Deferring Real-Time Traffic for Improved Non-Real-Time Communication in FDDI Networks

Moncef Hamdaoui, *Member, IEEE Computer Society*, and Parameswaran Ramanathan, *Member, IEEE*

Abstract—**The fiber distributed data interface (FDDI) is suitable for real-time communication because of the high speed and the performance guarantees it provides. However, these guarantees are achieved at the expense of non-real-time traffic because the real-time messages are given higher priority over non-real-time messages, resulting in excessive delays for non-real-time messages. In this paper, we propose a scheme for reducing the response time of non-real-time messages while still providing the same guarantees to real-time messages. In particular, the proposed approach gives higher priority to real-time messages only when it is absolutely necessary in order for them to meet their deadlines. Non-real-time messages are thus transmitted ahead of real-time messages whenever possible. We present an algorithm for determining when and by how much the transmission of a real-time message can be deferred without jeopardizing its deadline. The proposed approach is evaluated through simulation. The simulation results show that a substantial reduction in the mean response time of non-real-time messages is achieved when using the proposed approach.**

Index Terms—**Real-time communication, non-real-time communication, timed token protocol, FDDI.**

I. Introduction

COMPUTER networks which support the timed token medium access control (MAC) protocol have become increasingly prevalent. In this protocol, the stations in the system form a logical ring. The protocol provides support for two types of service: *synchronous* and *asynchronous*. Each station is allocated a portion of the network bandwidth for its synchronous traffic. When a station receives the token, it can transmit messages from its synchronous load for at least its pre-allocated time, called its *synchronous capacity,* before releasing the token to its downstream neighbor. Messages from its asynchronous load are transmitted only if time permits. (The protocol is formally described in Section II.) The fiber distributed data interface (FDDI) network is a 100 Mbits/sec token ring that supports this protocol [1], [13], [18], [19].

The main advantage of this protocol is that each station is guaranteed a certain average bandwidth as well as a bounded access time to the transmission medium. These two properties are very important for applications like digitized audio/video communication and distributed real-time control systems. In these applications, messages belong to one of two categories: *real-time* or *non-real-time*. The real-time messages are often generated at regular intervals. Each real-time message has a *deadline* by which it must reach its destination. For example, in a full motion video application, 30 frames are generated every second, and each frame must be delivered to the destination within a bounded delay to avoid any observable jitter. The deadline is usually equal to the period to ensure that a frame is delivered before the next frame is generated. In contrast, the non-real-time messages are generated sporadically and are not time-constrained. The objective in servicing non-real-time messages is to deliver them as soon as possible without jeopardizing the deadlines of real-time messages. In this paper, we propose an approach for transmitting messages in the timed token protocol that improves the response time of non-real-time messages while still guaranteeing the deadlines of real-time messages. Through simulation we demonstrate that the average response time of non-real-time messages is reduced considerably by the proposed approach.

Prior work on the timed token protocol can be classified into one of two categories. One category deals with guaranteeing the deadlines of real-time messages. The focus is on selecting the protocol parameters in such a way that each real-time message is guaranteed to meet its deadline. For instance, Sevcik and Johnson first showed that the worst-case token rotation time is twice the protocol parameter, *target token rotation time* (TTRT) [22], and, therefore, TTRT cannot be greater than one-half the minimum deadline among all time-constrained messages. In [16], Montushi et al. derived minimum synchronous capacity requirements to ensure that the bandwidth guaranteed to each station for synchronous messages is no lower than the traffic generated for that service class. Agrawal et al. proposed and analyzed several schemes for selecting the synchronous capacities to guarantee the deadlines of real-time messages [2], [3], [4]. Hamdaoui and Ramanathan extended this work to include the selection of both TTRT and synchronous capacities [8], [9]. In [15], Malcolm and Zhao accounted for arbitrary deadlines in selecting the parameters. Given a set of real-time streams, the above schemes can be used to select suitable protocol parameters to guarantee the deadline constraint of each real-time message.

The second category of prior work deals with the transmission of non-real-time messages. Jain showed that each station gets an opportunity to transmit asynchronous frames in a bounded time, although the bound is fairly large [10]. Pang and Tobagi analyzed the performance of token passing networks under heavy loads [17]. The emphasis of most other work is on selecting suitable protocol parameters to improve the response time of non-real-time messages [5], [11], [14], [21], [24]. For instance,

Manuscript received Nov. 29, 1993; revised May 15, 1994.

A preliminary version of this paper was presented at the Conference on Local Computer Networks, 1993.

M. Hamadoui is with Northern Telecom in Tunis, Tunisia.

P. Ramanathan is with the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706-1691; e-mail: parmesh@ece.wisc.edu.

To order reprints of this article, e-mail: transactions@computer.org, and reference IEEECS Log Number C95073.

Sankar and Yang studied, through simulation, the influence of TTRT on the average frame delay [21]. The effect of protocol parameter settings on the throughput of different classes of asynchronous traffic is studied in [5] and [11].

This paper proposes a new strategy which can be used in conjunction with any of the above schemes to further improve the response time of non-real-time messages. The response time is improved by transmitting non-real-time messages ahead of real-time messages whenever possible. The proposed approach makes use of the fact that the real-time messages only need to meet their deadlines; the value of a real-time message does not depend on the exact delivery time as long as it is delivered before the deadline.

In this paper, we derive expressions that each station can use to determine if the transmission of a real-time message can be deferred to a later time. Such a deferment is possible if the station is guaranteed to receive the token enough times in the future to ensure transmission of its real-time messages before their respective deadlines. If the transmission of real-time messages can be deferred, the station transmits its non-real-time messages and releases the token (without transmitting any real-time messages). Since the real-time messages are transmitted just prior to their deadline, there is also a reduction in the jitter experienced by them. This benefit is especially important in applications such as real-time video conferencing.

The rest of the paper is organized as follows. A brief description of the timed token protocol is given Section II. The system characteristics are formally described in Section III. In Section IV, the relevant properties of the timed token protocol are presented and the proposed approach is described. Simulation results are presented in Section V. The paper concludes with Section VI.

