Real-Times Systems, 7, 5–32 (1994)

Conventional and Early Token Release Scheduling Models for the IEEE 802.5 Token Ring

SHIRISH S. SATHAYE SATHAYE@NACTO.LKG.DEC.COM
Network Architecture & Performance Group, Digital Equipment Corporation, Littleton, MA 01460

JAY K. STROSNIDER
Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh PA 15213

Abstract. We develop analytical *scheduling models* for both the original IEEE 802.5 token ring protocol and a recent extension to the original protocol that allows early token release (ETR). A scheduling model is an abstraction that supports reasoning about the timing correctness of a given set of real-time messages scheduled on the network. Scheduling analysis of the original IEEE 802.5 token ring protocol has previously been discussed in Strosnider and Marchok (1989) and Pleinevaux (1992) in the context of improving responsiveness of soft deadline aperiodic messages. In contrast, this paper develops schedulability conditions for arbitrary periodic message sets. The main contributions of this work are: Scheduling models for both the original protocol and ETR protocol; comparison for maximum achievable utilizations for the two protocols; comparison between the original protocol and ETR from a schedulability viewpoint. We also demonstrate the utility of our scheduling models to select network operating parameters such as maximum packet size, and to quantify effects of parameters such as the number of stations, and network size on schedulability.

1. Introduction

In this paper we develop *scheduling models* for two versions of the IEEE 802.5 token ring. In the original protocol a station must wait for its packet to traverse the ring before releasing a new token. A recent protocol extension termed *Early Token Release* (ETR) (IEEE 1992) forces token release at the end of transmission. The scheduling model facilitates reasoning about the timing correctness of a set of messages on the network. It incorporates the effects of network components that potentially affect message timing behavior. The model is useful not only in determining the schedulability of a message set, but also in selecting network parameters such as maximum packet size. We use the models to compare the ETR protocol with the original protocol, designated *Conventional Token Release* (CTR) for easy reference. The ETR protocol increases throughput, particularly in larger networks by allowing multiple packets on the network. ETR reduces the time that stations hold the token, hence reducing packet transmission overhead. This has a favorable impact on schedulability. However, ETR may allow a large number of lower-priority message transmissions even when high-priority messages are waiting. This phenomenon, called *priority inversion*, has been shown to cause degraded schedulability (Sha, Rajkumar, and Lehoczky 1990). Therefore, depending on network parameters and traffic type, either protocol may exhibit better real-time scheduling characteristics.

Time-constrained communication in multi-access networks has been addressed by several researchers (Kurose, Schwartz, and Yemini 1984, Arvind, Ramamritham, and Stankovic 1991, Ramamritham 1987). Ring networks have been studied by (Lee and Shen 1990, Lim,

Yao, and Zhao 1991, Agrawal et al. 1992, Strosnider and Marchok 1989, Pleinevaux 1992). Lee and Shen (1990) present a dynamic control protocol for real-time communication in token ring networks. A window protocol for token ring networks which implements the earliest-deadline-first scheduling policy is described in Lim, Yao, and Zhao (1991). Agrawal *et al.* describe a bandwidth allocation technique for timed-token ring networks that guarantees deadlines below a certain utilization bound.

Scheduling the IEEE 802.5 Token Ring for hard real-time traffic has been discussed in Strosnider and Marchok (1989), Pleinevaux (1992), and Strosnider (1988). They considered the CTR token ring protocol and discussed techniques for improving soft real-time message responsiveness. The concept of a *schedulable unit* is introduced. The schedulable unit is the time necessary to transmit a packet, propagate it around the ring and have it arrive at a destination. The schedulable unit concept cannot be applied to the ETR protocol since the next transmission can begin prior to the current transmission's arrival at its destination. Further, the schedulable unit cannot be directly used to test an arbitrary periodic message set's schedulability. In this paper we overcome these shortcomings and develop scheduling models for both CTR and ETR using an identical theoretical framework, to allow easy comparison. In this paper we:

- Develop a generic scheduling framework for token ring networks.
- Develop a scheduling model for CTR based on the generic framework.
- Develop a scheduling model for ETR that is also based on the framework, and hence consistent with the CTR model.
- Compare the maximum network utilization achievable with CTR and ETR protocols.
- Demonstrate the impact of limited priority levels in token ring scheduling.
- Demonstrate the utility of scheduling models in making choices of network parameters such as maximum packet size and to quantify the impact of network size on the schedulability of a message set.
- Compare real-time scheduling characteristics of CTR and ETR for particular message sets. This approach can be used to select the protocol, given a particular configuration, and a set of periodic message streams or connections.

The remainder of the paper is organized as follows. In Section 2 we introduce our traffic model and define overhead, blocking and schedulability in a network context. Section 3 discusses token ring network operation and priority arbitration. In Section 2.1 we review real-time scheduling theory and develop a generic scheduling framework for token ring networks. Section 3.2 analyzes the CTR protocol and develops a scheduling model based in the generic framework. The ETR scheduling model is developed in Section 3.3. In Section 3.4 we develop expressions for maximum achievable utilization with each protocol. The CTR and ETR scheduling models are used in Section 3.5 to analyze the schedulability of an example message set. The protocols are compared from a schedulability viewpoint and operating regions where each is expected to perform better are determined. Concluding remarks are made in Section 3.6.

2. Background

This section introduces our traffic model and defines overhead, blocking, and schedulability, in the network context. We consider a token ring network with n stations $\{S_1, S_2, \ldots, S_n\}$. Each message stream (or connection) τ_i in this network is characterized by $\tau_i = (C_i, T_i, D_i)$, where C_i is the length of the message, T_i is its period in microseconds, and D_i is its deadline. We use the terms *connection* or *message stream* interchangeably. We assume that $D_i \leq T_i$. C_i is in units of transmission time expressed in microseconds. In a 16 Mbits/s token ring, the amount of time to transmit a byte (8 bit octet), is 0.5 μs. For example, message τ_i of length 256 bytes and period 4 ms will have $C_i = 0.5 * 256 = 128$ and $T_i = 4000$.

Let message streams $\{\tau_1, \tau_2, \ldots, \tau_n\}$ represent decreasing static priority order. Each message is assumed to be transmitted as one or more packets, depending on its length. Each packet is non-preemptable, however, messages which span multiple packets are preemptable at packet boundaries. Messages are independent and queued in priority order. Message transmission involves *overhead* and *blocking* which may be defined as follows:

- **Overhead:** Time spent by station on behalf of a message in addition to message transmission time, or time spent due to system issues independent of message transmission. Examples of overhead include time to transmit control bits in each packet, the time spent in acquiring and transmitting the token, the uncertainty due to unsynchronized clocks, etc.

- **Blocking:** Amount of time a high-priority message is delayed by lower-priority transmissions. Also known as the duration of priority inversion (Sha, Rajkumar, and Lehoczky 1990).

Tasks or messages on a single resource are said to be *schedulable* if each task/message completes before its deadline. This definition of schedulability must be extended in the network context. Strosnider (Strosnider and Marchok 1989) considered a message schedulable if it reaches its destination within its deadline. For networks such as IEEE 802.5 in ETR mode, FDDI, and dual link networks, where the next packet transmission can begin before a particular packet reaches its destination, it is more useful to consider the notion of *transmission schedulability* (Sha, Sathaye, and Strosnider 1992). A set of messages is said to be transmission schedulable (t-schedulable) if each message can be transmitted before its deadline. The end-to-end latency of the message is given bounded by the following expression.

$$\text{End-to-End Latency} \leq \text{Transmission Deadline} + PD_{se} \tag{1}$$

The end-to-end deadline of the message is satisfied if

$$\text{End-to-End Deadline} \geq \text{End-to-End Latency} \tag{2}$$

where PD_{se} is the propagation delay between the source and destination of the message. We use the traditional notion of schedulability defined in Strosnider and Marchok (1989) when discussing the CTR protocol, and use t-schedulability to discuss the ETR protocol.

To compare performance of the two models, we change the deadline of each message τ_i in the message set to $D_i = T_i - W_T$ when calculating t-schedulability in ETR mode. W_T is the time required to traverse the ring and as such represents the worst case propagation delay between any source and destination. The t-scheduling model checks whether each message can be transmitted by its deadline. Then since $T_i = D_i + W_T$, the message is guaranteed to reach its destination by the end of its period.

2.1. *Scheduling Framework*

The token-ring scheduling model we develop, is based on real-time theory for scheduling tasks on a single resource developed in Liu and Layland (1973) and Lehoczky, Sha, and Ding (1989). Consider tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$ arranged in decreasing priority order. Task τ_i is characterized by an execution time C_i, a period T_i, and a deadline D_i. This analysis considers only tasks with deadlines which are **not greater** than task-periods. We summarize useful results from Liu and Layland (1973) and Lehoczky, Sha, and Ding (1989).

