The Timely-Token Protocol*

Jorge A. Cobb† Miaohua Lin
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
{cobb,miaohua}@utdallas.edu

November 7, 2003

Abstract

The token protocols in FDDI and FDDI-M support synchronous and asynchronous traffic. However, FDDI suffers from a token-lateness problem, and FDDI-M may starve asynchronous traffic. To remove these two weaknesses, we propose the timely-token protocol. The main feature of this protocol is the addition of information to the token so that each station may determine the reason of an early token arrival. Using this information, each station limits its asynchronous transmission to ensure that the token is not late at subsequent stations. In addition, we present a synchronous allocation scheme for this protocol. Finally, a comparison is made against FDDI and FDDI-M.

Keywords: FDDI, FDDI-M, local-area-networks, quality-of-service.

1 Introduction

In this paper, we present an enhancement to the timed-token protocol proposed by Grow [1]. This protocol has been incorporated into the Fiber

*Supported in part by a grant from the Texas Advanced Research Program.
†Corresponding author: phone 972 883 2479, fax 972 883 2349

Distributed Data Interface (FDDI), and its purpose is to support a mixture of synchronous and asynchronous traffic in a token-ring network. Synchronous messages are transmitted periodically and have a deadline, while asynchronous messages are transmitted on a best-effort basis.

During initialization, a protocol parameter, called Target Token Rotation Time ($TTRT$), is chosen, and it indicates the expected time of each token rotation. Each station i is assigned a portion S_i of the $TTRT$. This is the maximum time station i is allowed to transmit synchronous traffic upon receiving the token. The remaining portion of the $TTRT$, A^*, where $A^* = TTRT - \sum_i S_i$, is the time available to transmit asynchronous messages during each token rotation.

Given the value of $TTRT$, a *Synchronous Allocation* (SA) *scheme* assigns a value of S_i to each station i. Several SA schemes have been proposed for FDDI [2, 4, 5].

In FDDI, the rotation time of the token may exceed $TTRT$. Due to this lateness of the token, an FDDI ring can use at most half of its bandwidth to transmit synchronous traffic [6]. To alleviate this deficiency, Shin et al. proposed the FDDI-M token protocol [6]. In FDDI-M, the token is never late. This allows FDDI-M to support synchronous traffic with a larger range of deadline constraints than FDDI.

However, FDDI-M has some weaknesses. First, in some cases, FDDI-M may not be able to transmit asynchronous traffic, although intuitively it should. That is, there are scenarios where the portion A^* of the $TTRT$ is non-zero, yet asynchronous traffic is starved. In addition, the analysis of SA schemes for FDDI-M is quite unintuitive and unmanageable, and does not lead to closed form expressions. Therefore, although there is a large number of SA schemes for FDDI, such as PA [7], FLA [7], MCA [2], EMCA [5], ELA [8] and EGA [9] (an overview of most of the above SA schemes and more may be found in [10] and [11]), no SA schemes have been designed specifically for FDDI-M.

In this paper, we present the timely-token protocol. In this protocol, the token is never late. In addition, at every token rotation, A^* time units are available for asynchronous traffic. Furthermore, the protocol is simple and yields a straightforward SA scheme.

The paper is organized as follows. The network and traffic models are introduced in Section 2. FDDI and FDDI-M, along with their weaknesses, are described in Section 3. The timely-token is described in Section 4. An SA scheme for the timely-token is given in Section 5. Section 6 compares

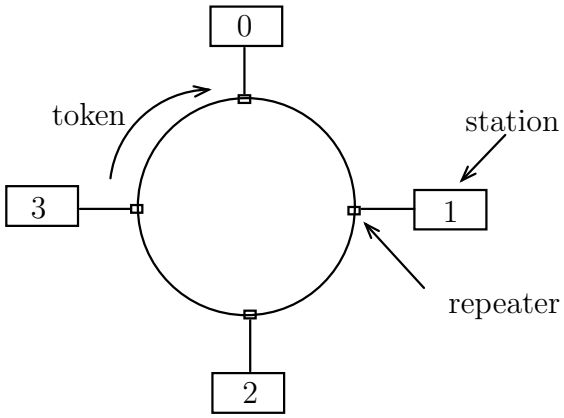

Figure 1: Token-ring network.

the timely-token against FDDI and FDDI-M. In Section 7, we show how the timely-token can be implemented with each repeater storing only a couple of bits. Finally, concluding remarks are given in Section 8.

2 Token Protocols and Traffic Model

The token protocols in this paper operate on a ring consisting of N stations. Each station has a unique number in the range $0 \,..\, N-1$. In addition, each station is connected to two others by a unidirectional medium that forms a single closed path (see Figure 1). For each station i, the next station along the unidirectional medium is station $(i+1) \bmod N$.

A small message, called *token*, circulates when all stations are idle. Before a station can transmit, it must first grasp the token. After transmitting, the station releases the token.

During each token rotation, there is a constant amount of time, denoted by τ, that is unusable for transmission. The value of τ includes the propagation delay around the ring, and a few bits of store-and-forward delay at each station. Note that the definition of A^* given in the introduction needs to be modified slightly as follows:

$$A^* = TTRT - \tau - \sum_i S_i$$

A *synchronous stream* is described by a triple, (P, D, C), where P is the inter-arrival time of messages from the stream, D is an upper bound on message delay, and C is the maximum time needed to transmit a message of the stream. Although most SA schemes assume $D = P$, we do not make this assumption. However, we do assume $D \leq P$. Also, messages may be fragmented into smaller packets, and packet transmission cannot be preempted.