II. Timed Token MAC Protocol

For completeness, a brief overview of the timed token medium access control (MAC) protocol is presented in this section. A more detailed description of the timed token protocol and/or of FDDI can be found in [1], [7], [13], [18], [19].

The stations in the network are connected to form a logical ring. A special bit pattern, called the *token*, rotates around the ring to signify the right to transmit. The timed token protocol regulates possession of the token and thus access to the transmission medium. The protocol supports two types of service, synchronous and asynchronous. Synchronous traffic is assigned a guaranteed bandwidth and is usually used for periodic, time-critical messages that require a predictable response time. The leftover bandwidth (unallocated, unused or both) is dynamically shared among all the stations for asynchronous traffic. Asynchronous service is used for messages where the response time is less critical.

During network initialization, the stations negotiate a protocol parameter called the *Target Token Rotation Time* (TTRT), which is the *expected* time it takes the token to make one rotation around the ring. The negotiated value of TTRT is often referred to as the operational value (T_Opr). Each station is assigned a portion of TTRT, called *synchronous capacity*, to transmit its synchronous messages. The synchronous capacity allocated to station i is denoted by H_i. Each station has the following timers/counters:

1) *Token Rotation Timer* (TRT): TRT always counts down and is always enabled. It is reset to T_Opr every time it expires.
2) *Late Count* (LCT): LCT records the number of times TRT has expired since the token was last received by the station. If LCT is zero when the token is received at the station, the token is said to be *early*. Otherwise, the token is said to be *late*.
3) *Token Holding Timer* (THT): THT also counts down. It is enabled only during the transmission of asynchronous frames.

When a station receives the token, it does one of the following (depending on LCT):

- If the token arrives early (i.e., LCT = 0), then the current value of TRT is placed in THT and TRT is reset to T_Opr. The station transmits its synchronous frames for a time not to exceed its allocated synchronous capacity. Asynchronous frames may then be transmitted until THT or TRT expire.
- If the token arrives late (i.e., LCT > 0), then LCT is reset to 0 and TRT continues counting down. The station transmits its synchronous frames for a time not to exceed its allocated synchronous capacity. Note that in this case, TRT is not reset to T_Opr and no asynchronous frames are transmitted.

The synchronous capacities allocated to the stations must be such that the sum is no greater than the usable portion of TTRT. That is,

$$\sum_{i=1}^{n} H_i \le \text{TTRT} - \tau, \tag{2.1}$$

where τ is the ring overhead which includes the physical ring latency (the time needed for the token to propagate once around the ring when not disturbed) and other protocol-related overheads.

III. System Model

Consider an FDDI network with n stations numbered $1, 2, \ldots, n$. A mixture of real-time (time-critical) and non-real-time (non-time-critical) messages is generated at each station. Real-time messages have deadlines before which they must be transmitted. These messages are generated periodically and are referred to in this paper as *periodic real-time streams*. Examples of such streams include voice or video transmissions and periodic sensor readings in real-time control applications. A periodic real-time message stream S is characterized by a triple (P, C, D) where

- P – Message interarrival time.
- C – Maximum time needed to transmit a message from the stream.
- D – Relative deadline, i.e., a message arriving at time t

must be fully transmitted before $t + D$. We assume that $D \leq P$ so that a message is always transmitted before the next message from the stream is generated.

For example, consider the transmission of compressed real-time video, where each frame has to be transmitted before the generation of the next. If 25 frames are to be transmitted per second and if the maximum frame size is 150 Kbits, then each frame has a deadline of 40 ms and a maximum transmission time of 1.5 ms on a 100 Mbps network. Such stream is therefore denoted by (40ms, 1.5ms, 40ms).

Let p_i be the number of real-time streams at station i and let the jth stream be denoted by $S_{ij} = (P_{ij}, C_{ij}, D_{ij})$. Let h_{ij} denote the synchronous capacity assigned to stream S_{ij}. The overall synchronous capacity assigned to station i is $H_i = \sum_{j=1}^{p_i} h_{ij}$. We assume that the synchronous capacities are selected in such a way that the deadline of every real-time message in the system is guaranteed. Algorithms for this purpose can be found in [2], [3], [4], [8], [9], [15]. The basic idea of these algorithms is to assign capacities such that the transmission time available to a station between the arrival of a message and its deadline is sufficient to fully transmit the message. Messages may have to be fragmented and transmitted in multiple token visits in order to ensure that all the deadlines are met.

At each station, the earliest-deadline-first policy is used to service the real-time messages. In this policy, when a station is ready to transmit a real-time message, it picks the one with the closest deadline and transmits it (or part of it). However, if while a message is being transmitted, a message with an earlier deadline is generated (i.e., becomes ready to transmit), the ongoing transmission is continued. The nonpreemptive earliest-deadline-first policy coupled with suitable parameter values guarantees timely delivery of all real-time messages.

Non-real-time messages, on the other hand, are generated sporadically and are not time-constrained. They do not have deadlines; however, they must be transmitted as soon as possible without jeopardizing the timely delivery of real-time messages.

In the following section we present a scheme to reduce the average delivery time of non-real-time messages while guaranteeing the deadline of each real-time message.

IV. Proposed Approach

The timed token protocol supports two types of service, synchronous and asynchronous. Each station is guaranteed a certain average bandwidth for its synchronous traffic. This bandwidth is usually used to service real-time messages. The leftover bandwidth is shared among all the stations and is used to service non-real-time messages (asynchronous traffic). Because real-time messages are time critical, the timed token protocol gives higher priority to real-time messages over non-real-time messages. When a station receives the token, real-time messages are always transmitted. The non-real-time messages are transmitted only if timing permits. Consequently, non-real-time messages may encounter large delays. To alleviate this problem, the proposed approach gives higher priority to non-real-time messages whenever possible. The idea is to transmit non-real-time messages ahead of real-time messages as long as the real-time messages can still meet their deadlines. In fact, the real-time messages are transmitted only when it is absolutely necessary to do so in order to ensure their timely delivery. A station may also release the token without transmitting its real-time messages if it can still guarantee the timely transmission of its real-time messages, thus allowing other stations to transmit their non-real-time messages.