- The longest response time for any task τ_i occurs at its *critical instant* which occurs when it is instantiated simultaneously with all higher priority tasks.
- All task deadlines will be met if the first request of each task meets its deadline.
- A task set is schedulable if the following equation holds.

$$\forall i, \quad 1 \leq i \leq n, \quad \min_{0<t\leq D_i} \sum_{j=1}^{i} \frac{C_j}{t} \left\lceil \frac{t}{T_j} \right\rceil \leq 1 \tag{3}$$

In the above equation, each task τ_i is evaluated over its period, up to its deadline D_i. The summation of the workload is evaluated over the interval $(0, D_i]$. If the cumulative workload's minimum value normalized by time is not greater than unity at any time t such that $0 < t \leq D_i$, then the task set is schedulable. In practice, the schedulability condition is easily evaluated using an iterative technique discussed in Sha and Sathaye (1992).

Equation 3 assumes that: the scheduler instantaneously observes all task arrivals; tasks are perfectly preemptable; and tasks are never delayed by lower priority tasks. However scheduling in a network is different from scheduling in a centralized environment. In a network, distributed scheduling decisions must be made with incomplete information. Stations in a network are not always guaranteed to have a consistent view of the network state. Due to this, a high-priority message at any station may be delayed by low-priority messages. The challenge is to achieve predictability under these circumstances.

A generic scheduling model can be developed based on the scheduling condition in Equation 3. To achieve this, the schedulability conditions must be modified as follows. C_j is replaced by $C_j + Overhead_j$ which is the total amount of time task τ_j occupies the resource every period. $Overhead_j$ represents the additional time that must be spent on behalf of the task, and includes the packetization delay, the time to transmit the source address, destination address, the overhead in waiting for an acknowledgement. The sources

of overhead in a network are due to packetization of messages, waiting for a token, and any system level overhead due to unsynchronized clocks (Kopetz and Ochsenreiter 1987). The term $Overhead_j$ in the generic scheduling model, which represents the total overhead in transmitting C_j units of information per period, consists of the sum of the header (C_{header}), trailer ($C_{trailer}$), and the acknowledgment (C_{ack}), multiplied by the number of packets that are needed to transmit C_j units of information. $Overhead_j$ can be written as:

$$Overhead_j = (C_{header} + C_{trailer} + C_{ack}) \left\lceil \frac{C_j}{P_{\max} - (C_{header} + C_{trailer})} \right\rceil \tag{4}$$

Where, $P_{\max}$ denotes the maximum sized packet on the network. Note that $P_{\max}$ contains the header and trailer in addition to the information. We also need to add two new terms $Overhead_{sys_i}$ and $Blocking_i$ to the schedulability conditions. $Overhead_{sys_i}$ captures task independent system level overhead encountered on some resources at priority level i. An example of this is O_{clock} the penalty due to unsynchronized clocks (Kopetz and Ochsenreiter 1987).

Equation 5 shows a generic *scheduling model* for a system. The detailed development of a network scheduling model is given in Sathaye (1993).

$$\begin{aligned} &\forall i, \quad 1 \le i \le n, \\ &\min_{0<t\le D_i} \sum_{j=1}^{i} \frac{C_j + Overhead_j}{t} \left\lceil \frac{t}{T_j} \right\rceil + \frac{Overhead_{sys_i}}{t} + \frac{Blocking_i}{t} \le 1 \end{aligned} \tag{5}$$

The IEEE 802.5 protocol in ETR mode permits transmission of a new packet before a previous packet has been acknowledged. In these cases C_{ack} is zero. In this case, the scheduling model is a t-schedulability model which checks only whether messages can be transmitted by their deadline. End-to-end latency of the message on the link can be determined using Equation 1. Satisfaction of the end-to-end deadline can be determined using Equation 2.

$Blocking_i$ captures the time task τ_i is delayed by lower priority tasks. This is also called *priority inversion* (Sha, Rajkumar, and Lehoczky 1990). Blocking can occur in a network due to several reasons. One source of blocking is that packets are non-preemptable. The network may either support fixed size or variable size packets. Without loss of generality we can assume that the maximum size packet on the network is denoted by $P_{\max}$. Hence a high-priority packet may be delayed for as long as the transmission time of a maximum sized lower-priority packet. Another reason for blocking is that the arrival of a high-priority packet is not immediately detected by all stations. This can lead to blocking. This blocking is due to imperfect global scheduling, and can be denoted as B_{gs}.

Finally blocking can occur since priority levels are insufficient. This happens when a set of message streams with a large number of *natural priorities* is scheduled on a network that supports a smaller number of priority levels. The natural-priority level is the priority that would have been assigned to the connection on a system with sufficient priority levels. With insufficient priority levels, a high natural-priority activity is grouped into the same level as a lower natural-priority activity and hence cannot be differentiated by the scheduler. Multi-access communication networks typically support very few priority levels. For example,

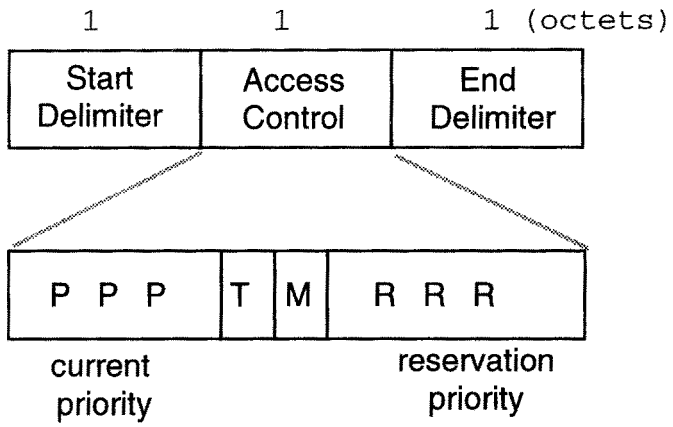


Figure 1. IEEE 802.5 token format.

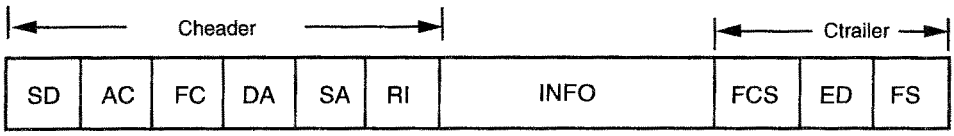
IEEE 802.5 token ring supports only eight priority levels and IEEE 802.6 DQDB supports only four levels. Blocking due to insufficient priority levels can be denoted as B_l, and is addressed below. A smaller number of priority levels than required by the scheduling algorithm causes a potential loss in schedulability. The impact of limited priority levels on the utilization based schedulability test (Liu and Layland 1973) has been discussed by Lehoczky and Sha (1986). The impact of limited priorities on the time-based schedulability test of Lehoczky, Sha, and Ding (1989) is discussed in Sathaye, Katcher, and Strosnider (1992). This is a technique to quantify the schedulability loss due to limited priorities for a particular connection set on a network with a known number of priority levels. The technique finds that grouping of tasks into a priority level, that minimizes the blocking due to limited priorities. We denote this blocking B_l, and use it in the rest of this paper.

Considering the blocking contributions of non-preemptability of the network, effects of global priority arbitration, and limited priority levels, the $Blocking_i$ term of the generic scheduling model can be written as:

$$Blocking_i = P_{\max} + B_{gs} + B_l \tag{6}$$

3. Token Ring Operation and Priority Arbitration

We briefly describe the aspects of the token ring protocol that are relevant in the development of the scheduling model. We do not discuss several issues such as ring initialization and maintenance. A comprehensive, definitive description of the protocol can be found in the standard document (IEEE 1992). The operation of the ring is based on the use of a special packet called the token to select the next message to be transmitted. The token format is shown in Figure 1 and the packet format is shown in Figure 2.



SD = Starting Delimiter(1 octet)
AC = Access Control (1 octet)
FC = Frame Control (1 octet)
DA = Destination Address (2 or 6 octets)
SA = Source Address (2 or 6 octets)
RI = Routing Information
(0 to 30 octets)

INFO = Information
(0 or more octets)
FCS = Frame Check Sequence
(4 octets)
ED = End Delimiter (1 octet)
FS = Frame Status (1octet)

Figure 2. IEEE 802.5 packet format.

The token has three functions. First, it represents permission to use the medium. A *free* token may be captured by a qualified station so that it can transmit a message. Secondly, a *busy* token is used as a message header. After capturing a free token, a station changes it to a busy token by setting the **T** bit. It then retransmits it and appends a packet behind the token. The third function of the token is global priority arbitration. The ring supports prioritized operation using the priority (PPP) and reservation (RRR) fields within the tokens as follows: As discussed above, upon claiming a free token, the transmitting station appends its packet behind the claimed token. Each station examines the reservation (RRR) field of the busy token as it passes. If the reservation field is of lower priority than the priority of the station's pending packet, the contents of the reservation field are updated with the pending packet's priority.

CTR and ETR modes differ in terms of releasing the free token. In CTR mode, the transmitting station waits until both, its claimed token and packet return (indicated by a match between the station's address and the received packet's **SA** field) and then transmits a free token. The priority of the released token is the priority in the returned RRR field. Subsequent stations do not capture free tokens unless their pending packet's priority is greater than or equal to the free token's priority.

In ETR mode, the station releases a free token as soon as it finishes transmitting its packet, independent of the time of the packet's return. The priority used for tokens released prior to receiving the packet's header will be the priority in the RRR field of the most recently received claimed token. A complete description of the protocol can be found in (IEEE 1992).

