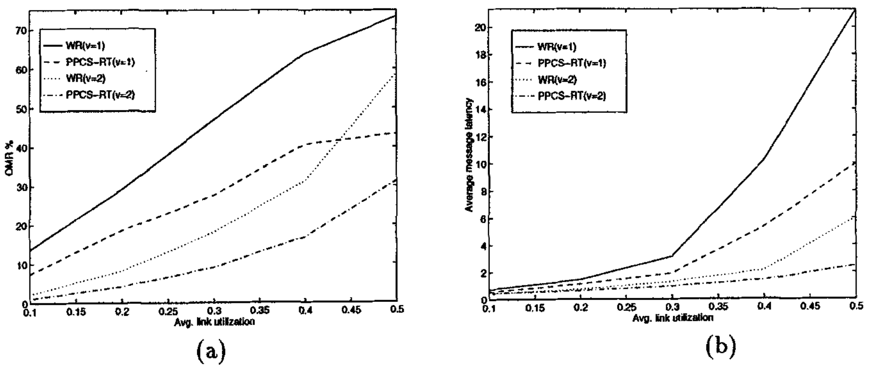
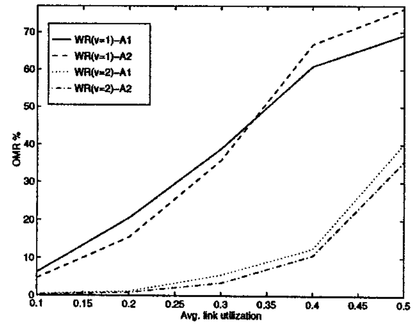


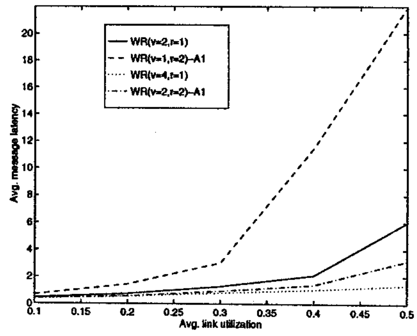
Fig. 4. WR versus PPCS-RT with dimension order routing(a) $OMR\%$ versus average link utilization (b) Average message latency versus average link utilization

Figure 4(b) shows the average latency of a message using WR and PPCS-RT for $v = 1$, and $v = 2$. PPCS-RT results in lower average message latency compared to WR for the same v despite the overhead of the PE phase. By resolving VC contention according to global priority rather than local priority, PPCS-RT achieves a lower average blocking time, and hence a lower average latency compared to WR. Although, PPCS-RT($v = 1$) has lower $OMR\%$ compared to WR($v = 2$) at increased link utilization, the mean latency of messages using PPCS-RT($v = 1$) is higher than that of WR($v = 2$) messages, as seen in Figure 4 (b). While mean latency is a good indicator of real-time performance with WR, this is not the case with PPCS-RT. With WR, most messages that miss their deadlines, do so by a small margin, while with PPCS-RT, the low priority messages miss by a large margin, contributing to the increased average message latency.

Next, we evaluated the effectiveness of adaptive routing with WR using a variable number of lanes v. Figure 5(a) shows the $OMR\%$ with the two implementations of fully adaptive minimal routing. A2 results in lower $OMR\%$ than A1 at low link utilizations with $v = 1$ and, for all link utilizations with $v = 2$. At high link utilizations and with only one lane ($v = 1$), the centralized round robin arbitration scheme performs worse than strict priority-based waiting for a channel in the dimension order routing class. This is due to the fact that with a single lane a large number of messages are blocked, and the centralized arbitration scheme becomes a bottleneck. A1 is also easier to implement in hardware. Comparing Figures 4(a) and 5(a), fully adaptive routing using A1 results in lowering the $OMR\%$ as compared to dimension order routing, for the same number of lanes v. However, the fully adaptive scheme requires twice the number of VCs compared to dimension order routing for the same v. When $v = 1$, adaptive routing alone performs marginally better ($<$ 10% difference in $OMR\%$) than dimension order routing. This is not the case when $v = 2$, where the difference in the $OMR\%$ is over 25%. In meshes, fully adaptive routing alone with no parallel lanes is not very effective since a large number of messages have only a single shortest routing path. Fully adaptive routing with parallel lanes effectively distributes the traffic resulting in good real-time performance. Figure 5(b) compares the average latency of wormhole routed messages using dimension order routing and adaptive routing (using A1) that require the same number of total VCs per link. As also seen from the $OMR\%$, the average latency of a WR message using adaptive routing is lower than a message using dimension order routing for the same v. However if the total number of VCs/link is fixed, then parallel lanes are more effective compared to adaptive routing. For example, WR($v = 2, r = 1$) performs better than WR($v = 1, r = 2$), where both require 2 VCs/link. Note that with priority-based link bandwidth arbitration, the physical link bandwidth is not continuously time multiplexed as with demand multiplexing. Hence the overhead of multiplexing is small as compared to demand multiplexing. We conclude that parallel lanes are more effective for priority-based traffic and require less hardware compared to adaptive routing.