The example in Fig. 1 illustrates the proposed approach. Suppose that a station receives the token at time t_0. Also suppose that, at time t_0, the station has a non-real-time message M_N and a real-time message M_R with a deadline of $t_0 + 4 \cdot \text{TTRT}$. Furthermore, suppose that the synchronous capacity of the station and the remaining length of M_R and M_N are such that it will take two token visits to fully transmit M_R and one token visit to fully transmit M_N. Fig. 1a shows the order of transmission of these two messages in the traditional timed token protocol. The real-time message is transmitted during the first two token visits and the non-real-time message is transmitted during the third visit. The proposed approach is depicted in Fig. 1b. Because the remaining time to the deadline is $4 \cdot \text{TTRT}$, the station is guaranteed to receive the token at least three times before the deadline (cf. Section IV.A.) Since the station requires only two token visits to transmit M_R, the proposed approach will transmit M_N in the current token visit and M_R in the following two token visits. Note that, M_R is still guaranteed to meet its deadline and the response time of M_N is substantially reduced.

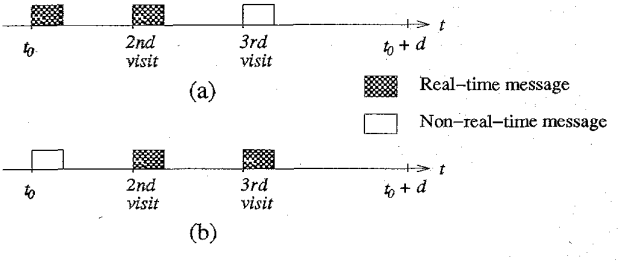


Fig. 1. Order of transmission of a real-time message and a non-real-time message using (a) the conventional approach and (b) the proposed approach.

It might seem at this point that the benefit from the proposed approach is due to the fact that a station has more capacity than necessary to satisfy the deadline of its real-time messages. A closer look reveals two main reasons for the reduction in the response times of non-real-time messages. First, to provide worst-case guarantee, parameter selection schemes assume that the token visits a station as few times as possible between the generation time of a real-time message and its deadline. However, the actual number of token visits is often more than what is assumed. This is because stations do not always have non-real-time messages to fully utilize the bandwidth unused by the real-time traffic. Second, since the deadlines of the real-time messages have to be guaranteed in the *worst-case,* parameter selection schemes assume that all real-time messages are of maximal length. However, the actual lengths are often smaller due to compression or inherent variation in the amount of data to be transmitted, e.g., the ratio be-

tween the maximum and the average frame sizes can be as large as three, if the video frames are compressed using the MPEG standard [6].

For the above two reasons, stations will often be able to defer the transmission of their real-time messages to later token visits and reduce the response time of non-real-time messages. Stations can further improve the response time of the non-real-time messages by reordering the transmission of real-time and non-real-time messages during each visit. In other words, even if a station has to transmit some real-time frames, it may be able to transmit some of its non-real-time frames first and then transmit the real-time frames.

In Section IV.B, we present an algorithm to determine when and by how much the transmission of a real-time message can be deferred. We begin by presenting some relevant timing properties of the timed token protocol.

A. Timing Properties

A key property of the timed token protocol is that it provides each station a bounded access time of 2 · TTRT to the transmission medium. This is because the maximum token rotation time has been shown to be 2 · TTRT [12], [22]. Agrawal et al. generalized this result to provide an upper bound on the time between any v consecutive visits of the token to a given station [2], [3]. Let $t_i(l)$ denote the time when the token makes its lth visit to station i.

THEOREM 1. [From [2], [3]] *For any $l > 0$, $v > 0$, and any station i ($1 \le i \le n$)*

$$t_i(l+v-1) - t_i(l) \le v \cdot \text{TTRT} - H_i. \tag{4.1}$$

Suppose that at time t_0, station i has a real-time message with a relative deadline d and a transmission time C queued for transmission. In order to guarantee that the message can be fully transmitted before its deadline, we need to know the total transmission time guaranteed to station i in the interval $[t_0, t_0 + d]$. The next corollary is a direct result of Theorem 1. It gives a lower bound on the transmission time available to station i during any interval of duration d.

COROLLARY 1. *For any $d > 0$, the transmission time available to a station with a synchronous capacity h during any time interval of duration d is at least:*

$$X(h, d) = \begin{cases} 0 & \text{if } d \le \text{TTRT} \\ \left(\left\lfloor \frac{d}{\text{TTRT}} \right\rfloor - 1\right) \cdot h + \max\left\{0, \left(d - \left\lfloor \frac{d}{\text{TTRT}} \right\rfloor \cdot \text{TTRT}\right) - (\text{TTRT} - h)\right\} & \text{otherwise} \end{cases} \tag{4.2}$$

PROOF. Suppose at time $t = 0$, a real-time message with deadline d is generated at a station. Also suppose that the station has a synchronous capacity h (< TTRT). The time available to the station during the interval $[t, t + d]$ is minimal if it has just released the token at time $t = 0^-$ (since the station may have to wait the longest before it receives the token).

Case 1: $d \le \text{TTRT}$

Since a token rotation may take up to 2 · TTRT $- h > d$, the station may not receive the token before the deadline d. Hence, $X(h, d) = 0$.

Case 2: $d > \text{TTRT}$

Let $d = q \cdot \text{TTRT} + r$ where $q = \left\lfloor \frac{d}{\text{TTRT}} \right\rfloor$ and $0 \le r < \text{TTRT}$.