We now define some additional terms that will be used in the analysis. Assume a 16 Mb/s token ring, 6 octets each for DA and SA and 0 octets for the RI field. In a 16 Mb/s token ring it takes 0.5 μs to transmit one octet.

- W_T : The *walk-time*, which is the amount of time it takes to perambulate the ring. This is the sum of station delay, propagation delay, and a latency buffer. The station delay

(D_B) is two bit times and the latency buffer (*Lat_Buff*) is 39 bit times for a 16 Mb/s ring. The walk time is directly related to the distance spanned by the ring by the following expression given in (IEEE 1992).

$$W_T = \frac{(n-1)D_B + Lat\_Buff}{B_L} + \frac{Media_L}{Media_{ps}} \tag{7}$$

where B_L is the transmission bit rate (4 Mb/s or 16 Mb/s), $Media_L$ is the length of the ring, and $Media_{PS}$ is the propagation speed (usually half the speed of light for copper cable).

- C_{token} : The time to transmit the token. The length of the token is 3 octets. In units of transmission time, $C_{token} = 3 * 0.5 = 1.5\mu s$.
- C_{SA} : The length of the packet up to the last bit of the SA field. As shown in Figure 2, this consists of the SD, AC, FC, DA, and SA fields. This is 15 octets long assuming 6 octet DA and SA fields. In units of transmission time, $C_{SA} = 15 * 0.5 = 7.5\mu s$.
- C_{header} : The packet *header*, which as shown in Figure 2, is the length of all fields prior to but not including the INFO field, in terms of transmission time.
- $C_{trailer}$: The packet *trailer* which, as shown in Figure 2, is the length of all fields after the INFO field, in terms of transmission time.

3.1. Assumptions

Before developing the scheduling model we describe the assumptions under which the model will be valid. We assume the following:

- A set of periodic connections $\tau_1, \tau_2, \ldots, \tau_n$ arranged in decreasing priority order exists in the network. Unless otherwise specified, deadlines are at the end of the period.
- Each station buffers packets in priority order.
- A lower-priority packet at the head of a station's priority queue, is replaced if a higher-priority packet becomes ready to transmit from the station. Some implementations of the IEEE 802.5 protocol do not permit a low-priority packet at the head of the priority queue to be replaced, even when a higher-priority packet becomes ready to transmit at the station. In this case, unbounded blocking can occur, since the station makes a reservation at the lower-priority, even though a higher-priority packet is waiting, and can be prevented from transmitting by an unbounded amount of medium priority transmissions.
- The stacking mechanism is not taken into account.
- The ring is in single-packet per token mode and hence each station transmits only one packet on every token arrival.
- No fault occurs, consequently, no recovery procedure is started.

3.2. *Schedulability Analysis: CTR*

We now use the protocol description to develop a scheduling model for the CTR protocol. We develop expressions for $Overhead_j$ and $Blocking_i$. We then use these terms in an overall scheduling model.

The model can be used to determine if a set of connections in a CTR 802.5 network meets its deadlines. The deadline of each message of the connection is defined as the maximum allowable difference between the time when all its packets reach the destination and the time when the message becomes ready to transmit at the source.

3.2.1. *Message Transmission and Overhead*

In this section we quantify both the transmission time of a periodic connection τ_j and any associated overheads. In the initial analysis we assume that C_j, the amount of information that needs to be transmitted per period, can fit in a single packet.

We wish to develop an expression for the total bandwidth demand per period. This is equal to $C_j + Overhead_j$, the demands due to information and the overhead that must be transmitted. Each station's opportunity to transmit consists of the arrival and capture of a free token, transmission of its packet, propagation of the packet around the ring, and transmission of a free token. As explained in Section 3, the transmitting station can not release a free token until it observes the last bit of the SA of its own packet. All of the above actions except the transmission of the information C_j can be considered to be *overhead.*

The time to capture the token even when the station has the highest priority message, and a correct reservation has been made in a previous claimed token, can be as large as the walk-time, W_T. This occurs in the worst case the token may have been released by the same station as the one which captures it[1]. After capturing the free token, a packet of length $C_{header} + C_j + C_{trailer}$ is transmitted. We will refer to the sum of the header and trailer as C_{enc}, the packet encapsulation. The transmitting station can not release a free token until it observes the last bit of the SA of its own packet. The time for the packet to propagate around the ring is given by W_T. An additional C_{SA} units of time must be spent before the address can be recognized.

If the packet transmission time $C_j + C_{enc} \geq W_T + C_{SA}$, the transmitter will have recognized its own packet on the ring prior to the end of packet transmission and a free token can be released as soon as packet transmission ends. If the packet transmission time $C_j + C_{enc} < W_T + C_{SA}$, the token can not be transmitted until $W_T + C_{SA}$ units of time have elapsed since the beginning of transmission. The transmission of the free token takes C_{token} units of time. Therefore,

$$C_j + Overhead_j = W_T + \max\{(C_j + C_{enc}), (W_T + C_{SA})\} + C_{token} \tag{8}$$

A different form of the expression for $C_j + Overhead_j$ may be written as:

$$C_j + Overhead_j = \begin{cases} C_j + C_{enc} + W_T + C_{token} & \text{when } W_T + C_{SA} \leq C_j + C_{enc} \\ 2W_T + C_{SA} + C_{token} & \text{when } W_T + C_{SA} > C_j + C_{enc} \end{cases} \tag{9}$$

We have assumed so far that the information to be transmitted per period (C_j) can be sent in a single packet, i.e $C_j + C_{enc} \leq P_{\max}$. Releasing that assumption, if $C_j + C_{enc} > P_{\max}$, then C_j must be split into multiple maximum size packets, except the last packet which may be smaller than $P_{\max}$. The total bandwidth required in one period $C_j + Overhead_j$, under these conditions is given by:

$$C_j + Overhead_j = \begin{cases} C_j + \left\lceil \frac{C_j}{P_{\max} - C_{enc}} \right\rceil (C_{enc} + W_T + C_{token}) & \text{when } W_T + C_{SA} \leq \min(C_j + C_{enc}, P_{\max}) \\ \left\lceil \frac{C_j}{P_{\max} - C_{enc}} \right\rceil (2W_T + C_{SA} + C_{token}) & \text{when } W_T + C_{SA} > \min(C_j + C_{enc}, P_{\max}) \end{cases}$$

Observe that the C_{ack} term discussed in Section 2 is equal to the sum of the walk-time and C_{token}. An interesting point to note is that when $W_T + C_{SA} > C_j + C_{enc}$, the time to transmit information C_j does not appear in the expression.

Furthermore, due to a lack of synchronized clocks it is possible that the destination may have a clock that shows a later time than the source. Hence we can add a system level overhead component O_{clock}.

3.2.2. *Blocking Effects*

In this section we investigate the effects of blocking, by quantifying the time a packet at a station is potentially delayed by transmission at a lower priority. Section 2 gave a generic expression for $Blocking_i = P_{\max} + B_{gs} + B_l$. We will consider the effects of limited priorities when we consider an example application of the model. Since packets are non-preemptable, blocking equal to a maximum sized packet is unavoidable. Due to imperfections in global priority arbitration, additional blocking is incurred (B_{gs}) as shown below. First we restate an example given in Pleinevaux (1992) to give an intuitive understanding of the phenomenon.

Example 1. Consider a network with three stations S_M, S_L, and S_H as shown in Figure 3. Let each station be a source of periodic packets such that S_M, S_L and S_H generate medium, low and high-priority packets respectively.

- At time t_0 let station S_M be holding the token and transmitting a maximum size packet.
- Let a high-priority packet arrive at S_H just after the reservation field of S_M's packet has passed S_H, preventing it from making a high-priority reservation.
- After transmitting its packet and recognizing it on the ring as discussed in Section 3.2.1, S_M releases a free token.
- Let S_L have a packet to transmit. The token is captured at S_L, which also transmits a maximum size packet.
- Station S_H makes a high-priority reservation in S_L's packet. S_L releases the token with S_H's priority after transmitting and observing S_L's packet.
- S_H captures the token and transmits. □

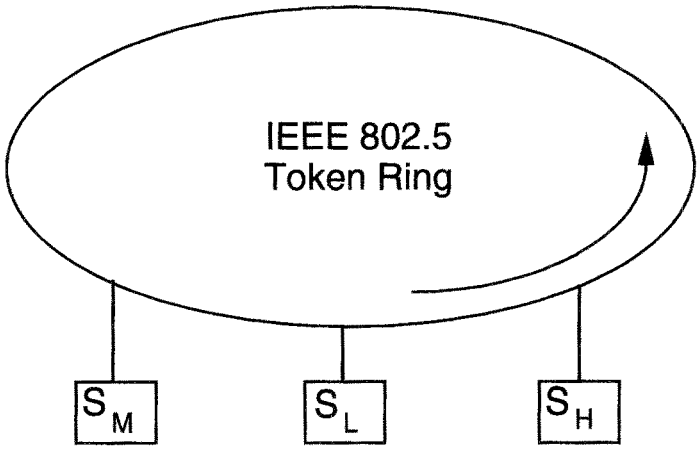


Figure 3. Blocking in IEEE 802.5 token ring: CTR protocol.