Agrawal et al. [7] showed how a token-ring network having multiple synchronous streams per station could be transformed into a logically-equivalent network with one synchronous stream per (logical) station. Therefore, without loss of generality, we assume a single synchronous stream per station.

A *synchronous stream set* consists of one synchronous message stream per station. The synchronous stream of station i is denoted by the triple (P_i, D_i, C_i).

3 Previous Token Protocols

In this section, we provide an overview of the token protocols of FDDI and FDDI-M. In addition, we describe weaknesses in each of them. These weaknesses are overcome by the timely-token protocol given in the next section.

3.1 FDDI Token Protocol

We next describe the token protocol used in FDDI[1]. During ring initialization, each station i supporting synchronous traffic is assigned a fixed portion S_i of the $TTRT$. During each token rotation, station i can transmit synchronous packets for at most S_i time units.

Each station i has a token-rotation timer, TRT_i. This timer always increases. However, it may be reset to zero. The purpose of this timer is to measure the time between token arrivals.

If TRT_i reaches a value of at least $TTRT$ before the token arrives at the station, then TRT_i is reset to zero, and the token is marked as "late" by incrementing the station's late count L_i by one.

Each station i has an asynchronous-limit variable, A_i. In this variable, station i stores the amount of time it may transmit asynchronous messages.

[1]This description differs from the typical description in the literature. However, it is equivalent. We present this alternative definition for clarity and ease of comparison with the timely-token.

Contrary to S_i, the value of A_i may vary each time the token is received.

When a station receives the token, it performs the following steps:

1. If $L_i > 0$, set $L_i := L_i - 1$ and $A_i := 0$. Otherwise, $A_i := TTRT - TRT_i$ and $TRT_i := 0$.

2. If the station has synchronous packets, it transmits them for a time period of up to S_i time units, or until all the synchronous packets are transmitted, whichever occurs first.

3. If station i has asynchronous packets, it transmits them for a time period of up to A_i time units, or until all the asynchronous packets are transmitted, whichever occurs first.

4. Station i passes the token to station $(i + 1) \bmod N$.

To initialize all timers, no packets are allowed to be transmitted during the first token rotation. Also, all late counts L_i are initialized to zero.

3.2 Late Token in FDDI

Consider a ring network with four stations, as shown in Figure 1. Assume there is no traffic (either synchronous or asynchronous) in the previous token rotation before the token arrives to station zero. The ring parameters are as follows: $TTRT = 100$, and for each i, $S_i = 20$. For simplicity, $\tau = 0$.

When station zero receives the token, $TRT_0 = 0$ and $A_0 = TTRT - TRT_0 = TTRT$. Assume station zero has asynchronous packets, but it has no synchronous packets. Also assume that immediately after the station transmit its first asynchronous packet, it generates a synchronous message. Station zero then transmits asynchronous packets for $TTRT$ time units, and releases the token to station one.

After this, assume all stations have plenty of synchronous and asynchronous packets. Then, stations one, two, and three transmit their synchronous packets for 20 time units each (because $S_1 = S_2 = S_3 = 20$). Thus, the token rotation time of stations one, two, and three is 100, 120, and 140, respectively, all of which are at least $TTRT$. These stations receive a late token, and cannot transmit asynchronous packets.

When the token reaches station zero for a second time, the token rotation time will be $TTRT + \sum_{i=1}^{3} S_i = 160$. This is greater than the $TTRT$, which

equals 100. The $TTRT$ is considered to be the expected token rotation time, but from this example we see that the token rotation time may significantly exceed the $TTRT$.

Note that as the token rotation time grows, the ability to satisfy a set of synchronous streams diminishes. Thus, the direct effect of late tokens is the reduced ability to schedule synchronous traffic. In this example, the synchronous message arriving at station zero experiences a waiting time of 160 time units. For its deadline to be satisfied, it must be greater than 160.

When station zero transmits asynchronous traffic for $TTRT$ time units, it takes advantage of unused synchronous bandwidth of other stations. However, each station is allowed to transmit S_i time units of synchronous traffic at *every* rotation. Thus, at the *next* rotation, the other stations transmit their allotted S_i time units of synchronous bandwidth, causing the token to arrive late at station zero. From this observation, we conclude that *the lateness of the token is due to a station taking advantage of unused synchronous bandwidth of other stations*. This observation plays a significant role in the design of the timely-token of Section 4.

3.3 FDDI-M Token Protocol

We next overview the token protocol of FDDI-M[2]. Similar to FDDI, during ring initialization, each station i supporting synchronous traffic is assigned a fixed portion S_i of the $TTRT$. During each token rotation, station i can transmit at most S_i time units of synchronous packets. Also similar to FDDI, each station i has an asynchronous-limit variable, A_i, and a token-rotation timer TRT_i. To initialize the token-rotation timers, no packets are transmitted during the first token rotation.

When station i receives the token, it performs the following steps:

1. $A_i := TTRT - (TRT_i + \sum_i S_i) = A^* + \tau - TRT_i$.
2. If station i has synchronous packets, it transmits them for a time period of up to S_i time units, or until all the synchronous packets are transmitted, whichever occurs first.
3. $TRT_i := 0$.