From time 0 to $(q \cdot \text{TTRT} - h)$, the station is guaranteed to receive the token at least $(q - 1)$ times (by Theorem 1). During each visit, the station can transmit for h time units, giving a total transmission time of $(q - 1) \cdot h$. Again, by Theorem 1, the latest time the qth token visit can occur is $t_q = (q + 1) \cdot \text{TTRT} - h$. If $d \ge t_q$ (i.e., the qth occurs before the deadline), then the station can transmit frames for $d - t_q$ during this last visit. On the other hand, if $d < t_q$, the deadline may expire before the qth visit and thus no transmission is possible in the qth visit. Therefore, the station can transmit frames for $\max\{0, d - t_q\}$ during the qth visit. The overall transmission time hence guaranteed to a station in any interval of length d is

$$(q-1) \cdot h + \max\{0, d - t_q\}. \tag{4.3}$$

The corollary follows by substituting $(q + 1) \cdot \text{TTRT} - h$ for t_q and $\left\lfloor \frac{d}{\text{TTRT}} \right\rfloor$ for q in the above equation. □

B. Formal Description

As described in Section II, the timed token protocol allows station i to transmit synchronous frames for H_i time units each time it receives the token. If the token is early, it can also transmit asynchronous frames for THT_i time units. The overall transmission time, however, must not exceed TTRT. Therefore, when station i receives the token, its capacity, or the duration for which it can transmit frames is given by (recall that THT_i is 0 when the token is late)

$$\text{CAP}_i = \min\{H_i + \text{THT}_i, \text{TTRT}\} \tag{4.4}$$

The station must then determine how much of the real-time messages and the non-real-time messages it should transmit. In order to reduce the response time of the non-real-time messages, the station should transmit as little of the real-time messages as possible, without jeopardizing their timely delivery.

B.1. Deferment of Real-Time Messages

Let $M_{ij} = (c_{ij}, d_{ij})$ denote the message from stream j where c_{ij} is the remaining transmission time of the message and d_{ij} is the time remaining to the deadline of the message. Note that the message may have been partially transmitted and so $c_{ij} \le C_{ij}$ and $d_{ij} \le D_{ij}$. Let $M_{ij} = (0, \infty)$ if no message from jth stream at station i is ready for transmission.

When a station receives the token, it must determine how much of the real-time messages can be deferred to later token visits and how much should be transmitted during the current visit. The next theorem provides an answer to this question.

THEOREM 2. *Suppose station i has just received the token. Then, the station will be able to transmit all its real-time messages before their respective deadlines if it transmits real-time frames for*

$$\text{RT\_CAP}_i = \max \sum_{j=1}^{p_i} \left\{0, c_{ij} - X\left(h_{ij}, d_{ij} + (\text{TTRT} - \text{TRT}_i)\right)\right\} \tag{4.5}$$

time units during the current token visit.

PROOF. Suppose that, at time t_0, station i receives the token and sets TRT_i appropriately. That is, TRT_i is set to TTRT if token is early and left unchanged otherwise. The amount of time guaranteed to station i in the future (not including the current visit) to transmit message M_{ij} is $X(h_{ij}, d_{ij} + (TTRT - TRT_i))$ as shown below.

Case 1: (Token is early). From Corollary 1, the time guaranteed to station i to transmit M_{ij} in future token visits is $X(h_{ij}, d_{ij})$. This guaranteed time does not depend on whether or not the station transmits some frames during the current visit because other stations will, in the worst-case, use the unutilized time to transmit their messages. Since the token is early, $TRT_i = TTRT$ and hence $X(h_{ij}, d_{ij})$ can be written as $X(h_{ij}, d_{ij} + (TTRT - TRT_i))$.

Case 2: (Token is late). Let the current visit be the lth visit to station i, i.e., $t_0 = t_i(l)$. Let $t_e = t_i(l - k)$, $k > 0$, be the last time at which station i received the token early. That is, the $(l - k)$th visit was early but every visit since then has been late. Fig. 2 shows a plot of TRT_i between t_e and t_0 for $k = 3$. TRT_i was set to TTRT at time t_e (since the token was early) and also at every time it expired. Note that, TRT_i expires exactly once between every two token visits since the token is late at every visit. Hence,

$$t_0 - t_e = k \cdot TTRT + (TTRT - TRT_i), \qquad (4.6)$$

where TRT_i denotes the value in the TRT counter at node i at time t_0. Using Corollary 1, the time guaranteed to station i at time t_e for the transmission of M_{ij} would have been $X(h_{ij}, t_0 + d_{ij} - t_e)$. Using (4.6), we have

$$\begin{aligned} X(h_{ij}, t_0 + d_{ij} - t_e) &= X(h_{ij}, d_{ij} + k \cdot TTRT + (TTRT - TRT_i)) \\ &= X(h_{ij}, d_{ij} + TTRT - TRT_i) + k \cdot h_{ij}. \end{aligned} \qquad (4.7)$$

Subtracting the time available during the last k visits (i.e., $k \cdot h$), we get the time guaranteed in the future as $X(h_{ij}, d_{ij} + TTRT - TRT_i)$.

Since only $X(h_{ij}, d_{ij} + (TTRT - TRT_i))$ time units are guaranteed in the future, $\max\{0, c_{ij} - X(h_{ij}, d_{ij} + (TTRT - TRT_i))\}$ of M_{ij} must be transmitted during the current visit to ensure the timely delivery of M_{ij}. Transmitting real-time messages for

$$RT\_CAP_i = \sum_{j=1}^{p_i} \max\{0, c_{ij} - X(h_{ij}, d_{ij} + (TTRT - TRT_i))\} \qquad (4.8)$$

will therefore ensure the station's ability to transmit all its real-time messages before their deadlines. □

Since the synchronous capacities have been selected such that all real-time messages are guaranteed, RT_CAP_i will be no more than H_i. Also, if RT_CAP_i is zero, real-time messages need not be transmitted even if they are available for transmission.

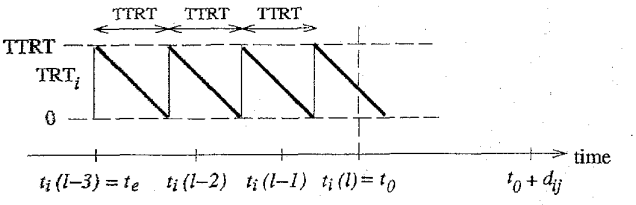


Fig. 2. A plot of TRT_i versus time.