In the above example, a high-priority packet at station S_H is delayed by the transmission and overhead of two lower-priority packets. The result of the above example can be generalized to the case where there are multiple stations between S_M and S_H of priority lower than that of S_H:

THEOREM 1 *Given the assumptions of Section 3.1, The worst case blocking in an IEEE 802.5 token ring network operating in CTR mode is bounded by*

$$Blocking_i \leq \begin{cases} 2(P_{\max} + C_{token}) + W_T & when\ W_T + C_{SA} \leq P_{\max} \\ 2(W_T + C_{SA} + C_{token}) + W_T & when\ W_T + C_{SA} > P_{\max} \end{cases} \quad (10)$$

Proof: Let a high priority message from stream τ_i become ready to transmit at station S_i at time t_0. Due to the priority arbitration mechanism, τ_i can be blocked only by lower-priority transmissions that start before S_i can make a reservation in a claimed token. We need to consider two cases based on the *first* transmission S_i observes on the ring after t_0:

Case 1: S_i observes a free token.

- If the free token's priority is not higher than the priority of τ_i, the station captures the free token and transmits, and τ_i is not blocked.
- If the free token's priority is higher than the priority of τ_i, S_i does not capture the token. The token can only be captured by a station with a higher-priority message than τ_i. Hence the next packet transmitted is guaranteed to be of higher priority. S_i can make a reservation in the claimed token of this transmission, and no lower-priority transmissions can occur before a reservation can be made and τ_i is not blocked.

Case 2: S_i observes a packet being transmitted.
Let the time at which the reservation field of the packet being transmitted is observed by S_i be denoted t_r. Two cases need to be considered.

Case 2a: $t_r \geq t_0$
The reservation field of the transmission in progress is observed by S_i after τ_i becomes ready to transmit. S_i makes a reservation in the field, and no new lower-priority transmissions can occur. However, the transmission currently in progress may be of lower priority and of maximum size. In this case, the worst case blocking is given by the bandwidth demands of transmitting a maximum size packet.

Case 2b: $t_r < t_0$
S_i can not make a reservation in the reservation field of the current transmission, and a new lower-priority transmission can occur. The reservation field of this lower-priority transmission must be observed by S_i after t_0. S_i makes a reservation in this field and no further lower-priority transmissions can occur. τ_i can be blocked by the transmission of the packet currently in progress and an additional packet transmission. Since both packets may be of lower priority and of maximum size, the worst case blocking in this case is given by two times the bandwidth demands of transmitting a maximum size packet, with the difference that the sum of the token acquisition times for both transmissions is only one walk-time, (since the free token can rotate at most once around the ring before the high-priority transmission).

The bandwidth demands of transmitting a maximum size packet can be obtained by replacing $C_i + C_{enc}$ in Equation 9 by $P_{\max}$ as given below:

$$P_{\max} + Overhead_{\max} = \begin{cases} P_{\max} + W_T + C_{token} & \text{when } W_T + C_{SA} \leq P_{\max} \\ 2W_T + C_{SA} + C_{token} & \text{when } W_T + C_{SA} > P_{\max} \end{cases}$$

In all cases the worst case blocking is bounded by two times the bandwidth demands of transmitting a maximum size packet with the modification that the sum of the token acquisition times for both transmission is W_T. Therefore, blocking is bounded by Equation 10. ■

3.3. *Schedulability Analysis: ETR*

The ETR option increases the available bandwidth and improves the data transmission efficiency of the token ring protocol (IEEE 1992). It allows the transmitting station to release a free token as soon as it completes transmission even when the station has not yet received its packet back from the ring. This tends to reduce overhead since station does not have to wait for acknowledgment. The priority used for tokens released prior to receiving the packet's header will be the priority of the most recently received packet. In this case the transmitting station may not be aware of a high-priority reservation when it releases a new free token. It may release the token at a lower priority, causing greater priority inversion than with CTR. Hence B_{gs} contribution to blocking is greater for ETR than for CTR. The ability to release the token as soon as packet transmission ends, increases the throughput

of the ring. However this increased throughput comes at the cost of increased worst-case priority inversion in the ETR protocol.

The model can be used to determine if a set of connections in a ETR 802.5 network meet their deadlines. Since the ETR protocol permits multiple packets on the ring, the transmission of a particular packet can begin before the previously transmitted packet has reached its destination. The ETR scheduling model can be used to check if each message is transmission schedulable, i.e whether it can be transmitted by the end of a certain deadline. It is only a *t-schedulability* model, and end-to-end schedulability can be determined as discussed in Section 2.

3.3.1. Message Transmission and Overhead

Since the token is released immediately after transmission, the time to transmit C_j amount of information is given by the same expression as the one in Equation 9 when $W_T + C_{SA} \leq \min(C_j + C_{enc}, P_{\max})$.

$$C_j + Overhead_j = C_j + \left\lceil \frac{C_j}{P_{\max} - C_{enc}} \right\rceil (W_T + C_{token} + C_{enc}) \tag{11}$$

As before if $C_j + C_{enc} > P_{\max}$, the information is sent in multiple maximum size packets. In the case when walk-time is larger than the packet transmission time, the last packet is not guaranteed to have reached its destination by the time the token is released. As discussed before the scheduling model is a t-schedulability model since it can test only whether the entire message is transmitted by its deadline.

If the end to end deadline of connection τ_i is E_i, then the end-to-end deadline is satisfied if the following condition is true:

$$E_i \geq D_i + D_{prop_i} \tag{12}$$

where D_{prop_i} is the propagation delay between the source and destination of connection τ_i.

Furthermore, as with CTR we add a system level overhead component $Overhead_{sys_i} = O_{clock}$ in the scheduling model to account for clock synchronization effects.

3.3.2. Blocking Effects

We now consider the effects of blocking when ETR is implemented. We discussed in Section 3.2.2 that the distributed nature of the system makes the global priority arbitration imperfect, and blocking can occur.

The reservation fields in a busy token are used to indicate the arrival of high-priority traffic at a station. As shown in Section 3.2.2, blocking occurs if the reservation field passes the station just prior to high-priority traffic arrival. This effect can also occur when ETR is enabled. In addition to this, since a station releases a free token immediately after transmission, if a high-priority reservation is not observed by the transmitting station before generating a free token, another low-priority transmission can occur. In fact, in the worst

case, a high-priority connection in the network may have to wait for transmissions from every lower-priority station.

In order to simplify the discussion we assume that station S_i is the source of message stream τ_i, where $\{\tau_1, \tau_2, \ldots, \tau_n\}$ are in decreasing priority order. A message of stream τ_i can be potentially blocked by messages from streams $\{\tau_{i+1}, \ldots, \tau_n\}$. We will show that a message at station S_i can be blocked by all lower-priority messages from stations between the station currently transmitting and S_i. This worst case blocking occurs when each station S_i has exactly one connection τ_i as we will discuss in the proofs below. In order to compute the bandwidth demands of intermediate stations, it would be unrealistic to assume that each takes an entire W_T to acquire the free token. In fact, each station's token acquisition time is the propagation delay between the previous station and itself. In the worst case, the sum of token acquisition times of all stations is bounded by W_T. To minimize unnecessary complexity in the expression for blocking, we assume that there are *n equally spaced stations* in the network, giving a walk-time of W_T/n between stations.

THEOREM 2 *Given the assumptions of Section 3.1, The worst case blocking in an IEEE 802.5 token ring network operating in ETR mode is bounded by*

$$Blocking_i \leq \begin{cases} 2P_{\max} + (n-i)C_{token} + (n-i-1)W_T - iW_T/n & when\ W_T < P_{\max} \\ (n-i)(P_{\max} + C_{token}) + (n-i)W_T/n & when\ W_T \geq P_{\max} \end{cases} \quad (13)$$

where station S_i is the source of a single connection τ_i.

Proof: Let a high priority message from stream τ_i become ready to transmit at station S_i at time t_0. We wish to quantify the maximum delay experienced by τ_i due to lower-priority transmissions. Due to the priority arbitration mechanism τ_i can be blocked only by lower-priority transmissions that meet the following conditions:

τ_i can be blocked by lower-priority transmissions that occur before S_i can make a reservation in a claimed token and the reservation is observed by the transmitting station before it generates a free token. Observe that this is in contrast to the CTR case where it was only necessary to make a reservation to prevent further blocking.

We need to consider two cases based on the *first* transmission S_i observes on the ring after t_0:

Case 1: S_i observes a packet being transmitted.
Let the time at which the reservation field of the packet being transmitted is observed by S_i be denoted t_r. Two cases need to be considered.

Case 1a: $t_r < t_0$
S_i can not make a reservation in the reservation field of the current transmission. Therefore a new lower-priority transmission can occur. The reservation field of this lower-priority transmission must be observed by S_i at a time after t_0. S_i makes a reservation in this field.

- If the reservation field is observed by the transmitting station (say S_{i+1}), before generating a free token, no further lower-priority transmissions can occur as in the CTR

case. τ_i can be blocked by the transmission of the packet currently in progress and an additional packet transmission. Since both packets may be of lower priority and of maximum size, the worst case blocking in this case is given by two times the bandwidth demands of transmitting a maximum size packet.