[2]Again, this description differs from the typical description in the literature. However, it is equivalent. We present this alternative definition for clarity and ease of comparison with the timely-token.

4. If station i has asynchronous packets, it transmits them for a time period of up to A_i time units, or until all the asynchronous packets are transmitted, whichever occurs first.

5. Station i passes the token to station $(i + 1) \bmod N$.

3.4 Starvation of Asynchronous Traffic in FDDI-M

We next present an example where asynchronous traffic is starved in FDDI-M, even though not all of the $TTRT$ has been allocated to synchronous traffic. The example is similar to the one in Section 3.2.

Consider four stations, as shown in Figure 1. Let $TTRT = 100$, and for each i, $S_i = 20$. For simplicity, $\tau = 0$. Thus, $A^* = TTRT - \sum_i S_i = 20$.

Assume there is no traffic before the token reaches station zero in the first rotation. When the token reaches station zero again, let there be plenty of both synchronous and asynchronous traffic at all stations. Then, $TRT_0 = 0$, and $A_0 = A^* - TRT_0 = A^* = 20$. That is, station zero can transmit asynchronous packets up to the total ring asynchronous allocation A^*, and also its own synchronous allocation S_i. For the following stations, the values of TRT are 40, 60 and 80, respectively. Because these values are all greater than A^*, these stations cannot transmit any asynchronous packets.

When the token reaches station zero for the third time, again, let there be plenty of synchronous and asynchronous traffic at all stations. Recall that in FDDI-M, TRT_i is reset to zero *after* transmitting the synchronous traffic. Thus, when the token returns to station zero, $TRT_0 = A_0 + \sum_{i=1}^{3} S_i = 20 + 60 = 80$, which is larger than A^*. Therefore, station zero is unable to transmit asynchronous traffic. Nonetheless, station zero does transmit its synchronous traffic. For the following stations, the value of TRT_i, $1 \leq i \leq 3$, is 60 (counting all synchronous bandwidth used by other stations). This, however, is larger than A^*. Thus, no asynchronous traffic is transmitted in this rotation even though there are asynchronous packets at every station. Even worse, the situation continues as long as there are plenty of synchronous packets in each station during the following rotations.

In FDDI-M, the token is never late. That is, $TRT_i \leq TTRT - S_i$ whenever station i receives the token. Although not mentioned in [6], we believe the reason the token is never late is as follows. Recall our earlier argument that a late token in FDDI is caused by a station taking advantage of unused synchronous bandwidth from other stations. In FDDI-M this is not possible,

as follows. FDDI-M makes the *pessimistic* assumption that no synchronous packets were transmitted during the last rotation. Therefore, to prevent a late token, the station does not take advantage of any synchronous bandwidth from any station. To ensure this, $\sum_i S_i$ is added to TRT_i when calculating A_i.

4 Timely-Token Protocol

In this section, we present the timely-token protocol. This protocol overcomes the two weaknesses mentioned in the previous section.

4.1 Insight

In FDDI, tokens arrive late because a station may take advantage of unused synchronous bandwidth during the token rotation. In FDDI-M, to prevent late tokens, no station takes advantage of unused synchronous bandwidth. However, each station is unaware of how much synchronous bandwidth was used during the token rotation. Thus, the worst-case is assumed. That is, it is assumed that no synchronous bandwidth was used. To avoid taking advantage of this unused bandwidth, a total of $\sum_i S_i$ is added to TRT_i in the calculation of A_i. However, in doing so, asynchronous bandwidth may be starved.

The above discussion implies that if a station becomes aware of how much synchronous bandwidth was left unused by other stations, it could accurately determine how much to add to TRT_i as opposed to the worst case assumption of $\sum_i S_i$. To achieve this, an integer u is added to the token, where u represents the sum of unused synchronous bandwidth of all stations during the previous token-rotation. When the token arrives to station i, u should also include the unused synchronous bandwidth of station i in the previous token-rotation.

We next present an outline of the timely-token[3].

[3]We presented an earlier version of this protocol in [3]. However, in [3], it is possible for a station to use A^* time units of asynchronous bandwidth at every token rotation, and starve other stations of asynchronous traffic. The protocol we present below does not suffer from this drawback.

4.2 Outline

As in the previous protocols, each station i has a token-rotation timer, TRT_i, an asynchronous-limit variable, A_i, and a constant portion S_i of the $TTRT$. In addition, station i maintains a variable s_i, where it stores the time units of synchronous bandwidth it used in the previous token-rotation.

When station i receives the token, it performs the following steps:

1. $A_i := (TTRT - u - TRT_i)^+$
2. $TRT_i := 0$
3. $u := u - (S_i - s_i)$
4. If station i has synchronous packets, it transmits them for a time period of up to S_i time units, or until all its synchronous packets are transmitted, whichever occurs first.
5. s_i is assigned the number of time units of synchronous transmission used in step 4.
6. $u := u + (S_i - s_i)$
7. If station i has asynchronous packets, it transmits them for a time period of up to A_i time units, or until all its asynchronous packets are transmitted, whichever occurs first.
8. Station i passes the token to station $(i + 1) \bmod N$.

During the first token rotation, to initialize timers, no station is allowed to transmit any packets. In addition, s_i is set to zero for all i, and $u = \sum_i S_i$. Note that A_i is initially the same as in FDDI-M.