B.2. Order of Transmission

When station i receives the token, it can transmit frames for CAP_i time units, as given by (4.4). During this time, the station must transmit real-time frames for RT_CAP_i in addition to any non-real-time frames it can transmit. What should the order of transmission be?

A straightforward solution is to transmit the real-time frames *first* and then transmit non-real-time frames until the station's capacity is exhausted. Using this approach, all deadlines are guaranteed to be met, but, some real-time frames may be transmitted earlier than necessary. In general, it may be possible to transmit some non-real-time frames before transmitting the real-time frames. For example, suppose a station has two real-time messages with deadlines d_1 and d_2 as shown in Fig. 3. The figure shows the possible scenarios. In Fig. 3a, d_1 and d_2 are greater than CAP and the transmission of the real-time frames can be deferred until the end. That is, the station can transmit non-real-time frames for up to CAP − RT_CAP then transmit the real-time frames for the remaining duration. In Fig. 3b, d_1 is less than CAP but d_2 is greater. Therefore, the transmission of the second message can be deferred until the end but the first message must be transmitted earlier. In Fig. 3c, both d_1 and d_2 are less than CAP. Consequently, both messages must be transmitted before some of the non-real-time frames. The solution depicted in Fig. 3 is optimal in the sense that it minimizes the response time of the non-real-time messages.

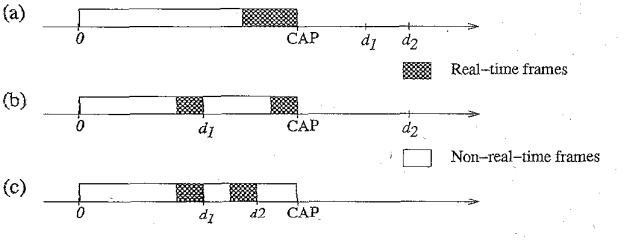


Fig. 3. Optimal order of transmission of real-time and non-real-time frames during a token visit.

The implementation of the optimal solution is, however, impractical because a schedule for transmitting the real-time frames has to be constructed every time the station receives the token. Therefore, we adopt a simpler solution in which all RT_CAP time units are transmitted at once. The station transmits few non-real-time frames (if any), then transmits real-time frames for RT_CAP time units and then transmits more non-real-time frames (if any) for as long as the protocol allows. The start time of the real-time transmission (or

equivalently the duration of early non-real-time transmission) is based on the earliest deadline of all the real-time messages awaiting transmission. This duration, called non-real-time capacity, is computed as

$$\text{NRT\_CAP}_i = \max\left\{0, \min\left\{d_i^{min}, \text{CAP}_i\right\} - \text{RT\_CAP}_i\right\} \quad (4.9)$$

where $d_i^{min} = \min\{d_{ij} : j = 1, 2, \ldots, p_i\}$. The idea is to transmit all real-time frames before the earliest deadline. Note that, $\min\{d_i^{min}, \text{CAP}_i\} - \text{RT\_CAP}_i\} < 0$ implies that not all RT_CAP_i time units can be transmitted before the earliest deadline. However, this is not a problem; the deadlines are still met as long as the messages are transmitted in the earliest-deadline-first order because some of the frames belong to a message with a later deadline.

The flowchart in Fig. 4 formally describes the proposed approach. When a station receives the token, it computes RT_CAP and NRT_CAP as given by (4.5) and (4.9).[1] Both RT_CAP and NRT_CAP are down counters. RT_CAP is enabled (counts down) only when a real-time frame is being transmitted. Similarly, NRT_CAP is enabled only when a non-real-time frame is being transmitted. The station transmits non-real-time frames (if any) until NRT_CAP reaches zero. Real-time frames are then transmitted until RT_CAP reaches zero. The station then continues transmitting non-real-time frames until either its overall capacity is exhausted, or there are no more non-real-time frames to transmit.

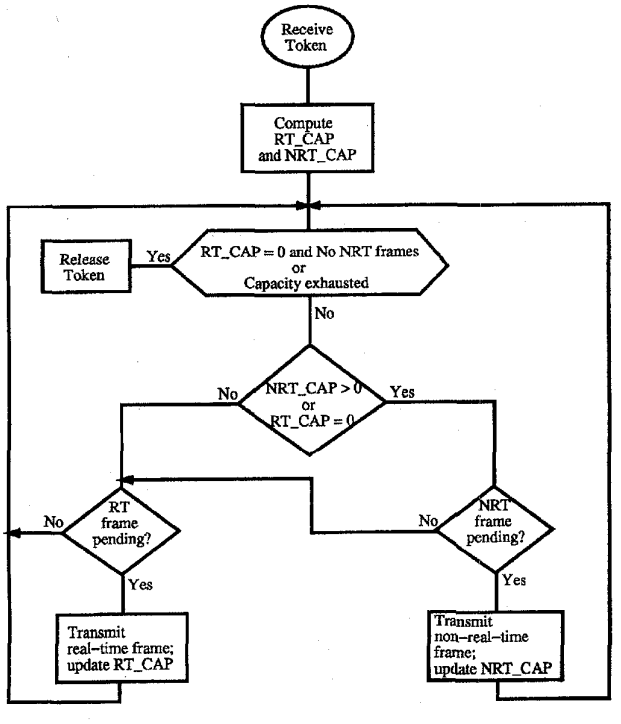


Fig. 4. Proposed approach.

V. Performance Evaluation

The performance of the proposed approach was evaluated through simulation. The following parameters are specified to the simulator:

- the number of stations (n),
- the periodic real-time streams at each station,
- the non-real-time streams at each station,
- the synchronous capacities, TTRT, and the ring latency τ.

