A Study of Preemptable vs. Nonpreemptable Token Reservation Access Protocols

W. Timothy Strayer

Computer Science Report No. TR-90-32
December 5, 1990

A Study of Preemptable vs. Non-Preemptable Token Reservation Access Protocols

W. Timothy Strayer
Department of Computer Science
University of Virginia
wts4x@virginia.edu

ABSTRACT

LAN protocols based on token reservation, like those found in the IEEE 802.5 Token Ring and the SAE High Speed Ring Bus LAN specifications, provide service to messages in *nearly* priority order. Token Reservation fails when higher priority messages enter the system too late to be included in the priority bidding scheme or after a lower priority message successfully captures the token. Although allowing preemption in the media access protocol could reduce or eliminate these failures, no preemptable protocols exist among the extant LAN protocols. We seek to determine if the performance difference between a preemptable and a non-preemptable protocol is noticeable or significant; since no preemptable protocols exist there is no basis for comparison. Here we present an investigation of the effectiveness of token reservation with respect to providing service in priority order. We describe a preemptable token reservation protocol, and implement it, along with the non-preemptable token reservation protocol, in a simulator. We compare them using conventional metrics (such as throughput and delay), as well as introducing a new metric, the *priority inversion ratio*, to quantify the effectiveness of the protocols for providing priority ordered service. We show that preemption in a token reservation protocol can provide better service to high priority messages.

1. Introduction

Each extant media access protocol is designed to provide some form of discrimination between messages, whether it is decidedly nothing (as in Ethernet [IEEE85a]), or class-based (as in Token Bus [IEEE85b] and FDDI [ANSI86]), or priority-based (as in Token Ring [IEEE85c] and SAE High Speed Ring Bus [SAE87]). Users provide some ranking information and the media access protocol acts upon it. With priority-based protocols, the goal is to provide the best service (i.e., lowest latency) to the highest priority messages without regard to fairness to the lower priority messages. Current priority-based protocols cannot meet this goal completely because they may not always be serving the highest priority messages. Once the protocol has decided which message is the highest priority, it services that message; yet a higher priority message may have arrived during the decision process or during the service of the message, and thus would have to wait. To ensure that waiting time for the highest priority messages is minimized, the protocol must allow preemption.

Yet no extant protocol provides preemption. The intuitive justification is that since message size is bounded, and thus each station with a highest priority message will be allowed access to the medium in a bounded amount of time, then preemption is not necessary. The question is really whether or not the priority-based schemes in extant protocols are close enough approximations to the purely preemptable scheme.

Is the performance difference between a preemptable and a non-preemptable protocol noticeable? Is it significant? In this paper we describe a preemptable token reservation protocol, and

implement it within a simulator to compare it with the non-preemptable version. We also simulate a protocol where the access method is trivial: the global queue of messages is treated as though it were a single queue rather than distributed about a network. This is the optimal priority-based protocol, presented as a benchmark for comparison.

To facilitate the evaluation of the token reservation priority scheme, we introduce a new metric, the *priority inversion ratio*. It provides a measure of the effectiveness of a priority scheme. Thus we examine the question of whether or not an extant priority-based scheme, such as token reservation, is an adequate approximation to a priority preemptable scheme by attacking it from two angles: how it compares to a preemptable protocol, and how effective it is at providing priority ordered service.

2. Media Access Protocols

The media access protocol provides the method by which the messages at the various stations contend for use of the medium. It defines the arbitration policy and dictates the mechanisms used to enforce this policy. Below we briefly describe the token reservation method of media access control. We also describe a preemptable version of this protocol, and for comparison, we include the trivial protocol for a centralized queue.

2.1. Token Reservation

A token reservation protocol, like that found in both the IEEE 802.5 and the SAE High Speed Ring Bus (HSRB), attempts to determine which message among those messages pending service should be the next to receive the use of the medium. It uses a special frame, the token, in a threefold capacity. First, the token represents the single authorization to use the medium, preventing chaos during contention for use of the medium. A *free* token is available for capture by a qualified station so it can transmit a message; since only one such free token exists at any moment, only one station can transmit at any time. Once the token is captured, it is termed a *busy* token. Second, a busy token is a message header. The capturing station appends its message to the busy token. The third use is as a method for the stations distributed about the network to share state. In particular, the token provides the mechanism for the stations about the ring to determine which among them has the next message to be serviced. The mechanism for state sharing is called the *reservation* field.