- If the reservation field is not observed by the transmitting station before generating a free token, a lower-priority station, (say S_{i+2}), between S_{i+1} and S_i can transmit. As before, S_i makes a reservation in this claimed token and depending on whether or not S_{i+2} observes the reservation before releasing a free token, a lower priority station between S_{i+2} and S_i can transmit. In fact in the worst case each station S_{i+1} through S_n can transmit a lower priority packet, before S_i can transmit. In the worst case, station S_{i+1} transmits a maximum size packet, S_{i+2} through S_{n-1} transmit a packet of size such that the transmitting station does not observe the reservation before releasing the token. This size is bounded by $P_{\max}$ if $P_{\max} \leq W_T$ and by W_T otherwise. Finally S_n can transmit a maximum size packet.

Hence, the blocking is equal to two times bandwidth demands of transmitting a maximum size packet plus $(n-i-2)$ times worst case demands of transmitting a packet of size $P_{\max}$ so that transmitting station transmits a free token before observing the reservation. The expression for bandwidth demand is similar to that given in previous sections except that the time to acquire a token is W_T/n rather than W_T, and the blocking is given by

$$Blocking_i \leq \quad (14)$$
$$\begin{cases} 2(P_{\max} + C_{token}) + (n-i-2)(W_T + C_{token}) + (n-i)W_T/n & \text{when } W_T < P_{\max} \\ 2(P_{\max} + C_{token}) + (n-i-2)(P_{\max} + C_{token}) + (n-i)W_T/n & \text{when } W_T \geq P_{\max} \end{cases}$$

which reduces to Equation 13.

Case 1b: $t_r > t_0$
In this case S_i can make a reservation in the packet transmitted by S_{i+1}.

- If S_{i+1} observes the reservation field before releasing a free token then blocking is limited to the bandwidth demands of transmitting a single maximum size packet.

- Consider the situation when S_{i+1} does not observe the reservation before it releases a free token. The only way this can occur is if the packet transmitted by S_{i+1} is not larger than $\min(P_{\max}, W_T)$. Other than this restriction, a condition similar to that of *Case 1a* can occur. This gives the following expression for blocking.

$$Blocking_i \leq$$
$$\begin{cases} (P_{\max} + C_{token}) + (n-i-1)(W_T + C_{token}) + (n-i)W_T/n & \text{when } W_T < P_{\max} \\ 2(P_{\max} + C_{token}) + (n-i-2)(P_{\max} + C_{token}) + (n-i)W_T/n & \text{when } W_T \geq P_{\max} \end{cases}$$

which is not greater than that given in Equation 14.

Case 2: S_i observes a free token.

- If the free token's priority is not higher than the priority of τ_i, the station captures the free token and transmits. Therefore τ_i is not blocked.
- If the free token's priority is higher than the priority of τ_i, S_i does not capture the token. The token can only be captured by a station with a higher-priority message than τ_i. The next packet transmitted is guaranteed to be of higher priority. S_i can make a reservation in the claimed token of this transmission. The blocking in this case is similar to *Case 1b* and therefore not greater than in Equation 13.

The worst case blocking is bounded by that in Equation 13, and the theorem follows. ■

The form of the scheduling model for the ETR protocol is identical to the model for the CTR protocol. The differences in overhead and priority arbitration and worst case arrival phasing have been incorporated in to Equation 11 and Equation 13.

This consistent model form will allow us to easily compare the behavior of the two protocols in Section 3.5.3. The scheduling model and $Overhead_j$ and $Blocking_i$ parameters for CTR and ETR are summarized in Table 1.

3.4. Maximum Achievable Utilization

In the previous sections we have developed scheduling models for both CTR and ETR token ring protocols. Before we compare the real-time performance of the two protocols, we develop expressions for maximum achievable utilization without regard to real-time performance. Let there be n equally spaced stations in the network and let each station always have a maximum size packet ready to transmit.

3.4.1. Utilization with CTR Protocol

Case 1: $W_T + C_{SA} \leq P_{\max}$

Since stations are equally spaced and always have a packet to transmit, the time to capture a free token can be as large as W_T/n. After capturing the free token, a packet of length $P_{\max}$ is transmitted. Since $W_T + C_{SA} \leq P_{\max}$ the station can release a free token immediately after ending packet transmission. The time to transmit a free token is C_{token}. A packet of size $P_{\max}$ is transmitted every cycle of $(W_T/n) + P_{\max} + C_{token}$, so the utilization is:

$$U_{\max} = \frac{P_{\max}}{(W_T/n) + P_{\max} + C_{token}} \tag{15}$$

Case 2: $W_T + C_{SA} > P_{\max}$

As in case 1, the time to capture a free token can be as large as W_T/n. After capturing the free token, a packet of length $P_{\max}$ is transmitted. Since $W_T + C_{SA} > P_{\max}$ the station

Table 1. IEEE 802.5 scheduling model summary.

Scheduling Model

$$\forall\, i = 1, 2, \ldots, n \ \min_{0<t\le D_i} \sum_{j=1}^{i} \frac{C_j + Overhead_j}{t} \left\lceil \frac{t}{T_j} \right\rceil + \frac{Overhead_{sys_i}}{t} + \frac{Blocking_i}{t} \le 1$$

CTR Parameters

$C_j + Overhead_j$	$\begin{cases} \text{when } W_T + C_{SA} \le \min(C_j + C_{enc}, P_{\max}) \\ C_j + \left\lceil \frac{C_j}{P_{\max} - C_{enc}} \right\rceil (W_T + C_{token} + C_{enc}) \\ \text{when } W_T + C_{SA} > \min(C_j + C_{enc}, P_{\max}) \\ \left\lceil \frac{C_j}{P_{\max} - C_{enc}} \right\rceil (2W_T + C_{SA} + C_{token}) \end{cases}$
$Overhead_{sys_i}$	O_{clock}
$Blocking_i$	$\begin{cases} 2(P_{\max} + C_{token}) + W_T + B_l & \text{when } W_T + C_{SA} \le P_{\max} \\ 2(W_T + C_{SA} + C_{token}) + W_T + B_l & \text{when } W_T + C_{SA} > P_{\max} \end{cases}$

ETR Parameters

$C_j + Overhead_j$	$C_j + \left\lceil \frac{C_j}{P_{\max} - C_{enc}} \right\rceil (W_T + C_{token} + C_{enc})$
$Overhead_{sys_i}$	O_{clock}
$Blocking_i$	$\begin{cases} \text{when } W_T < P_{\max} \\ 2P_{\max} + (n-i)C_{token} + (n-i-1)W_T - iW_T/n + B_l \\ \text{when } W_T \ge P_{\max} \\ (n-i)(P_{\max} + C_{token}) + (n-i)W_T/n + B_l \end{cases}$

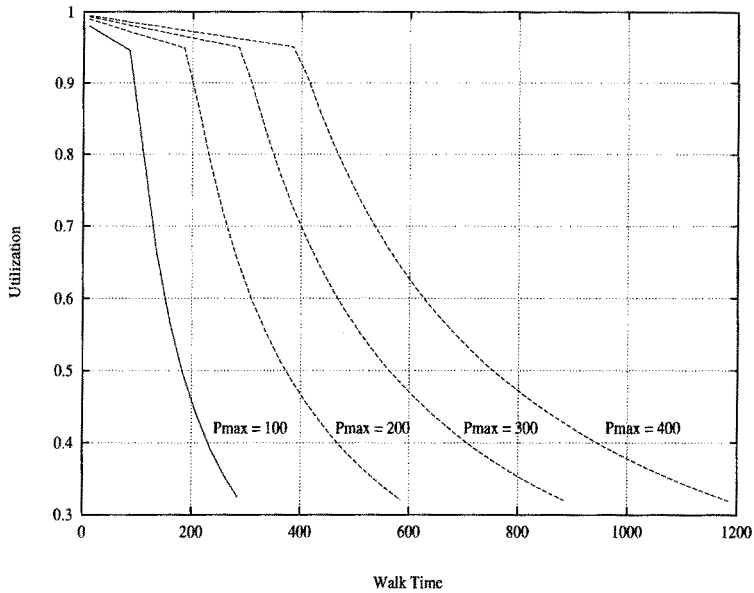


Figure 4. CTR protocol: utilization vs. W_T for various $P_{\max}$ values.

can not release a free token after transmission but must wait for the packet to return and be recognized. Therefore the elapsed time after beginning transmission before a free token can be released is given by $W_T + C_{SA}$. The time to transmit a free token is C_{token}. A packet of size $P_{\max}$ is transmitted every cycle of $(W_T/n) + W_T + C_{SA} + C_{token}$, so the utilization is:

$$U_{\max} = \frac{P_{\max}}{(W_T/n)) + W_T + C_{SA} + C_{token}} \tag{16}$$

The maximum achievable utilization as function of the walk-time is plotted in Figure 4. Note that we have assumed $n = 10$. Observe that the utilization is high when the packet transmitted is not larger than the walk-time. For a large network, (or equivalently a small $P_{\max}$) bandwidth is wasted while stations wait before releasing the token. As expected the utilization curve has a knee at $W_T = P_{\max} - C_{SA}$ after which the utilization decreases rapidly.