In the results presented here, the transmission time of a real-time message is selected randomly at the time of its generation from a uniform distribution in the interval $[C_{min}, C_{max}]$, where C_{min} and C_{max} are parameters specified for each real-time stream. Two types of non-real-time message streams were considered: *aperiodic* and *bursty*. In an aperiodic stream, messages are generated according to a Poisson process. The transmission times are exponentially distributed with an average C_{ave}. In a bursty stream, messages are generated in *bursts*. Such a stream can be thought of as a process with two states, ON and OFF [20], [23]. Messages are generated during the ON state only. During the ON state, messages are generated periodically. Their transmission times are selected randomly at the time of their generation from a uniform distribution in the interval $[C_{min}, C_{max}]$, where C_{min} and C_{max} are parameters specified for each non-real-time bursty stream. The durations of the ON state and the OFF state are exponentially distributed with averages ON_{ave} and OFF_{ave}, respectively. The generation rate is varied so that the overall non-real-time traffic load is varied from about 10% to about 80% of the leftover bandwidth (overall bandwidth minus the average bandwidth used by real-time traffic). The delay of a non-real-time message is computed as time period between its generation at the source station and its arrival[2] at the destination station minus the time required to transmit the message. The averages are computed over 200,000 non-real-time messages.

The protocol parameters are selected such that all real-time messages are guaranteed to meet their deadlines [8], [9]. The capacities are allocated using the *normalized proportional allocation scheme*, where the usable portion of TTRT (i.e., TTRT $- \tau$) is divided among all the stations such that the synchronous capacity allocated to a station is proportional to the real-time load at that station [2], [3], [8], [9].

The first system simulated is described in Table I. It consists of four identical stations, each with a periodic real-time message stream. Each station also has an aperiodic stream of non-real-time messages. Since the deadlines are equal to 3 · TTRT, stations will often not be able to defer the transmission of their real-time messages to later token visits. As a result, the reductions in the average delay are mainly due to the fact that, within each token visit, a station transmits some of its non-real-time frames ahead of its real-time frames. Fig. 5 shows a plot of the average delay incurred by non-real-time messages under the various traffic loads. At moderate loads, this amounts to a 20% to 30% reduction in the average delay.

1. RT_CAP and NRT_CAP can be periodically computed. When the token arrives, the station can use the most recently computed values.

2. The arrival time of a message is defined as the time when the last bit of the message is received at the destination.

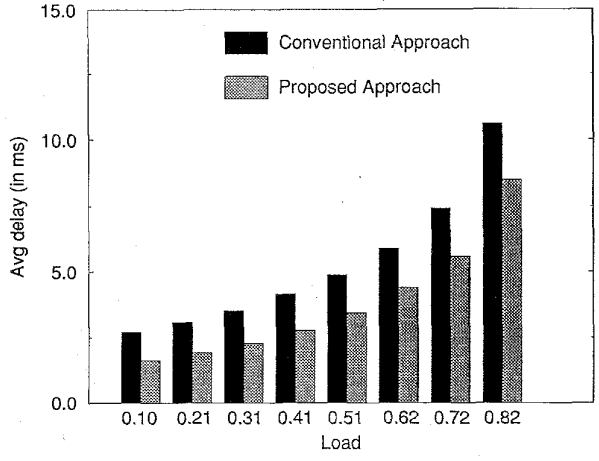


Fig. 5. Average delay of non-real-time messages in the system described in Table I.

TABLE I
REAL-TIME AND NON-REAL-TIME STREAMS IN SYSTEM 1.
TTRT = 33ms AND τ = 1ms

Station #	Real-time stream	Non-real-time stream	H
1-4	P = 100ms, D = 100ms C_{min} = 1ms, C_{max} = 10ms	C_{ave} =0.5ms	8ms

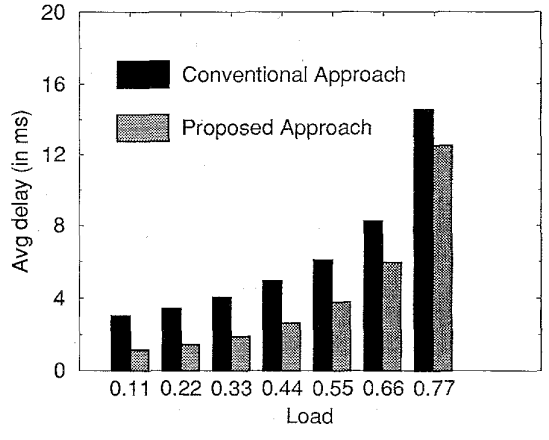


Fig. 6. Average delay of non-real-time messages for the system described in Table II.

TABLE II
REAL-TIME AND NON-REAL-TIME STREAMS IN SYSTEM 2.
TTRT = 8.325ms AND τ = 1ms

Station #	Real-time stream	Non-real-time stream	H
1-3	P = 33ms, D = 33ms C_{min} = 0.5ms, C_{max} = 2ms	C_{ave} = 0.5ms	0.916ms
4-6	P = 100ms, D = 100ms C_{min} = 1ms, C_{max} = 10ms	C_{ave} = 0.5ms	1.525ms
7-20	—	C_{ave} = 0.5ms	0

In the above system, all stations have similar real-time and non-real-time loads. The second system simulated consists of a network of 20 stations, six of which have real-time streams (see Table II). Furthermore, the real-time streams are not all the same. The maximum real-time utilization is also higher in this case (about 48%). Again, the protocol parameters are selected to ensure timely delivery of all real-time messages. Note that, the stations with no periodic real-time streams are not assigned any synchronous capacity. All 20 stations have aperiodic non-real-time message streams. A plot of the average delay of non-real-time messages is shown in Fig. 6. At moderate loads, the reduction is over 50%. Note that, the reductions in this case are greater than those shown in Fig. 5. This is because more deferments of real-time frames are possible for the system in Table II as compared to that in Table I. The reason for more deferments is as follows. In the Table I system, real-time frames can never be deferred to the next token visit because:

1) a station is guaranteed only two visits in each period of its stream and
2) the periodic stream requires two token visits for complete transmission.

Therefore, the reduction observed in Fig. 5 is only due to reordering of transmission in each token visit. In contrast, in the Table II system, the reduction in response time is due to deferment of real-time frames to next token visit and reordering of frames in each token visit.