The reservation field takes message priority values as bids from the stations; the station with the highest bid qualifies to claim a free token and thus will eventually be able to append its highest priority message to the token. Stations continue to bid every time they see a token (busy or free) pass, until eventually they, too, are able to capture the token. A free token may be captured if its *priority* field value is equal to or less than the station's message's priority. The priority field of the token is set to the highest reservation bid when the token becomes free. Within one rotation of the ring the token will be captured by the station which won the bid, and the other stations must again express their bids for the next time the token is free.

The token reservation protocol we implemented within the simulator is similar to the SAE HSRB in that it does not require *stacking* like IEEE 802.5 does. Also like the HSRB, it allows only a single message to be serviced per token visit. Unlike the HSRB, however, it does not allow the Short Message Protocol optimization.

2.2. Preemptable Token Reservation

As a general scheduling policy, priority preemptive service requires knowledge of the highest priority task in the system at every point in time. If a task enters the system with a higher priority than the task being serviced, then it may preempt that lower priority task and begin accepting service. Unfortunately, in a distributed queueing system, knowing when a higher priority task should preempt requires some sharing of information between the distributed queues. Even then, the wrong choice may be made if the sharing of the information must only happen at discrete times. Such is the case

with the token reservation protocol.

Maintaining the threefold capacity of the token, we designed a preemptable token reservation protocol which allows preemption only on the token visit. In this respect, a token still represents the authority to use the medium, it is still a message header when claimed, and is still used to reserve circulation of the token at a certain priority. Unlike the non-preemptable protocol, however, a busy token may be *confiscated* by a station with a higher priority message, and the lower priority message will be preempted. This preempted message is simply discarded, so work is lost; however, a higher priority message does not have to wait until the token becomes free. Thus the highest priority messages never have to wait on the service of a lower priority message; they only wait on the token to visit. Clearly, the preemptable reservation protocol trades away throughput (via lost work) in favor of reducing the latency of the highest priority messages.

2.3. Centralized Queue

For comparison purposes we also simulated a centralized queue, both with and without preemption. By making the queue of messages centralized rather than distributed, the access method is trivial: the server simply takes the highest priority message to service next. This allows us to observe the effect of having to spend bandwidth in order to determine the highest priority message in the distributed queue, as well as giving us the optimal values for the performance metrics.

3. Priority Inversion Metric

In the general task/server model, priority based service has a single goal: to offer service to the highest priority task. As a consequence, lower priority tasks receive diminished service. If the priority scheme is effective, then the highest priority tasks will never be impeded by a lower priority task. Preemption is independent of the prioritization, though clearly preemptability of tasks gives higher priority tasks better service.

Priority inversion ([GOOD88]) occurs when a higher priority task must wait on a lower priority task. This phenomenon is especially troublesome in multitasking operating systems where tasks are not independent of other tasks, and must be synchronized. This can lead to a high priority task being locked out of a critical region, and hence prevented from proceeding, while a lower priority task holds the lock and thus proceeds. In communications, the messages are always independent, but one message may gain access to the communications processor forcing all others to wait. Priority inversion in a communications environment occurs when a lower priority message is processed while a higher priority message is delayed.

The *priority inversion ratio* (PIR) is a metric designed to measure the extent to which a priority scheme can meet its goal of servicing tasks in priority order. This ratio is the number of tasks enqueued which are at a higher priority than the task currently being serviced, divided by the number of tasks enqueued when the task is accepted for service. A measure of 0 implies that at every decision point the highest priority task is always serviced. A measure of 1 implies that every enqueued task is higher priority than the one being serviced (complete inversion of the priority scheme).

Mathematically, the priority inversion ratio for priority i is:

$$PIR_i = \frac{\sum_t Q_{ti}}{\sum_t Q_t}$$

Where Q_t is the number of tasks queued at time t and Q_{ti} is the number of tasks of priority greater than i queued at time t.