(a)



(b)

Fig. 5. Comparison of WR with minimal fully adaptive routing using A1 and A2(a) $OMR\%$ versus average link utilization for two implementations (b) Average latency versus average link utilization for adaptive and dimension order routing using the same number of VCs

Finally, PPCS-RT is much better suited than WR for real-time traffic when only a few VCs per link exist, which is the case in most systems today. For example, PPCS-RT($v = 2$) employing dimension order routing has a lower $OMR\%$ than WR($v = 2$) using fully adaptive minimal routing at a link utilization of 0.5. Hence, we infer that complex adaptive routing schemes are not really required if a well suited priority-based flow control scheme exists for real-time messages under heavy traffic conditions. Our simulation experiments did not account for the overhead of preemption. This overhead, affects the preempting message delivery time by a small amount. Although the number of misses is expected to increase marginally with this overhead, the general trend that low priority messages will miss with large margins, the enforcement of priority order, and the lower $OMR\%$ of PPCS-RT compared to WR for small v, are expected to remain.

4 Conclusions

In this paper we compared the performance of PPCS-RT, a new flow control model that we proposed for priority-driven traffic, and WR, in terms of overall miss ratio. We also examined the effect of VCs for routing adaptivity and parallel lanes using WR. Parallel lanes provide better performance compared to adaptive routing for the same number of VCs/link. An important problem that we are investigating is a comparison of PPCS-RT and WR taking into account the increased latency due to additional flow control complexity, and routing complexity, and the transmission times of preempt and free flits.

References

1. Li, J.-P., Mutka, M.W.: Real-time virtual channel flow control. J. Par. Dist. Comp. **32**(1996) 49–65
2. Balakrishnan, S., Özgüner, F.: A priority-based flow control mechanism to support real-traffic in pipelined direct networks. To appear Proc. Int. Conf. Par. Proc. (1996)
3. Ni, L.M., McKinley, P.K.: A survey of wormhole routing techniques in direct networks. IEEE Computer (1993) 62–76
4. Duato, J.: A new theory of deadlock-free adaptive routing in wormhole networks. IEEE Trans. Par. Dist. Sys. **4(12)**(1993) 1320–1331
5. Dally, W.J.: Virtual-channel flow control. IEEE Trans. Par. Dist. Sys. **3(2)**(1992) 194–205
6. Leung, J.Y.-T., Whitehead, J.: On the complexity of fixed-priority scheduling of periodic real-time tasks. Performance Evaluation **2(4)**(1982) 237–250
7. Gaughan, P.T., Yalamanchili, S.: A family of fault-tolerant routing protocols for direct multiprocessor networks. IEEE Trans. Par. Dist. Sys. **6(5)**(1995) 482–497
8. Dolter, J. *et. al.*: SPIDER: Flexible and efficient communication support for point-to-point distributed systems. Proc. Int. Conf. Dist. Comp. Sys.(1995) 574–580
9. Shukla, S.B., Agrawal, D.P.: Scheduling pipelined communication in distributed memory multiprocessors for real-time applications. Proc. Int. Symp. Comp. Arch. (1991) 222–231
10. Balakrishnan, S., Özgüner, F.: Providing message delivery guarantees in pipelined flit-buffered multiprocessor networks. To appear Proc. IEEE Real-time Technology and Applications Symp. (1996)
11. Schwetman, H.: CSIM: A C-based process-oriented simulation language. Proc.Winter Simulation Conf. (1986) 387–396