Recall that, a deferment of real-time frames to the next token visit is possible due to two reasons. First, messages are usually shorter than the maximum length assumed by the parameter selection schemes. Second, the token rotates faster than expected because the stations do not always have non-real-time messages for transmission. For instance, in Fig. 7, observe the difference between the expected token rotation time (TTRT) and the actual average rotation time for the system described in Table II. To study the effect of faster token rotations, we simulated a system with no variation in the sizes of real-time messages within a given periodic stream. The exact parameters are shown in Table III. The parameters are such that the average real-time load is the same as in the previous system (described in Table II). The non-real-time load is also the same as in the previous system. Fig. 8 shows the average delay of non-real-time messages versus the non-real-time load. Even though the reductions are smaller than before, they are still substantial. It shows that the reduction in average delay is substantial even if there is no variation in the size of real-time messages.

TABLE III
REAL-TIME AND NON-REAL-TIME STREAMS IN SYSTEM 3.
TTRT = 16.650ms AND τ = 1ms

Station #	Real-time stream	Non-real-time stream	H
1-3	P = 33ms, D = 33ms C_{min} = C_{max} = 1.25ms	C_{ave} = 0.5ms	2.237ms
4-6	P = 100ms, D = 100ms C_{min} = C_{max} = 5ms	C_{ave} = 0.5ms	2.980ms
7-20	—	C_{ave} = 0.5ms	0

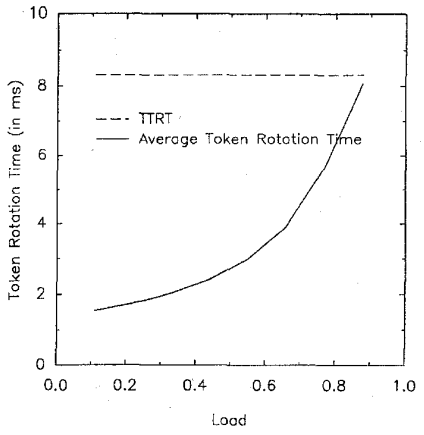


Fig. 7. Average token rotation time in the system described in Table II.

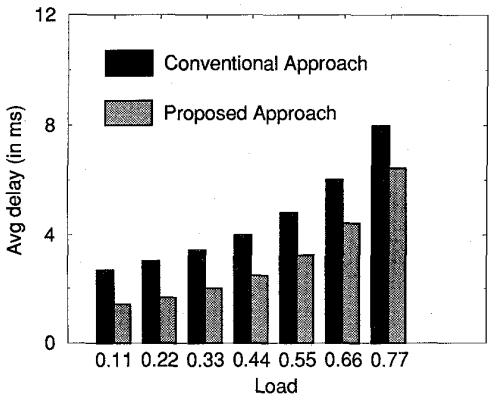


Fig. 8. Average delay of non-real-time messages for the system described in Table III.

In all the three systems considered so far, the stations are assumed to have similar non-real-time traffic loads. Fig. 9 shows the average delay of non-real-time messages in a system similar to that described in Table II except that the non-real-time traffic load at station 4 is ten times higher than the non-real-time traffic loads at other stations. A reduction of more than 1 ms in the average delay is achieved.

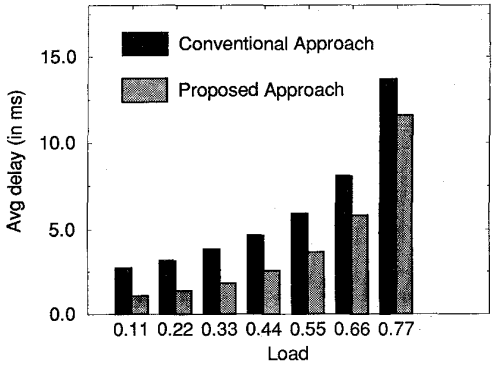


Fig. 9. Average delay of non-real-time messages for a system with one heavily loaded station.

The system described in Table IV is similar to that described in Table II except that the non-real-time message streams are now bursty. A plot of the average delay of non-real-time messages versus the non-real-time traffic load is shown in Fig. 10. At moderate loads, the reductions in the response time are around 50%. Thus, we conclude that the proposed approach reduces the response time of non-real-time messages under almost all circumstances.

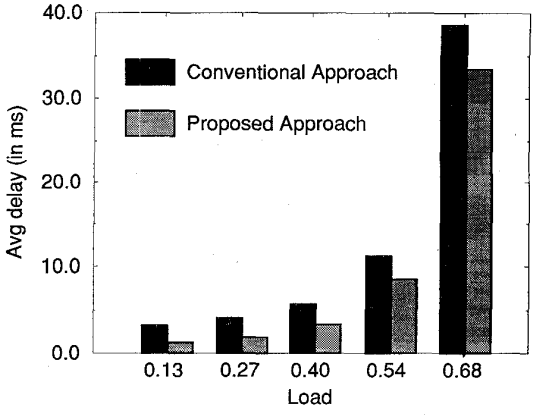


Fig. 10. Average delay of non-real-time messages for the system described in Table IV.

TABLE IV
A SYSTEM WITH BURSTY NON-REAL-TIME STREAMS.
TTRT = 8.325ms AND τ = 1ms

Station #	Real-time stream	Non-real-time stream	H
1-3	P = 33ms, D = 33ms C_{min} = 0.5ms, C_{max} = 2ms	P = 20ms C_{min} = 0.1ms, C_{max} = 0.9ms	0.916ms
4-6	P = 100ms, D = 100ms C_{min} = 1ms, C_{max} = 10ms	ON_{ave} = 50ms, OFF_{ave} = 200ms	1.525ms
7-20	—		0

VI. CONCLUSION

Real-time messages have deadlines by which they must reach their destinations. The exact delivery time of a real-time message is unimportant as long as it meets its deadline. The non-real-time messages, on the other hand, do not have deadlines.