In LANs, the priority scheme is implemented by the media access protocol, and the tasks are messages. The media access protocol must determine which station has the highest priority message and then grant access to that message in such a manner that it is clear when a message may claim access and when it may not. As described above, token passing is one such method for arbitrating the use of the medium, as well as allowing stations to vie for the use of the medium through the priority reservation scheme.

Measuring the effectiveness of a priority scheme within a distributed queueing system such as a LAN is interesting because the highest priority message cannot be known without first spending some time to determine it. Even when the highest priority message is determined, it is not guaranteed that a higher priority message will not arrive subsequently. Such effectiveness measurements are most often shown by graphs of the delays of messages at each individual priority level as a function of offered load. The priority scheme is deemed effective if messages of higher priority are delayed less than messages of lower priority. However, measuring delay alone cannot adequately reveal the effectiveness of the priority scheme since it cannot show the percentage of messages impeding the service of the higher priority messages, or at what offered load this impedance is greatest. The priority inversion ratio augments the delay measurements by providing a measure of the failure of the priority scheme to choose the highest priority message in the system. Further, it quantifies this failure by reporting the portion of messages which are being prevented service by the service of a lower priority message.

4. Simulation Parameters

Simulators implementing the preemptable and non-preemptable token reservation protocols, and both versions of the centralized queue protocol, were subjected to various experimental situations. The parameters to the simulation were thought to be reasonable for a large distributed system. The medium was assumed to be a ring with a signaling speed of 100 megabits/second, a reasonable speed and technology which are soon to be commonplace. There were 100 stations uniformly distributed about the ring, 10 meters apart. Offered load varied from 10% to 110%. When multiple priorities were used, the total offered load was divided uniformly among each priority level. There were 8 priority levels, where priority level 8 was the highest. Events were generated with a Poisson distribution for each priority level. Message lengths were exponentially distributed about a mean of 1000 bytes. The maximum packet size on the ring was also 1000 bytes, so when a message exceeded this packet size, multiple events with the same arrival time were generated as though the group of packets were the result of segmentation by a higher layer protocol (such as a transport protocol).

5. Performance Measurements

The experiments we report here were not designed to exhaustively evaluate any of the protocols implemented, but rather to provide results for comparative evaluation of the effectiveness of token reservation for providing priority ordered service. The conclusions are conditioned on the reasonable choice of parameters; we do not attempt to assess the parametric sensitivity of the metrics. In this section we offer measurements of overall performance to show the cost of using the access protocol as well as the cost of preemption within the access protocol. We then examine the benefits and cost of preemption specifically, noting the performance of individual priority levels of service. Finally, we show the extent to which the non-preemptable version of token reservation allows priority inversion, noting that there would be no priority inversion for the preemptable token reservation protocol.

Figure 1 shows the total throughput and delay for the system as a function of offered load. The solid and dotted curves in the top graph show the results of experiments where all messages in the system have the same priority, effectively nullifying the priority scheme. These two curves exhibit the highest throughput since minimal bandwidth was spent trying to determine the next highest