3.4.2. Utilization with ETR Protocol

With ETR, each station releases the token as soon as packet transmission ends. The expression for utilization will be same as Equation 15 for all values of $P_{\max}$ and W_T.

As before, we plot the maximum achievable utilization as function of the walk-time for $n = 10$ (Figure 5). In this case, the utilization is high for smaller walk-times and decreases slowly as walk-time increases due to increased time spent in token propagation.

Table 2. Maximum achievable utilization.

CTR Protocol	$U_{\max} = \begin{cases} \frac{P_{\max}}{(W_T/n)+P_{\max}+C_{token}} & \text{when } W_T + C_{SA} \leq P_{\max} \\ \frac{P_{\max}}{(W_T/n)+W_T+C_{SA}+C_{token}} & \text{when } W_T + C_{SA} > P_{\max} \end{cases}$
ETR Protocol	$U_{\max} = \frac{P_{\max}}{(W_T/n)+P_{\max}+C_{token}}$

In summary, the maximum achievable utilization of the two protocols is identical when the packet size is not smaller than the walk-time, while the ETR protocol can achieve significantly better utilization than the CTR protocol when the $W_T > P_{\max}$. The utilization expressions are summarized in Table 2. As usual $P_{\max}$ is in units of transmission time in microseconds.

3.5. *Applying the IEEE 802.5 Scheduling Model*

In this section we demonstrate the usefulness of the scheduling models developed for the CTR and ETR token ring protocols by applying the analysis of Sections 3.2 and 3.3 to a particular system. We first describe the system under consideration. Then we demonstrate techniques to reason about system parameter selection, such as maximum packet size. Finally, we compare the schedulability performance of both the CTR and ETR protocols. The results in this section are specific to the connection set studied. In fact, the strength of the scheduling model is that it permits us to make statements about the behavior of the network for given connection sets rather than on average. It must be noted that the results obtained here can not be generalized to all types of traffic. The scheduling model must be viewed as a tool that can be used to analyze the behavior of specific given connection-sets.

Two different protocols can be compared from the schedulability viewpoint by using a measure called the *degree of schedulable saturation* (Sathaye 1993). It represents the degree to which the system is saturated from a schedulability viewpoint. A smaller $S_{\max}$ indicates greater remaining high-priority schedulable capacity. A particular scheduling situation is better if it results in a smaller $S_{\max}$. We define $S_{\max}$ as follows:

$$S_{\max} = \max_{1 \leq i \leq n} Saturation_i \tag{17}$$

$$Saturation_i = \min_{0 < t \leq D_i} W_i(t)/t \tag{18}$$

where $W_i(t)$ is the bandwidth demands up to time t by connection τ_i and higher-priority connections. $W_i(t)$ is given by:

$$W_i(t) = \sum_{j=1}^{i} (C_j + Overhead_j)\lceil t/T_j \rceil + Overhead_{sys_i} + Blocking_i \tag{19}$$

Given that $S_{\max} = Saturation_i$ for some $1 \leq i \leq n$, τ_i is called the *limiting connection.* If $S_{\max}$ is unity no new connections with priority greater than or equal to that of the limiting connection can be scheduled. If $S_{\max}$ is greater than unity the system is unschedulable and we set $S_{\max}$ to infinity. In this paper we primarily use the degree of schedulable saturation as a figure of merit.

Table 3 summarizes the requirements of two example connection sets that we will use to compare CTR and ETR scheduling. The example is patterned after the message set discussed in Strosnider (1988) which considers a sonar system. There are 10 connections ($n = 10$) on the network. There are two concurrent operating modes, one at 12 Hz (T = 76900 μs), and another at 13 Hz (T = 83300 μs). In addition, there are two higher frequency connections that broadcast the position and orientation of the mobile system. We call this connection set *Connection Set-1.* By transforming the periods of the broadcast connections to 75000 and 80000 respectively we construct *Connection Set-2.* We will use both connection sets to compare CTR and ETR scheduling.

Some clarification on the units in the following table is useful. The message length is in units of transmission time expressed in microseconds. For example a message length of 28 μs corresponds to a message of length 56 bytes. This is true in a 16 Mb/s network which takes 0.5μs to transmit one byte. The period is in microseconds. Each connection has an end-of-period deadline; i.e it is required to reach its destination station on the ring by the end of its period.[2] Note that the scheduling model for ETR is a t-schedulability model, while the CTR model is an end-to-end schedulability model. In order to compare the two protocols, we modify the deadline of each connection τ_i to $D_i = T_i - W_T$ when calculating t-schedulability in ETR mode.

In order to compare the protocols, we assume that they may exist on large networks. Since W_T is a parameter of the scheduling model, we convert distance to walk-time using Equation 7. The values for the parameters of Equation 7 are given in Table 4.

Using Equation 7 and the values in Table 4 we can compute the walk-time W_T for various distances.

3.5.1. *Impact of Limited Priority Levels*

We now consider the impact of limited priority levels on the schedulability of the connection sets in the example. Assume that out of the 8 available priority levels in an IEEE 802.5 token ring network, this particular token ring management software devotes the highest 4 levels to real-time traffic and the lower to non-real-time traffic. Hence the above connection-set must be mapped to four priority levels. Observe that the above connection-sets have five distinct periods. Therefore with a rate monotonic priority assignment, five priority levels would be necessary.

The problem of scheduling tasks with n distinct *natural*[3] priorities on a system with m priority levels when $m < n$ has been considered by Sha, Rajkumar, and Lehoczky (1991) and Sathaye, Katcher, and Strosnider (1992). As discussed in Section 2 a limited number of priority levels may result in increased blocking because the grouping of a lower natural priority task with a higher natural priority task will result in the blocking of the higher priority task. In the specific example considered here the connection with a period of

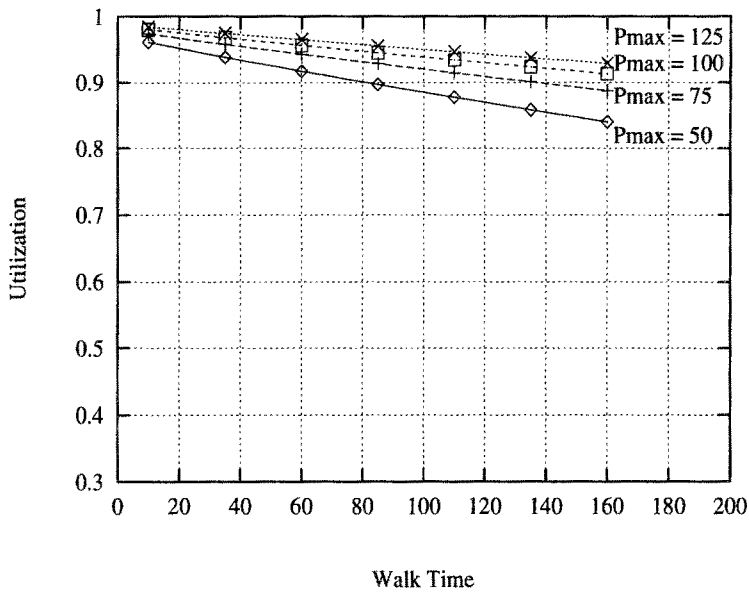


Figure 5. ETR protocol: utilization vs. W_T for various P_{max} values.

Table 3. Connections on an IEEE 802.5 network.

Priority	**Connection Set-1**		**Connection Set-2**	
	Msg Length C_j	Period T_j	Msg Length C_j	Period T_j
1	28	2500	840	75000
2	50	40000	100	80000
3	1382	76900	1382	76900
3	1049	76900	1049	76900
3	996	76900	996	76900
3	190	76900	190	76900
3	680	81000	680	81000
4	56	83300	56	83300
4	256	83300	256	83300
4	1338	83300	1338	83300

Table 4. Notation summary and values used here.

Notation	Parameter	Value
n	Number of stations	10
D_B	Station bit delay	2 bits
B_L	Bit signalling rate	16 Mb/s
Lat_Buff	Latency buffer	39 bits
$Media_L$	Media length	Var. (m)
$Media_{ps}$	Propagation speed	1.5e8 m/s

81000 μs is grouped with connections with a period of 76900 μs as shown in the priority assignment of Table 3. This causes connections with period 76900 $\mu secs$ to be blocked for a duration equal to the bandwidth demands of transmitting the message from the connection with period 81000 μs in addition to the blocking contribution of packet non-preemptability and B_{gs}.

3.5.2. *Maximum Packet Size Selection*

The selection of maximum packet size ($P_{\max}$) is an important design decision. A small $P_{\max}$ results in relatively large overhead due to packet encapsulation bits. On the other hand, a large $P_{\max}$ results in increased blocking, since connections are preemptable only at packet boundaries. We can use the scheduling model to calculate $S_{\max}$ for a particular connection set at various maximum packet sizes and walk-times.

Figures 6 and 7 show $S_{\max}$ versus $P_{\max}$ for various values of walk-time for both CTR and ETR for the Connection Set-1. Observe that for a given protocol and walk-time, $S_{\max}$ increases if $P_{\max}$ becomes very small or very large. At small values of $P_{\max}$ many more packets are required to send the connection and there is increased packet encapsulation overhead. At large values of $P_{\max}$, there is an increased amount of blocking. From Figure 6, a packet size of approximately $P_{\max} = 125$ minimizes $S_{\max}$ for CTR. Similarly from Figure 7, a packet size of approximately $P_{\max} = 75$ minimizes $S_{\max}$ for ETR.