We next present an example showing how synchronous and asynchronous bandwidth are managed using the timely-token. It is the same example given in Section 3.2 that caused the token to be late in FDDI.

Consider a ring network with four stations, as shown in Figure 1. Assume there is no traffic (either synchronous or asynchronous) in the previous rotation before the token arrives to station zero. The ring parameters are as follows: $TTRT = 100$, and for each i, $S_i = 20$. For simplicity, $\tau = 0$. Note that $A^* = TTRT - \sum_i S_i = 20$.

When station zero receives the token, $TRT_0 = 0$, $u = \sum_i S_i = 80$ (since no synchronous packets were transmitted in the first rotation), and $A_0 = TTRT - (TRT_0 + u) = 20 = A^*$. Therefore, the station can transmit asynchronous packets for the entire asynchronous allocation A^*.

Assume station zero has asynchronous packets to transmit, but it has no synchronous packets. Immediately after the station begins to transmit its first asynchronous packet, a synchronous message arrives at the station. The station then transmits asynchronous packets for A_0 (i.e., A^*) time units, and releases the token to station one. The value of u in the token remains 80, since station zero did not transmit synchronous packets.

Assume next that all stations have synchronous and asynchronous packets. At station one, $TRT_1 = 20$ and $u = 80$. Thus, $A_1 = 0$, and no asynchronous packets are transmitted. However, station one does transmit $S_1 = 20$ time units of synchronous traffic. It then forwards the token to station two, with an updated u value equal to 60, because station one used its entire synchronous allocation.

At station two, $TRT_2 = 40$ and $u = 60$. Thus, $A_2 = 0$, and no asynchronous packets are transmitted. Station two transmits $S_2 = 20$ time units of synchronous traffic. It then forwards the token to station three, with an updated u value equal to 40, because station two used its entire synchronous allocation.

The scenario at station three is similar. It forwards the token to station zero, with an updated u value equal to 20.

When the token reaches station zero again, $TRT_0 = 80$ (includes the asynchronous traffic of station zero and the synchronous traffic of the other stations) and $u = 20$. Thus, $A_0 = 0$, and it may not transmit asynchronous traffic. Note, however, that since $TRT_0 = 80$, the synchronous message that originated in the previous rotation may now be transmitted, and its delay is only 80, as opposed to 160 in $FDDI$. Thus, assume station zero transmits $S_i = 20$ time units of synchronous traffic, and forwards the token to station one with $u = 0$.

Note that at station one, $TRT_1 = 80$ and $u = 0$. Thus, $A_1 = 20 = A^*$. This allows station one to transmit asynchronous traffic, if desired. Similarly, if station one does not transmit asynchronous traffic, stations two and three may do so.

By continuing this example, we see that synchronous and asynchronous bandwidth can both be used if every station has unlimited synchronous and asynchronous packets. Also, even though the bandwidth is fully utilized, the

token is never late. We formalize this below.

4.3 Performance Bounds

We next formalize some bounds on the behavior of the timely-token. In particular, we show that the token is never late, and the transmission of asynchronous traffic is guaranteed.

For the results of this section and the next, the following constraints must be satisfied. We therefore consider only those synchronous stream sets satisfying these constraints.

Definition 1 *The protocol constraints of the timely-token are as follows.*

- *The required transmission time of a message must be at most the deadline of this message.*

$$(\forall\, i,\ C_i \leq D_i \leq P_i)$$

- *The message transmission time must be at most the available portion of the $TTRT$.*

$$(\forall\, i,\ C_i \leq TTRT - \tau)$$

- *The sum of the synchronous allocations to all stations must be at most the available portion of the $TTRT$.*

$$\sum_i S_i \leq TTRT - \tau$$

In order to reason about values that change over time, we enhance our notation to include rounds, that is, token rotations.

Definition 2

$R^{i,m}$: *round m of station i, i.e., time interval $[t, t']$, where t is the time when station i receives the token for the m*th *time, and t' is the time when station i receives the token for the $(m+1)$*th *time.*

$A_j^{i,m}$: *value assigned to* A_j *during* $R^{i,m}$. *In particular,* $A_i^{i,m}$ *is the value assigned to* A_i *when the token is received at the beginning of* $R^{i,m}$

$TRT_j^{i,m}$: *value of* TRT_j *when station* j *receives the token during* $R^{i,m}$. *In particular,* $TRT_i^{i,m}$ *is the value of* TRT_i *when the token is received at the beginning of* $R^{i,m}$

$a_j^{i,m}$: *duration of asynchronous transmission of station* j *during* $R^{i,m}$. *Note that* $a_j^{i,m} \leq A_j^{i,m}$.

$s_j^{i,m}$: *duration of synchronous transmission of station* j *during* $R^{i,m}$. *Note that* $s_j^{i,m} \leq S_j$.

Before presenting the theorems, we require the following two lemmas.

Lemma 1 *For every* i *and* m, *when station* i *receives the token at the end of* $R^{i,m}$, *the value of* u *in the token corresponds to the unused synchronous time of all stations during* $R^{i,m}$.

Proof: The proof is by induction on the number of stations visited by the token. For the base case, consider the ring after initialization. No packets are transmitted at the first token rotation, and furthermore, $u = \sum_j S_j$ and $s_j = 0$ for all j. This satisfies the lemma.