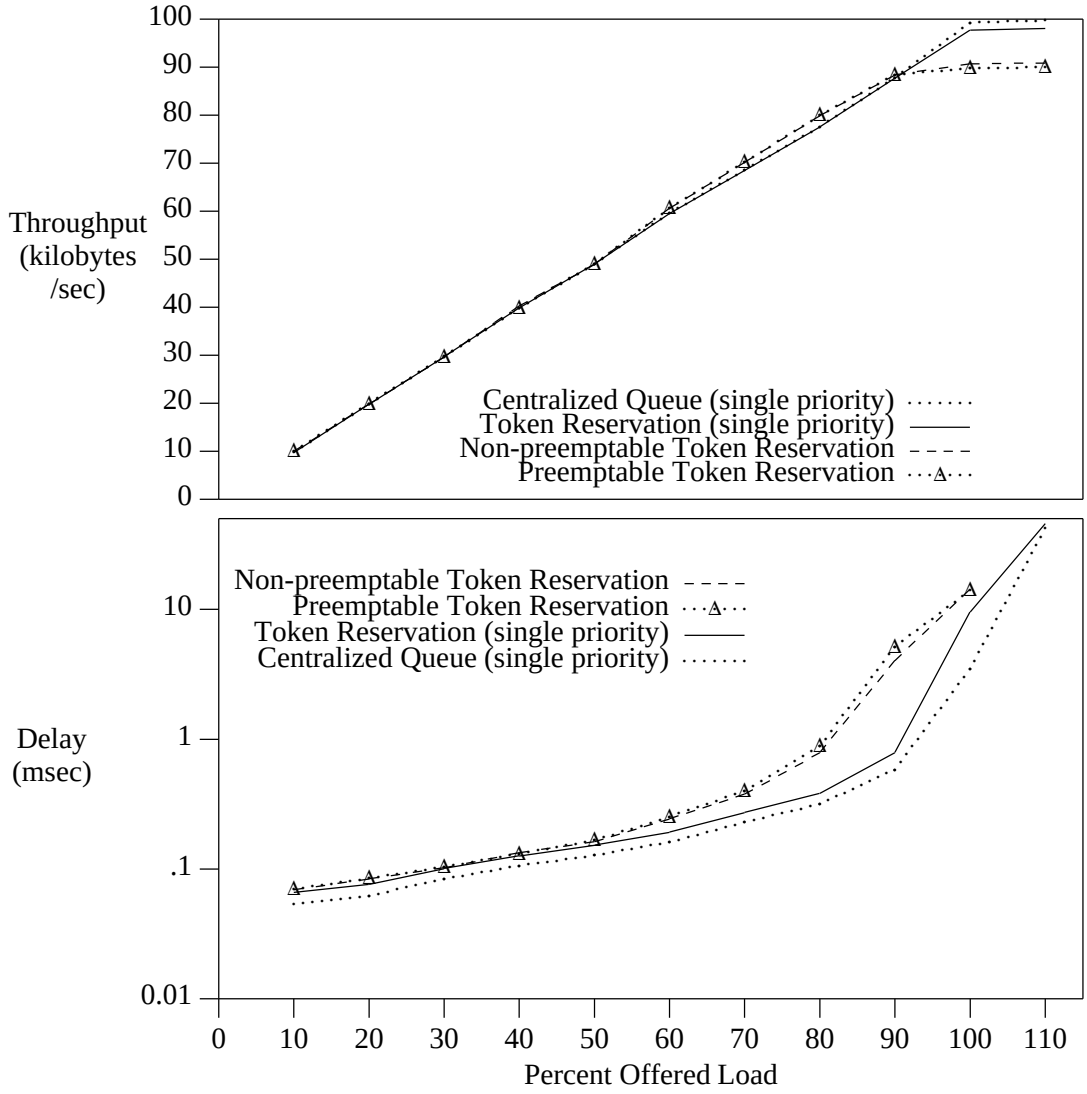

Figure 1 — Average Throughput (above) and Average Delay (below) for Centralized Queue and Token Reservation with and without Preemption

priority: the next highest is simply the next message in the next downstream station. The dotted curve shows the performance of the centralized queue, where no bandwidth is required to find the next message. Therefore, this curve shows the optimal throughput under the given conditions.

When priority is used to distinguish messages, some bandwidth must be used to determine which station has the message with the highest priority. Thus, for both the dashed and the dotted-delta curves, there is loss of throughput at the higher loads. We also see that there is a slight cost for preemption in terms of total throughput since some messages are being partially serviced and then