3.5.3. *CTR vs. ETR Schedulability Comparisons*

In this section we compare CTR and ETR scheduling by using our models to analyze schedulability and calculate $S_{\max}$ for the connection sets. The graphs in Figures 8 and 9 plot $S_{\max}$ as a function of walk-time W_T for different $P_{\max}$ values for both CTR and ETR. Note that the scheduling models are piecewise linear with respect to W_T, and the graphs are straight lines whose slopes change in different regions of the graph, either because a different equation for $C_j + Overhead_j$ or $Blocking_i$ is used in that region, or that the limiting connection changes. If the new limiting connection has a longer period, the slope of the

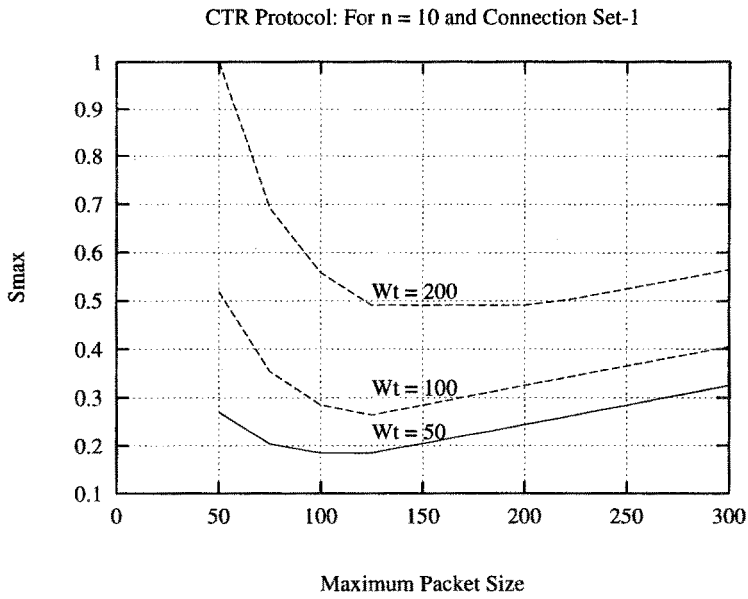


Figure 6. CTR protocol: optimum maximum packet size for various W_T values.

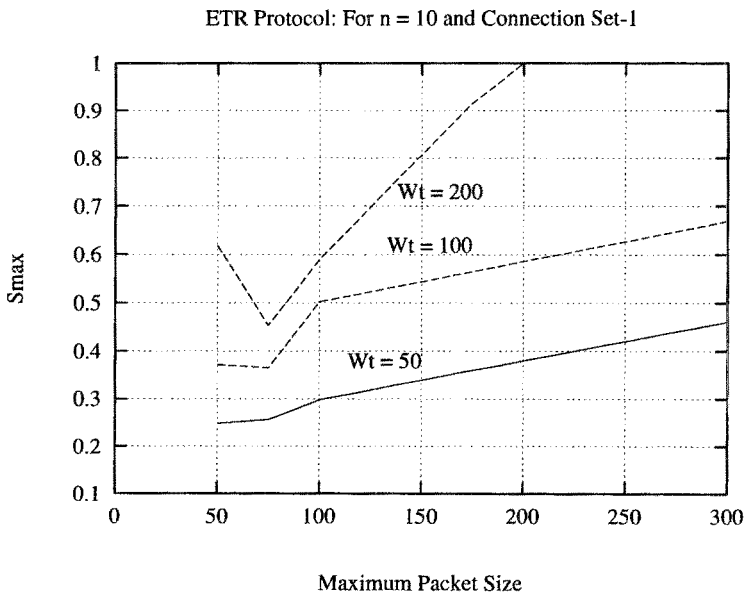


Figure 7. ETR protocol: optimum maximum packet size for various W_T values.

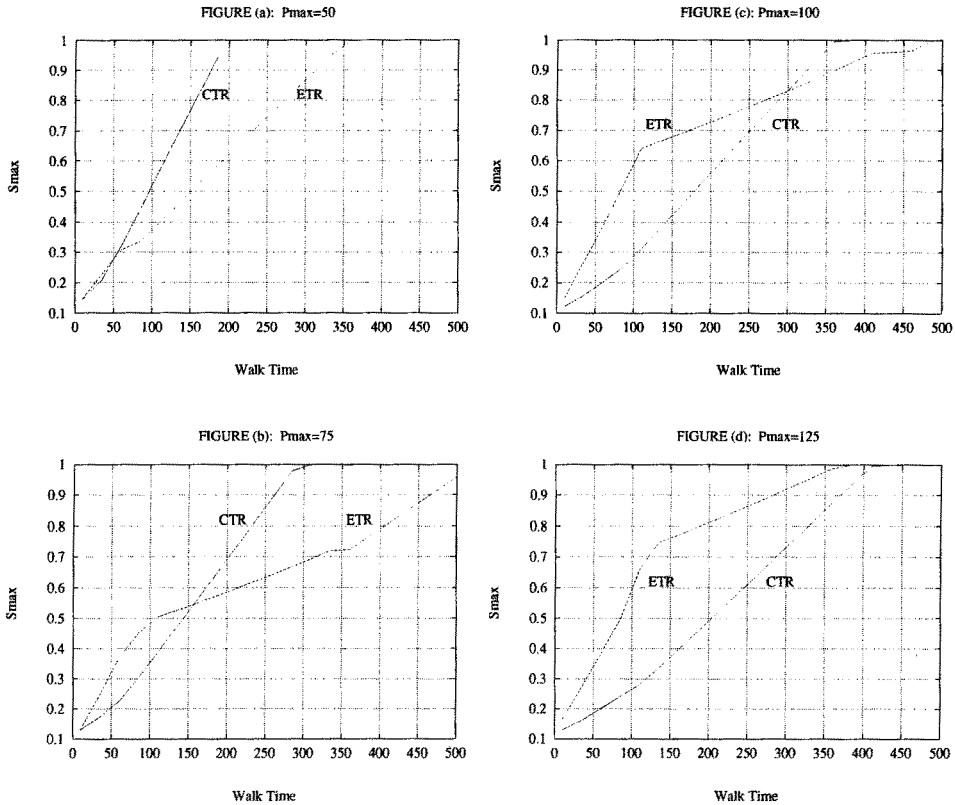


Figure 8. Connection set-1: $S_{\max}$ vs. walk-time for various $P_{\max}$ values, CTR and ETR protocol.

line decreases when the limiting connection changes, and vice versa. The slope of the lines indicates the rate at which schedulability saturation changes.

The performance of the protocols depends both on network parameters such as W_T and $P_{\max}$, and on the characteristics of the connection set. ETR has smaller overhead than CTR, particularly for large networks, since a station in ETR mode can release a free token as soon as transmission ends. ETR has a larger blocking component than CTR. If there are a large number of stations with relatively long deadlines and a few stations with short deadlines, ETR's large blocking tends to cause the short deadline connections to either miss deadlines or result in greater saturation. We expect ETR (which has greater blocking) to exhibit better real-time behavior than CTR when periods are similar, compared to its performance when periods are widely dissimilar.

First consider connection set-1. Figures 8 (a), (b), (c) and (d) show CTR and ETR performance for a $P_{\max}$ of 50, 75, 100 and 125 respectively. For each of these $P_{\max}$ values we plot schedulable saturation $S_{\max}$ as a function of walk-time W_T.

Consider the CTR curve in Figure 8(a) for $P_{\max} = 50$. The curve has a knee at approximately $W_T = P_{\max} - C_{SA} = 40$, where the slope increases. This is because as walk-time

increases beyond the packet size, stations have to hold on to the token, increasing the overhead. In the region $W_T > P_{\max} - C_{SA}$, the expression for $C_j + Overhead_j$ changes, and the $Overhead_j$ component of the scheduling model increases rapidly with W_T as indicated in Equation 9. Similarly the CTR curves in Figures 8(b), (c) and (d) have a knee at $W_T = P_{\max} - C_{SA}$ beyond which the slope increases.

Consider the ETR curve in Figure 8(b) for $P_{\max} = 75$. For small values of walk-time the efficiency of the ETR protocol offers no advantage over CTR, but the increased blocking results in a larger $S_{\max}$. In this region ($W_T < P_{\max}$) the blocking component for ETR increases rapidly with the walk-time and hence the slope of the curve is large. In the region $W_T \geq P_{\max}$ the slope decreases since $S_{\max}$ does not increase rapidly with W_T as indicated in Equation 13.

Similarly consider the ETR curves in Figures 8(c) and (d) The curves have a knee at $W_T = P_{\max}$, beyond which the slope decreases as explained in the preceding paragraph. The effect is not as apparent in Figure 8(a).

The following observations can be made about the behavior of the CTR and ETR protocols with connection set-1.