For the induction step, consider a station i, and assume the lemma holds for the previous station k. When the token arrives at i, for u to have the correct value, the unused synchronous time of k from the previous round must be removed from u, and the unused synchronous time of k from this round must be added to u. These steps are performed by station k according to the outline of the timely-token. Hence, the lemma holds when station i receives the token. ■

Lemma 2 *For every* i, j, m,

$$A_i^{i,m} = \left(A^* - \sum_j a_j^{i,(m-1)} \right)^+$$

Proof: From the definition of the asynchronous limit variable,

$$A_i^{i,m} = (TTRT - u - TRT_i^{i,m})^+$$

From Lemma 1,

$$u = \sum_j S_j - \sum_j s_j^{i,(m-1)}$$

From the definition of $TRT_i^{i,m}$,

$$TRT_i^{i,m} = \sum_j s_j^{i,(m-1)} + \sum_j a_j^{i,(m-1)} + \tau$$

Therefore,

$$\begin{aligned} A_i^{i,m} &= \left(TTRT - \left(\sum_j S_j - \sum_j s_j^{i,(m-1)} \right) - \left(\sum_j s_j^{i,(m-1)} + \sum_j a_j^{i,(m-1)} + \tau \right) \right)^+ \\ &= \left(TTRT - \sum_j a_j^{i,(m-1)} - \sum_j S_j - \tau \right)^+ \\ &= \left(A^* - \sum_j a_j^{i,(m-1)} \right)^+ \end{aligned}$$

■

Theorem 1 (Token is never late) *For every station i, upon token arrival, $TRT_i \leq TTRT$.*

Proof: Consider round $R^{i,m}$. From Lemma 2, we have $A_j^{i,(m-1)} \leq A^*$ for all j. Furthermore, if station j transmits $a_j^{i,(m-1)}$ time units of asynchronous traffic during $R^{i,(m-1)}$, then for any station k after j and before i, $A_k^{i,(m-1)}$ decreases by $a_j^{i,(m-1)}$. This is because $TRT_k^{i,(m-1)}$ increases by $a_j^{i,(m-1)}$. Therefore,

$$\sum_j a_j^{i,(m-1)} \leq A^* \tag{1}$$

Furthermore,

$$\begin{aligned} TRT_i^{i,m} &= \sum_j s_j^{i,(m-1)} + \sum_j a_j^{i,(m-1)} + \tau \\ &\leq \sum_j s_j^{i,(m-1)} + A^* + \tau \\ &\leq \sum_j S_j + A^* + \tau \\ &= TTRT \end{aligned}$$

■

Theorem 2 (Asynchronous traffic guarantee) *For each i and m, a total of A^* time units become available for asynchronous transmission during $R^{i,m}$.*

Proof: Consider $A_i^{i,(m+1)}$. From Lemma 2,

$$A_i^{i,(m+1)} = \left(A^* - \sum_j a_j^{i,m} \right)^+$$

We therefore have that

$$\sum_j a_j^{i,m}$$

time units have been used, and therefore became available, for asynchronous transmission during $R_{i,m}$. The remaining time units, if any, are assigned to $A_i^{i,(m+1)}$. Therefore, a total of A^* time units become available for asynchronous transmission during every round of every station. ■

From the above theorems, we see that the maximum token rotation time is upper-bounded by $TTRT$ (the token is never late), and this bound is tight. Furthermore, asynchronous bandwidth is always available to the stations at each rotation (provided $A^* > 0$). In particular, if the timely-token is used in the scenarios of Sections 3.2 and 3.4, the token is never late, and the asynchronous traffic is not starved.

5 Synchronous Bandwidth Allocation Scheme

There are two approaches to configure a token protocol to support a synchronous stream set. First, the $TTRT$ is given, and each S_i value is chosen to support the synchronous streams. This approach is known as a *Synchronous Allocation* (SA) scheme. In this section, we present a SA scheme for the timely-token. The second approach consists of choosing a $TTRT$ value given the synchronous stream set. We consider this approach in Section 6.

Before presenting our SA scheme, we measure how much synchronous transmission time is available at a station before the deadline expires. This measure was introduced in [5], and is defined as follows.

Definition 3 *Given any interval of size D_i, X_i is the minimum amount of time during which station i can transmit synchronous packets in this interval.*

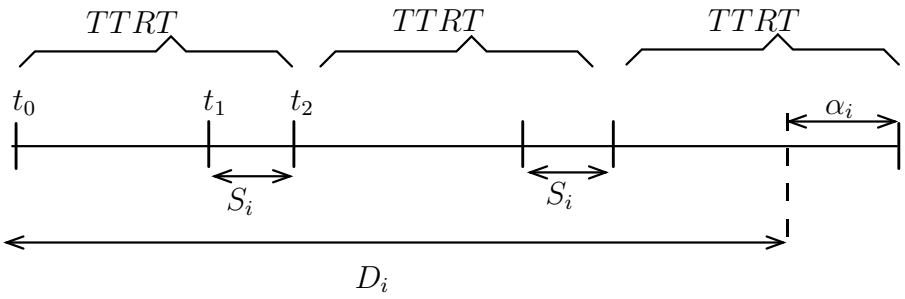

Figure 2: SA scheme illustration

Theorem 3 *For the timely-token,*

$$X_i = m_i \cdot S_i + (S_i - \alpha_i)^+$$

where m_i and α_i are as follows.

$$m_i = \lfloor D_i / TTRT \rfloor$$
$$\alpha_i = (m_i + 1) \cdot TTRT - D_i$$

Proof: Figure 2 illustrates the values of m_i and α_i used in our SA scheme ($m_i = 2$ in the figure). Our objective is to obtain the worst case scenario, that is, station i is able to transmit the least possible amount of synchronous traffic.