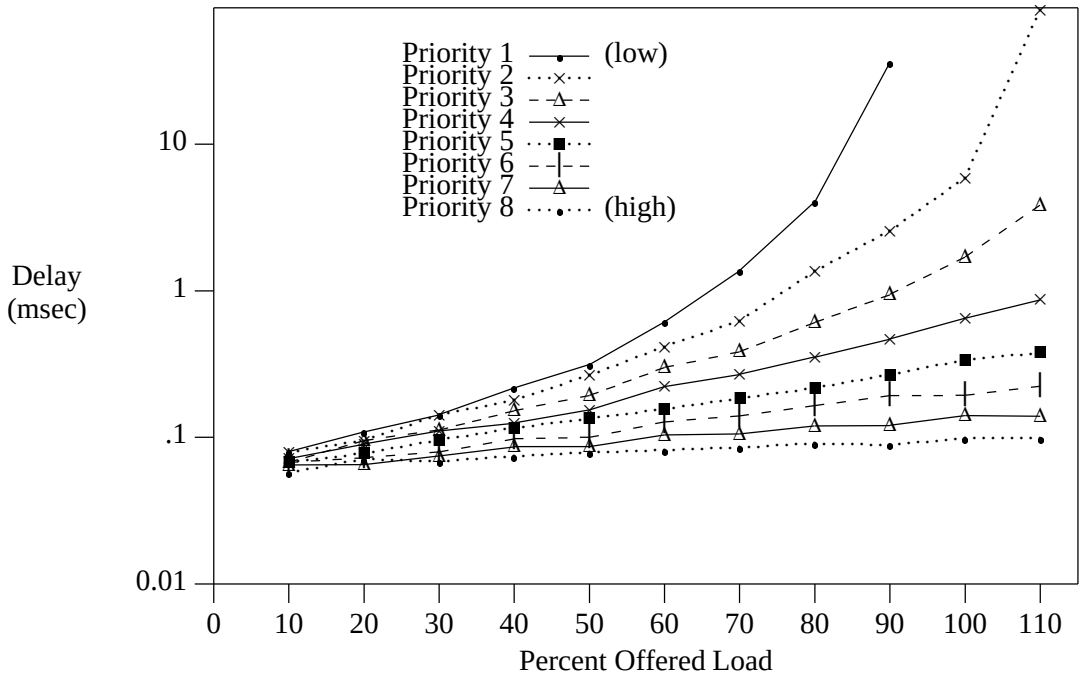

Figure 2 — Delay for Individual Priorities, Preemptable Token Reservation

dropped for higher priority messages. This work is lost, and so is the portion of throughput related to the dropped message.

Note that there is little or no difference in this average throughput for each of the configurations until the offered load exceeds 90%. Since both using a priority scheme and preempting within that scheme are designed to trade some total system performance for better performance of the highest priorities, it is interesting to see that this tradeoff does not become effective until the offered load is very high.

The bottom graph shows these same configurations with respect to delay. Notice that the difference between using and not using a priority scheme is more dramatic since the lower priority messages are being actively delayed to benefit the higher priority messages. Also notice that allowing preemption increases the average delay.

Figure 2 shows the delays for each individual priority when preemption is allowed. As we would hope, there is little difference in the delay of the highest priority messages from low loads to high. Even when the system is overloaded, these messages are provided very good service. Notice how the service to the lowest priority messages is effectively shut off at 90% offered load.

Figure 3 shows the difference in delay between highest and lowest priority messages for both non-preemptable and preemptable protocols. These two graphs demonstrate that preemption does provide the highest priority messages with better service than non-preemptable can, and that the decreased delay is at the expense of the lower priority messages. Having a non-trivial access method,