- In the region where $W_T \leq P_{\max}$, CTR and ETR lines appear to be diverging. This is because the CTR line has a relatively small slope due to small overhead, while the ETR line has a relatively large slope since its blocking increases with W_T in this region.
- In the region $W_T > P_{\max} - C_{SA}$, the CTR line has a larger slope. In this region the overhead of the scheduling model increases with W_T as indicated in Equation 9. We refer to this point as the CTR knee.
- In the region $W_T \geq P_{\max}$ the ETR line has a smaller slope since in this region the blocking component of the ETR model is more sensitive to $P_{\max}$ (which is held constant for each of the curves), and only increases slowly with W_T as indicated by Equation 13. We refer to this point as the ETR knee.

As walk-time increases beyond $P_{\max}$, there is a *crossover point* beyond which ETR performs better than CTR. Larger $P_{\max}$ values move both the CTR knee and the ETR knee to the right on the graph. The crossover occurs at increasing W_T values as $P_{\max}$ increases. CTR is so heavily penalized by the increased packet encapsulation overhead that the crossover point occurs at a very small value of W_T and ETR almost always performs better. The crossover point is at $W_T = 145$ in Figure 8(b) ($P_{\max} = 75$), and at $W_T > 200$ in Figure 8(c) ($P_{\max} = 100$) and 8(d) ($P_{\max} = 125$)

We now demonstrate the use of these graphs to select between a protocol for a token ring network that carries connection set-1. Let the size of the network be 15 kilometers yielding a walk time of $W_T = 100$ from Equation 7. For this walk-time Figure 6 gives an optimum packet size of $P_{\max} = 125$ for the CTR protocol. Similarly Figure 7 gives an optimum packet size of $P_{\max} = 75$ for the ETR protocol. From Figure 8 (b) and (d), we observe that the CTR protocol, results in a smaller value of $S_{\max}$ in both cases. CTR is the suitable protocol for this particular application. This can also be intuitively seen as follows. For this application, the maximum packet size is of the order of the walk time. ETR does not have an advantage in terms of reduced overhead, while increased blocking causes it to perform poorly compared with CTR.

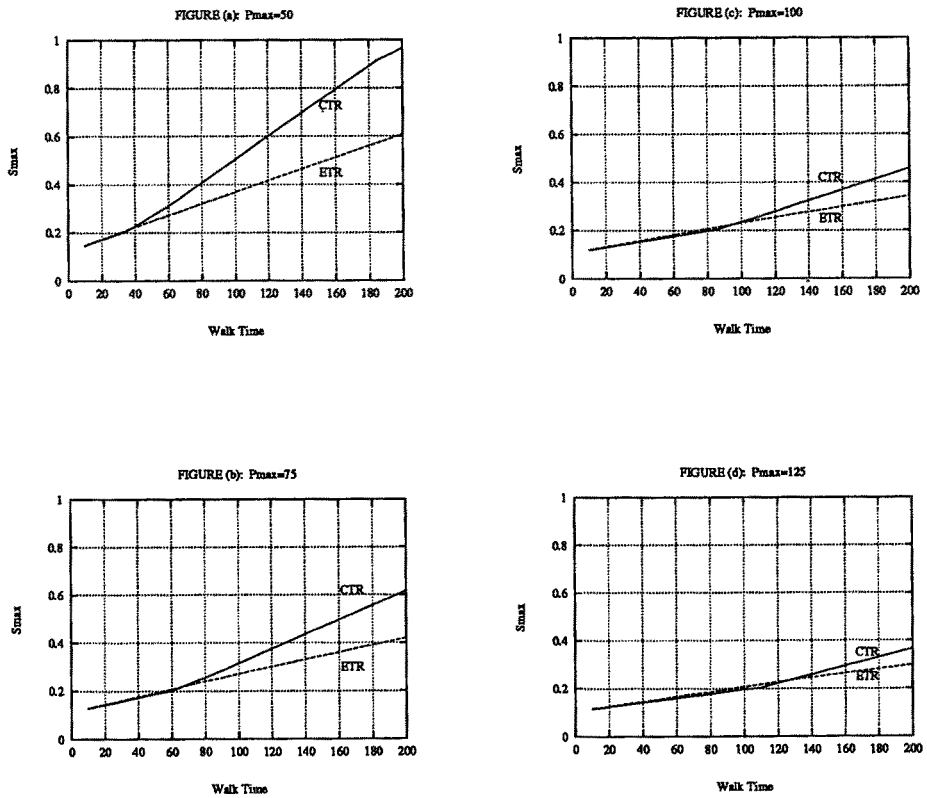


Figure 9. Connection set-2: $S_{\max}$ vs. walk-time for various $P_{\max}$ values, CTR and ETR protocol.

We now consider connection set-2 to demonstrate a case when ETR almost always performs no worse than CTR. Figures 9 (a), (b), (c) and (d) show CTR and ETR performance for a $P_{\max}$ of 50, 75, 100 and 125 respectively. As before, for each $P_{\max}$ value we plot schedulable saturation $S_{\max}$ as a function of walk-time W_T.

The periods of the connections in connection set-2 are very close to each other. Consider the CTR curves for different values of $P_{\max}$ in Figure 9. As with connection set-1, after the point $W_T + C_{SA} = P_{\max}$ the CTR line has a larger slope. Since this connection set is more sensitive to increasing overhead than to blocking, both CTR and ETR lines have a smaller slope, since larger values of $P_{\max}$ result in smaller overhead. Finally, since there is no high frequency limiting connection, the impact of blocking is reduced and ETR almost always performs better than CTR.

3.6. Conclusion

In previous sections we developed scheduling models for both the conventional, and early token release modes of the IEEE 802.5 token ring protocol based on a consistent scheduling

framework. These scheduling models can be used to reason about the timing correctness of an arbitrary set of periodic connections. We demonstrated the utility of the scheduling models in selecting network parameters such as maximum packet size. Since the models were developed using the same generic framework, we were able to compare the real-time schedulability behavior of both protocols. We found that under certain conditions the ETR protocol outperforms the CTR protocol in terms of real-time scheduling performance even though it is designed for high throughput rather than priority-based arbitration.

Acknowledgments

This work is part of the first author's doctoral dissertation at Carnegie Mellon University, which was supported by Digital Equipment Corporation.

The second author's research is supported in part by grants from the Office of Naval Research and the Naval Ocean Systems Center under contract N00014-91-J-1304.

Notes

1. Note that this assumes that the token is not captured by another high-priority station. That effect is accounted independently in the scheduling model.
2. The destination on the ring is not necessarily the destination of the call that created the connection
3. The natural-priority level is the priority that would have been assigned to the task on a system with sufficient priority levels.

References

G. Agrawal, B. Chen, W. Zhao, and S. Davari. Guaranteeing synchronous message deadlines in high speed token ring networks with timed token protocol. In *Proceedings of IEEE International Conference on Distributed Computing Systems*, 1992.

K. Arvind, K. Ramamritham, and J. Stankovic. A local area network architecture for communication in distributed real-time systems. *Journal of Real-Time Systems*, 3:115–147, 1991.

IEEE. *IEEE Std 802.5, Token Ring Access Method and Physical Layer Specification, 1992-06-12*, 1.0 edition, 1992.

H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, C-36(8):933–940, August 1987.

J. Kurose, M. Schwartz, and Y. Yemini. Multiple-access protocols and time-constrained communication. *Computing Surveys*, 16(1):43–70, March 1984.

C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 30(1):46–61, January 1973.

J. Lehoczky and L. Sha. Performance of real-time bus scheduling algorithms. *ACM Performance Evaluation Review, Special Issue*, 14(1), May 1986.

Y. Lee and L. Shen. Real-time communication in multiple token ring networks. *11th IEEE Real-Time Systems Symposium*, pages 146–153, December 1990.

J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. *10th IEEE Real Time Systems symposium*, 1989.

C. Lim, L. Yao, and W. Zhao. A comparative study of three token ring protocols for real-time communications. *11th IEEE International Conference on Distributed Computing Systems*, pages 308–317, May 1991.

P. Pleinevaux. An improved hard real-time scheduling for the IEEE 802.5. *Journal of Real-Time Systems*, 4(2):99–112, June 1992.

K. Ramamritham. Channel characteristics in local area hard real-time systems. In *Computer Networks and ISDN Systems*, pages 3–13, 1987.

S. Sathaye. *Scheduling Real-Time Traffic in Packet-Switched Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, June 1993.

S. Sathaye, D. Katcher, and J. Strosnider. Fixed priority scheduling with limited priority levels. Technical Report CMU-CDS-92-7, Carnegie Mellon University, August 1992.

J. Strosnider and T. Marchok. Responsive, deterministic IEEE 802.5 token ring scheduling. *Journal of Real-Time Systems*, 1:133–158, 1989.

L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

L. Sha, R. Rajkumar, and J. Lehoczky. Real-time computing using Futurebus+. *IEEE Micro*, June 1991.

L. Sha and S. Sathaye. Distributed real-time system design using generalized rate monotonic theory. *Second International Conference on Automation, Robotics and Computer Vision*, September 1992.

L. Sha, S. Sathaye, and J.K. Strosnider. Scheduling real-time communication on dual link networks. *13th IEEE Real-Time Systems Symposium*, pages 188–197, December 1992.

J. Strosnider. *Highly Responsive Real-time Token Rings*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, August 1988.