At time t_0, i.e., at the beginning of the interval of size D_i, station i either forwards the token or begins transmitting asynchronous packets. Note that at this time the station will not transmit synchronous packets until it receives the token again. Let t_1, be the time the token arrives again at station i.

The largest value of $t_1 - t_0$ is as follows. From Equation (1), the total asynchronous bandwidth used between t_1 and t_0, including that of station i, is at most A^*. Therefore, since all stations can transmit their allocation of synchronous bandwidth, $t_1 - t_0$ is at most

$$A^* + \sum_{j \neq i} S_j + \tau$$

Therefore, $t_2 - t_0 \leq TTRT$. In this way, station i receives the token every $TTRT$ time units during the interval of size D_i. From Theorem 1, this is the worst case since the token is never late.

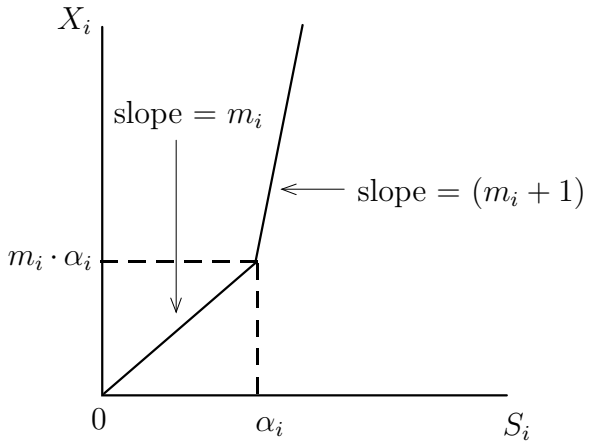

Figure 3: X_i as a function of S_i.

Thus, station i will be able to use its full S_i time units of synchronous transmission for a total of m_i times. During the last token arrival, it will only be able to transmit $S_i - \alpha_i$ time units of synchronous packets before the end of the interval.

■

We base our SA scheme on Theorem 3 above. Our objective is to find values of each S_i that ensure that each synchronous message of station i is transmitted within D_i time units. That is, we require $X_i \geq C_i$. However, S_i should be as small as possible. This ensures that the largest possible set of synchronous streams is supported. To minimize S_i, we choose S_i such that $X_i = C_i$.

Assume for the moment that for all i, $D_i \geq TTRT$. Figure 3 shows X_i as a function of S_i. To determine the value of S_i such that $X_i = C_i$, we must consider two cases. First, $C_i \leq m_i \cdot \alpha_i$. In this case, $C_i = S_i \cdot m_i$, and thus,

$$S_i = \frac{C_i}{m_i}$$

Second, $m_i \cdot \alpha_i > C_i$. In this case,

$$C_i = (S_i - \alpha_i) \cdot (m_i + 1) + (m_i \cdot \alpha_i)$$

Therefore,

$$S_i = \frac{C_i + \alpha_i}{m_i + 1}$$

Consider now the remaining case where $D_i < TTRT$ for some i. In this case, $m_i = 0$, and thus, $X_i = 0$. To avoid this situation, we must force the token to rotate faster. This is done by introducing a fictitious station g, and choosing S_g as follows.

$$S_g = TTRT - D_{min}$$

Above, D_{min} is the minimum message deadline.

A specific station in the ring is assigned the duties of station g. In particular, since g never transmits traffic, this station ensures that S_g is always included in u. Note that in this manner, no other station can use the synchronous allocation S_g to transmit asynchronous packets. Therefore, each token rotation will be at most $TTRT - S_g$, i.e., at most D_{min}.

To compute the allocation of the remaining stations, let $TTRT' = D_{\min}$. We assign to each S_i the same value as computed earlier, but we use $TTRT'$ in place of $TTRT$.

In summary, the SA scheme of the timely-token is as follows:

Case 1: if for all i, $D_i \geq TTRT$, then
  if $C_i \leq m_i \cdot \alpha_i$,
    let $S_i = C_i/m_i$
  otherwise,
    let $S_i = (C_i + \alpha_i)/(m_i + 1)$

Case 2: if there is an i, such that $D_i < TTRT$, then
  let $S_g = TTRT - D_{min}$, for a fictitious station g.
  let S_j be computed as in case 1 for all j, $j \neq g$,
    except that $TTRT'$ is used instead of $TTRT$,
    where $TTRT' = D_{min}$.

The above SA scheme ensures all messages are transmitted by their deadlines. This is formalized in the following theorem.

Theorem 4 *The SA scheme for the timely-token ensures that the set of synchronous streams is schedulable, i.e., no message deadline is violated. This is provided the protocol constraints hold.*

Proof: Given our initial assumption of $D_i \leq P_i$ for all i, note that each synchronous message of station i needs to be transmitted before the arrival

of the next synchronous message. Therefore, if $X_i \geq C_i$, this is sufficient for each message to be transmitted by its deadline. Above, we have shown that $X_i = C_i$ for all cases. Thus, each message will be transmitted no later than its deadline. ■

We conclude this section by applying the SA scheme to a couple of examples.