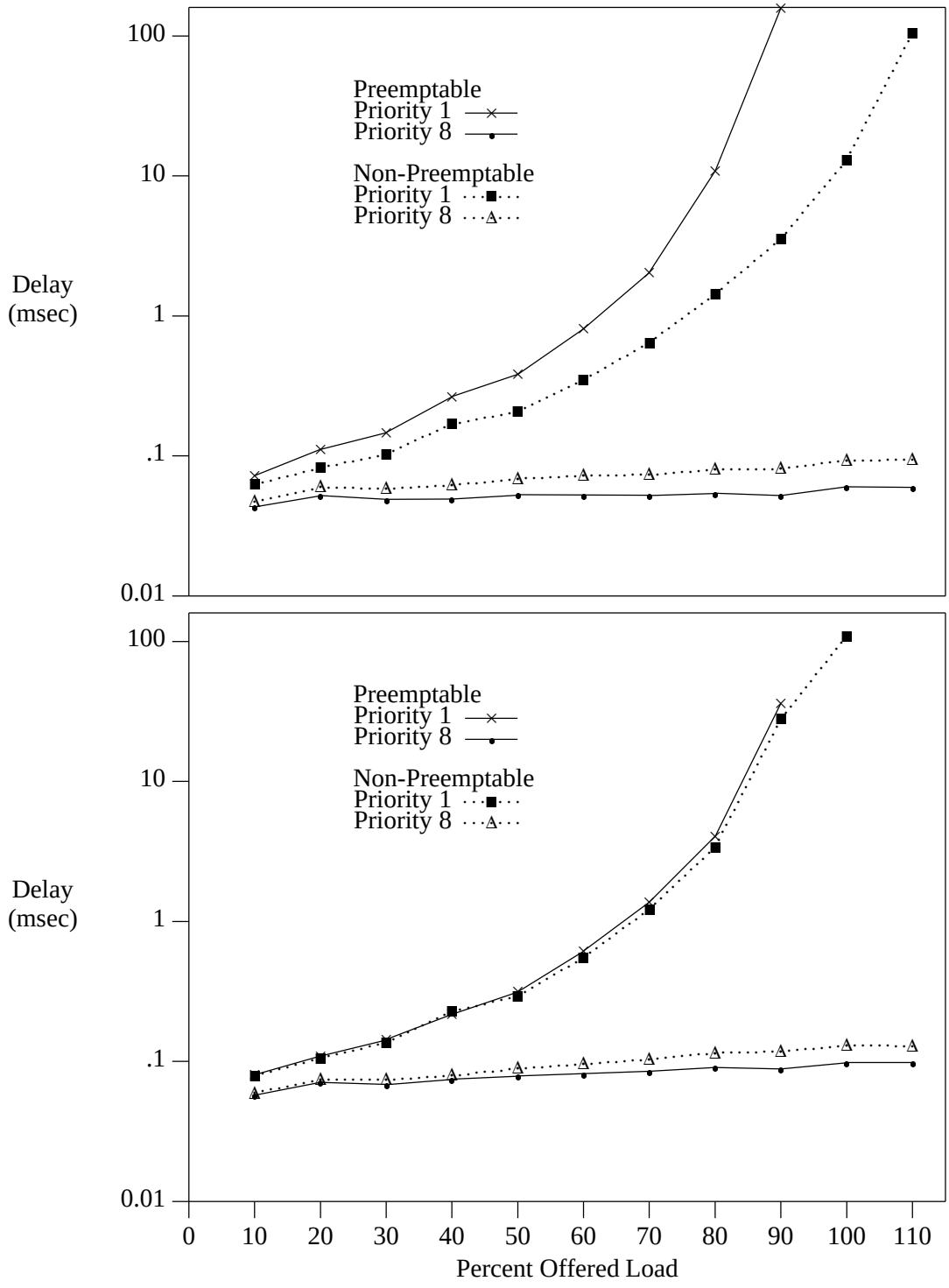

Figure 3 — Margin Between Delay for Highest and Lowest Priority Service
Centralized Queue (top) and Token Reservation (bottom)

as in the bottom graph, damps this effect somewhat from that shown in the top graph.

Figure 4 isolates the highest priority delays for both preemptable and non-preemptable token reservation protocols, and shows them in linear rather than logarithmic scale. We see that the preemptable service offers virtually constant delay for all offered loads, whereas for the non-preemptable service, delay of the highest priority messages at 100% offered load is twice that for 10% offered load. Thus, delay for the highest priority messages receiving preemptable service is load-independent.

Figure 5 shows the priority inversion ratios for each individual priority. At low offered loads the interarrival time between messages is large, so low priority messages can be serviced without interfering with the service of the higher priority messages. At 10% offered load, fewer than 1% of the messages queued in the system when a lowest priority message was serviced were messages of higher priority. As offered load was increased, that percentage also increased: at 60% offered load service of a lowest priority message occurred while about 7.7% of the messages queued where actually higher priority. In fact, the priority scheme was least effective at 60% offered load for all of the priority levels. This is due to the fact that the interarrival time between messages was large enough to allow each priority level to be serviced, but small enough that when a level was serviced some possibility existed that there were higher priority messages enqueued. At higher offered loads the priority scheme again favored the higher priority messages since the token remained reserved at these high priority levels longer. Also, as service was shut off to the lower priorities, the queue for service became longer causing the ratio to have a very large denominator.