In this paper, we proposed an approach that takes advantage of the timing properties of the timed token protocol to reduce the response time of non-real-time messages while still guaranteeing the deadlines of the real-time messages. Whenever possible, the transmission of real-time messages is delayed in favor of non-real-time messages. We presented a simple algorithm that can be used by a station to determine if it can defer the transmission of its real-time messages to a later time and still guarantee timely delivery of the real-time messages. An empirical evaluation shows that the proposed approach yields a substantial reduction in the response time of non-real-time messages.

ACKNOWLEDGMENTS

The work reported here is supported in part by the National Science Foundation under Grant MIP-9213716. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] "Fiber distributed data interface (FDDI)—Token ring media access control (MAC)," ANSI Standard X3.139, 1987.

[2] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed token protocol," *Proc. Distributed Computing Systems*, pp. 468–475, June 1992.

[3] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed token medium access control protocol," *IEEE Trans. Computers*, vol. 43, no. 3, pp. 327–339, Mar. 1994.

[4] B. Chen, G. Agrawal, and W. Zhao, "Optimal synchronous capacity allocation for hard real-time communications with the timed token protocol," *Proc. Real-Time Systems Symp.*, pp. 198–207, Dec. 1992.

[5] D. Dykeman and W. Bux, "Analysis and tuning of the FDDI media access control protocol," *IEEE J. Selected Areas in Comm.*, vol. 6, no. 6, pp. 997–1,010, July 1988.

[6] D. Gall, "MPEG: A video compression standard for multimedia applications," *Comm. ACM*, pp. 46–58, Apr. 1991.

[7] R.M. Grow, "A timed token protocol for local area networks," *Proc. Electro 82*, May 1982.

[8] M. Hamdaoui and P. Ramanathan, "Selection of timed token MAC protocol parameters to guarantee message deadlines," Technical Report ECE-92-10, Dept. of Electrical and Computer Eng., Univ. of Wisconsin–Madison, Nov. 1992.

[9] M. Hamdaoui and P. Ramanathan, "Selecting timed token protocol parameters to guarantee real-time messages," *Proc. Parallel and Distributed Real-Time Systems Workshop*, Apr. 1993.

[10] R. Jain, "Performance analysis of FDDI token ring networks: Effect of parameters and guidelines for setting TTRT," *IEEE Lightwave Telecommunications Systems*, vol. 2, no. 2, pp. 16–22, May 1991.

[11] A.P. Jayasumana and P.N. Werahera, "Performance of fibre distributed data interface network for multiple classes of traffic," *IEE Proc.*, vol. 137, Pt. E, no. 5, pp. 401–408, Sept. 1990.

[12] M.J. Johnson, "Proof that timing requirements of the FDDI token ring protocol are satisfied," *IEEE Trans. Computers*, vol. 35, no. 6, pp. 620–625, June 1987.

[13] S.P. Joshi, "High-performance networks: A focus on the fiber distributed data interface (FDDI) standard," *IEEE Micro*, pp. 8–14, June 1986.

[14] M. Keshtgary and A.P. Jayasumana, "Bandwidth allocation in FDDI-II for isochronous, synchronous and asynchronous traffic," *Proc. Conf. Local Computer Networks*, pp. 196–204, Sept. 1993.

[15] N. Malcolm and W. Zhao, "Guaranteeing synchronous messages with arbitrary deadline constraints in an FDDI network," *Proc. Conf. Local Computer Networks*, pp. 186–195, Sept. 1993.

[16] P. Montuschi, A. Valenzano, and L. Ciminiera, "Selection of token holding times in timed-token protocols," *IEEE Trans. Industrial Electronics*, vol. 37, no. 6, pp. 442–451, Dec. 1990.

[17] J. Pang and F.A. Tobagi, "Throughput analysis of a timer controlled token passing protocol under heavy load," *IEEE Trans. Computers*, vol. 37, no. 7, pp. 694–702, July 1989.

[18] F.E. Ross, "FDDI—A tutorial," *IEEE Comm. Magazine*, vol. 24, no. 5, pp. 10–17, May 1986.

[19] F.E. Ross, "An overview of FDDI: The fiber distributed data interface," *IEEE J. Selected Areas in Comm.*, vol. 7, no. 7, pp. 1,043–1,051, Sept. 1989.

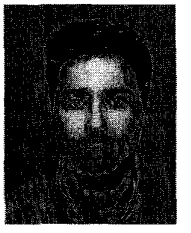
[20] M.A. Saleh, I.W. Habib, and T.N. Saadawi, "Simulation analysis of a communication link with statistically multiplexed bursty voice sources," *IEEE J. Selected Areas in Comm.*, vol. 11, no. 3, pp. 432–442, Apr. 1993.

[21] R. Sankar and Y.Y. Yang, "Performance analysis of FDDI," *Proc. IEEE Conf. Local Computer Networks*, pp. 328–332, Oct. 1989.

[22] K.C. Sevcik and M.J. Johnson, "Cycle time properties of the FDDI token ring protocol," *IEEE Trans. Software Engineering*, vol. 13, no. 3, pp. 376–385, 1987.

[23] K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data," *IEEE J. Selected Areas in Comm.*, vol. 4, no. 6,, Sept. 1986.

[24] J.N. Ulm, "A timed token ring local area network and its performance characteristics," *Proc. IEEE Conf. Local Computer Networks*, pp. 50–56, Feb. 1982.



Moncef Hamdaoui received the BS (with highest distinction), MS, and PhD degrees in electrical and computer engineering from the University of Wisconsin, Madison, in 1988, 1990, and 1994, respectively. He is currently working for Northern Telecom in Tunis, Tunisia.

His research interests include real-time systems, communication networks, and fault-tolerant systems. He is a member of the IEEE Computer Society.

Parameswaran Ramanathan received the BTech degree from the Indian Institute of Technology, Bombay, India, in 1984, and the MSE and PhD degree from the University of Michigan, Ann Arbor, in 1986 and 1989, respectively.

From 1984 to 1989, he was a research assistant in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. Dr. Ramanathan is now an associate professor in the Department of Electrical and Computer Engineering and in the Department of Computer Sciences at the University of Wisconsin, Madison. His research interests include the areas of real-time systems, fault-tolerant computing, distributed systems, and parallel algorithms.