First, let $TTRT = 100$, $N = 4$, and, for each i, $C_i = 20$ and $D_i = 100$. For simplicity, assume $\tau = 0$. This implies that for each i, $m_i = 1$ and $\alpha_i = 100$. Therefore, $m_i \cdot \alpha_i > C_i$ and $S_i = C_i/m_i = 20$. We need to check, however, if the protocol constraints are satisfied. In particular, $\sum_i S_i = 80 < TTRT - \tau = 100$. Since the protocol constraints are satisfied, the synchronous stream set is schedulable.

Next, consider another synchronous stream set as above, with the exception that, for each i, $D_i = 150$ and $C_i = 60$. This implies that, for each i, $m_i = 1$, $\alpha_i = 50$. Therefore, $m_i \cdot \alpha_i < C_i$, and $S_i = (C_i + \alpha_i)/(m_i + 1) = (60 + 50)/2 = 55$. When we check the protocol constraints we obtain: $\sum_i S_i = 220 > TTRT - \tau = 100$. Thus, this synchronous message set is not schedulable through our SA scheme.

6 Comparison Against FDDI and FDDI-M

In this section, we compare the timely-token against the FDDI and FDDI-M. Our comparison focuses on the ability of these protocols to support synchronous traffic.

We begin by comparing the timely-token against FDDI. We base our comparison on the value of X_i in these protocols, because this value directly reflects the ability of the protocol to provide service to a synchronous stream i. The value of X_i in FDDI, however, is dependent on how often a station is able to transmit synchronous traffic. This notion is represented by function I [5], whose definition follows.

Definition 4 *For any positive integer n, $I(n)$ is an upper bound on the maximum possible time that may elapse before any station i with synchronous messages uses $n \cdot S_i$ time units of synchronous transmission.*

Note that function I is independent of the station number. This is true for both the timely-token and FDDI, as we will see below.

From the above definition, consider a time interval

$$[t,\ t + I(n)]$$

for some t. Each station i is allowed to transmit $n \cdot S_i$ time units of synchronous traffic during this interval. This can be used to derive the value of X_i, as follows [5]. Consider an integer n_i, such that

$$I(n_i - 1) \leq D_i < I(n_i)$$

Consider now the interval

$$[t,\ t + D_i]$$

for some t. From above, station i is able to transmit synchronous traffic for at least $(n_i - 1) \cdot S_i$ time units during this interval. This is because $I(n_i - 1) \leq D_i$. Furthermore, station i may be able to transmit a portion of the next S_i synchronous time units. In the worst case, these S_i time units are given to station i starting at time $t + I(n_i) - S_i$. This implies that an additional

$$D_i - (I(n_i) - S_i)$$

synchronous time units are available to station i before the end of the interval of size D_i. Thus, X_i is as follows.

$$X_i = (n_i - 1) \cdot S_i + \max[0, D_i - I(n_i) + S_i] \tag{2}$$

Note that Equation (2) is also valid for the timely-token. That is, the equation was obtained without being specific about the token protocol. Therefore, we may compare the timely-token against FDDI simply by comparing their I functions.

To distinguish between function I in FDDI and in the timely-token, the latter is denoted by I_F and the latter is denoted by I_t.

Theorem 5 *For the timely-token protocol,*

$$I_t(n) = n \cdot TTRT - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot A^* \tag{3}$$

Proof: We need to construct a scenario where the interval of time from the longest scenario before a station can transmit $n \cdot S_i$ synchronous units. From

Theorem 2, the synchronous transmissions do not affect asynchronous transmissions. This implies that in our scenario all stations must have unlimited amounts of synchronous data to transmit.

For ease of presentation we consider station zero. The scenario can be reproduced for any other station.

Consider round $R^{0,m}$. Recall that each station transmits synchronous bandwidth before transmitting asynchronous bandwidth. Thus, the longest interval before station zero transmits $n \cdot S_0$ units of synchronous bandwidth begins immediately after station zero transmits its synchronous bandwidth in round $R^{0,m}$, and ends after transmitting S_0 synchronous units at the beginning of round $R^{0,m+n}$. Thus, the interval of time is

$$[start(R^{0,m}) + S_0,\ start(R^{0,m+n}) + S_0]$$

where $start(R^{0,m})$ denotes the starting time of round $R^{0,m}$. Note that the interval is of the same length as

$$[start(R^{0,m}),\ start(R^{0,m+n})]$$

Thus, we consider the duration of rounds $R^{0,m}$ up to and including $R^{0,m+n-1}$.

During the above rounds, $n \cdot \sum_{h=1}^{N} S_h$ units of synchronous data are transmitted. We next focus on the asynchronous data.