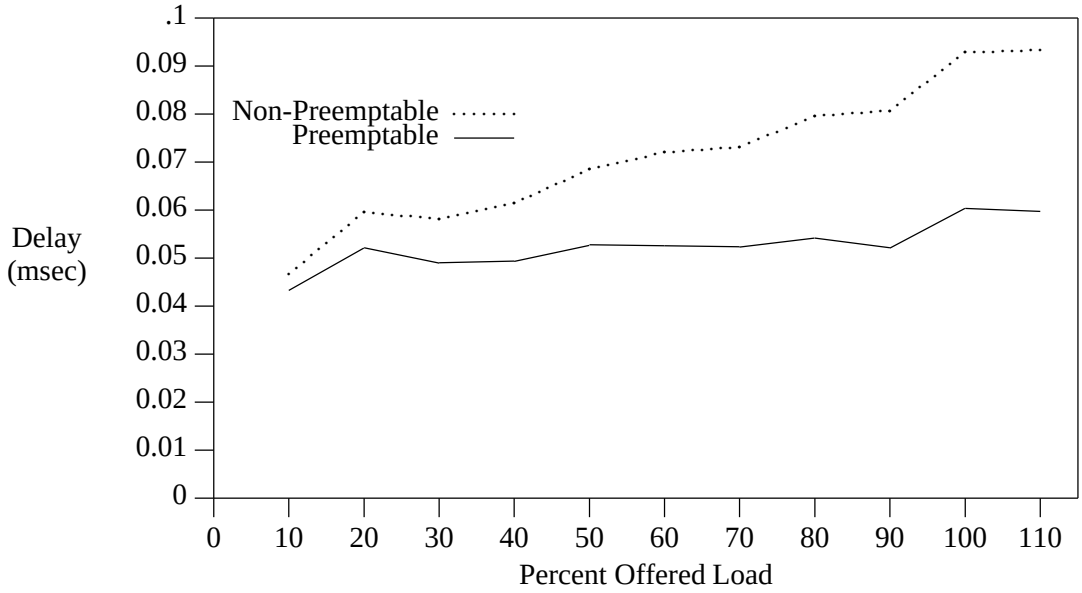

Figure 4 — Delay of Highest Priority Service,
Preemptable vs. Non-Preemptable Token Reservation

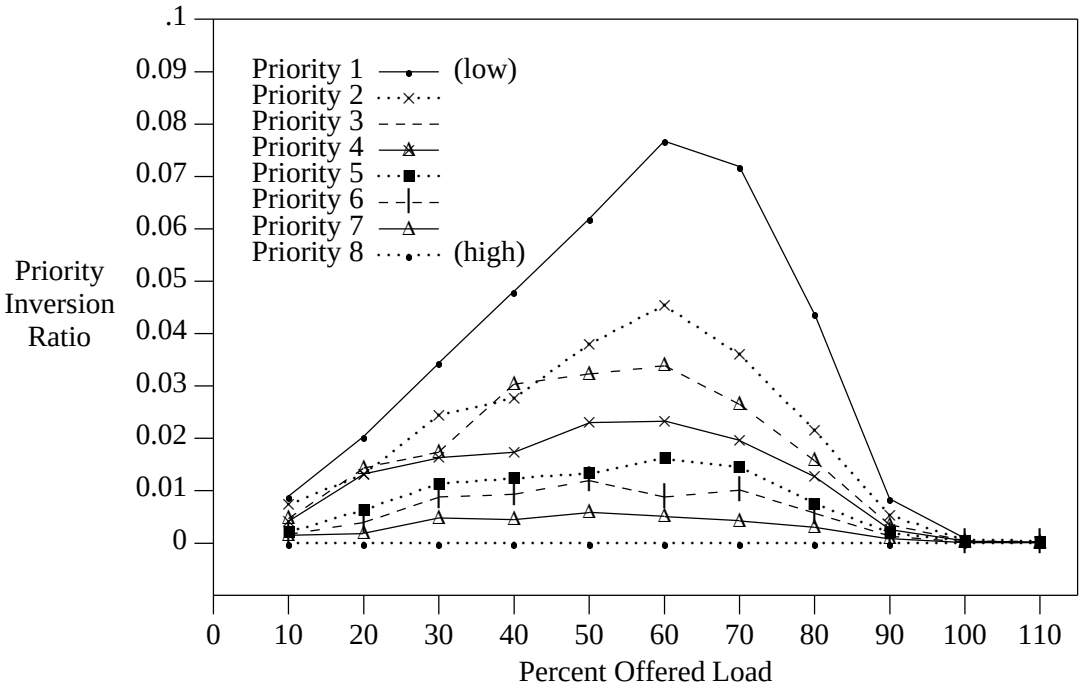

Figure 5 — Priority Inversion Ratio for Individual Priorities, Non-Preemptable Token Reservation

Note that the highest priority level always has a priority inversion ratio of 0. This is clearly true since there can never be a higher priority message enqueued at the time of service of the highest priority messages. Also note that the priority inversion ratio is measured at the point of service for a message, and thus does not measure the effect of messages arriving while the message is being serviced. This, too, is priority inversion, although the ratio does not reflect it. Consequently, if the priority scheme always chose the highest priority message next, with or without preemption, the priority inversion ratio would be 0, whereas the delay metric would reflect the difference between preemption and non-preemption. Our preemptable token reservation protocol presented an inversion-free service since a message could not survive until delivery unless it remained the highest priority message in the system.

6. Conclusions

We set out to determine if preemption in a media access protocol provides better performance for highest priority messages, and if so, how significant is this increase. Since no extant media access protocols allow preemption, we sought the answers through simulation and comparison of various metrics, including the newly introduced *priority inversion metric*. Our conclusions follow.

As shown by the priority inversion ratio, non-preemptable token reservation provides more nearly pure priority ordered service at the low and high extremes of offered load, yet for intermediate offered loads, service was allowed to low priority messages as higher priority messages were made to

wait. In our experimental configuration, at 60% offered load, about 7.7% of the messages enqueued at the time of service to a lowest priority message were messages of higher priority. These numbers are certainly functions of the configuration and the respective loads contributed by each of the priority levels. However, the fact that the token reservation protocol allowed priority inversion is not surprising. It is interesting that the protocol fails to provide the priority order service for the range of loads which would be most common for a LAN.

Introducing preemption at token visit time eliminated the delay incurred by priority inversion. Since no message is delivered without having had the highest priority in the system at that time, there is *no* priority inversion. The cost of using preemption to benefit the highest priority messages is increased delay to the lower priority messages—first they must wait to capture the token, and even then they may be preempted. When work is lost through preemption, the total system's throughput is reduced. Yet, allowing preemption did decrease the delay of highest priority messages and virtually eliminated the influence of offered load on that delay. So to answer our first question, there is a performance benefit gained by the highest priority messages when preemption is allowed.

The second question, is this benefit significant, is more qualified. If the system's goal is to provide the lowest possible latency to the highest priority messages, then the increased performance gained by preemption in the protocol may be necessary. However, from a pragmatic point of view, gaining a few microseconds for the highest priority messages in a fully loaded system may not be where effort is best placed. This difference may be quickly made negligible by delays in other aspects of the communication system, like data movement between buffers, backplane transfers, interrupt response times, and protocol processor implementation. Furthermore, to be truly effective, the policy of providing the best service to the highest priority messages with preemption of lower priority messages must be maintained at all message processing levels, not just at the media access layer. Yet, considering only the media access protocol, preemption provides a measurable improvement toward providing the best service to the highest priority messages at every point in time.

7. References

[ANSI86] American National Standards Institute, "FDDI Token Ring Media Access Control Standard", *Draft proposed Standard X3T9.5/83-16, Rev. 10*, February 1986.

[GOOD88] Goodenough, J. B. and Sha, L., "The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High-Priority Ada Tasks", Technical Report Carnegie-Mellon University, Carnegie-Mellon University Software Engineering Institute, Pittsburgh, PA/SEI-88-SR-4, March 1988.

[IEEE85b] Institute of Electrical and Electronics Engineers, "IEEE Standard 802.4 Token-Passing Bus Access Method and Physical Layer Specifications", 1985.

[IEEE85c] Institute of Electrical and Electronics Engineers, "IEEE Standard 802.5 Token Ring Access Method and Physical Layer Specifications", 1985.

[IEEE85a] Institute of Electrical and Electronics Engineers, "IEEE Standard 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", 1985.

[SAE87] Society of Automotive Engineers, "SAE AS4074.2 High Speed Ring Bus, Final draft Standard", June 1987.