Consider the duration of the asynchronous transmissions of every station in the above n rounds. That is, consider the sequence

$$a_0^{0,m}, a_1^{0,m}, \ldots, a_{N-1}^{0,m}, \ldots, a_0^{0,m+n}, a_1^{0,m+n}, \ldots, a_{N-1}^{0,m+n}$$

From Theorem 2, any value in this sequence plus the previous N values add to at most A^*. That is, any subsequence of length $N+1$ can add to at most A^*. We therefore break this sequence into segments with $N+1$ elements. If less than $N+1$ elements remain these may add to at most A^*. Hence, the total asynchronous transmission can add up to at most

$$\left\lceil \frac{n \cdot N}{N+1} \right\rceil \cdot A^*$$

The above is equal to

$$n \cdot A^* - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot A^*$$

Combining the synchronous and asynchronous transmission, this adds to

$$n \cdot \sum_{h=1}^{N} S_h + n \cdot A^* - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot A^*$$

To obtain the size of the desired interval, we need to add $n \cdot \tau$ since each of the n rounds has an overhead of τ. This, plus the definition of $TTRT$, gives us the final result.

$$I_t(n) = n \cdot TTRT - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot A^*$$

■

Notice that we developed our earlier SA scheme under the assumption that the token will visit each station during every $TTRT$ seconds, i.e., under the assumption that $I_t(n) \leq n \cdot TTRT$. With the tighter $I_t(n)$ bound proven above we can provide a better (although more complex) SA scheme, as discussed in the appendix.

The most accurate value for function I_F in FDDI has been derived by Zhang and Burns [5], and is as follows.

$$I_F(n) = n \cdot TTRT + \sum_{h=1}^{N} S_h + \tau - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot \left(TTRT - \sum_{h=1}^{N} S_h - \tau \right)$$

From our definition of A^*,

$$I_F(n) = n \cdot TTRT + \sum_{h=1}^{N} S_h + \tau - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot A^* \tag{4}$$

Note that a smaller value of I is desirable, since this yields a larger value for X_i, and allows the protocol to schedule a larger number of flows. If we compare the expressions for I in Equations (3) and (4), we find that, for any n, $I_t(n) < I_F(n)$. In consequence, for the same set of S_i values, the timely-token yields larger values for each X_i. Thus, any set of synchronous flows whose deadlines are satisfied by FDDI are also satisfied by the timely-token, in particular, by using the same S_i values as FDDI. The converse is not true, however, as shown by an example when we discuss FDDI-M below.

Another advantage of the timely-token is as follows. In [5], it is shown that the following is a requirement for any synchronous stream set in FDDI:

for every i, $D_i \geq TTRT$. However, as we have seen earlier, the timely-token is able to support synchronous stream sets where $D_i < TTRT$ for some i.

We next focus our comparison on FDDI-M. To our knowledge, there is no SA scheme for FDDI-M. However, as mentioned above, FDDI-M may starve asynchronous traffic, while the timely-token ensures A^* time units are available for asynchronous traffic per token rotation.

In [6], FDDI and FDDI-M are compared in their ability to support a set of homogeneous synchronous streams, i.e., all streams have the same D, C, and P values. In FDDI, $TTRT \leq D/2$ is required in order to meet the message deadlines. The largest number of streams are supported when $TTRT = D/2$ and $S = C$. Therefore, a maximum of $\frac{D/2}{S}$, i.e., $\frac{D}{2 \cdot C}$ streams can be supported concurrently. On the other hand, for FDDI-M, we may set $TTRT = D$, since the token is never late. Thus, a maximum of $\frac{D}{C}$ streams may be supported, i.e., twice those of $FDDI$. Note that for the timely-token, we may also set $TTRT = D$, and support the same number of synchronous streams as FDDI-M, and thus support twice the number of streams as FDDI.

7 Token Implementation

For an efficient implementation, a station must be able to quickly determine, with only a few bits delay, if the incoming frame is a token or not. Furthermore, it must quickly decide whether it should seize the token. One bit in the frame can be used to distinguish between data frames and token frames. Thus, the remaining issue is to quickly determine if the station should seize the token. A station with $S_i > 0$ should always seize the token, since it is allowed to transmit up to S_i time units of synchronous packets. Even if the station has no synchronous packets to transmit, seizing the token and then releasing it (which takes less than S_i time units) will not interfere with the guarantees of other stations.

Assume now that station i has $S_i = 0$. If the station has asynchronous packets to send, it must quickly determine if it should seize the token. From the outline of the timely-token, a station can transmit asynchronous packets for a total of $A_i = TTRT - TRT_i - u$ time units. Thus, it must look through all bits of u in the token and its timer TRT_i to decide if it should seize the token. A way to do this with a delay of only a couple of bits is as follows. The station keeps in a register the value of $TTRT - TRT_i$. The register has an initial value of $TTRT$ and counts down only. The station compares

the register with u (bit by bit) as the bits are received. The station cannot transmit asynchronous packets if the register value is less than u. On the other hand, if the register is greater than u, then the station aborts forwarding the token (and hence the next station cannot seize the token since it receives a partial token) and seizes the token. Note that deciding if u is less than or greater than the register can be done as the bits of the token are stored and forwarded, provided the bits of u are received in order of their significance. Hence, the token can either be seized or allowed to go by with a delay of only a few bits per station.

8 Conclusion

This paper proposes a new protocol: the timely-token protocol. This protocol adds a parameter in the token that allows a station to determine whether the early arrival of a token is due to unused synchronous bandwidth or unused asynchronous bandwidth. By doing this, it prevents the excessive use of asynchronous transmission, and thus it avoids the token lateness problem that exists in FDDI. At the same time, it guarantees that in each token rotation there is always A^* time units available for the transmission of asynchronous traffic. This makes up for the weakness existing in FDDI-M, where asynchronous traffic may be starved.

A simple Synchronous Allocation scheme is given for the timely-token. It guarantees that if a set of synchronous streams satisfies the protocol constraints, then synchronous bandwidth is allocated to each station such that no deadline is violated. The timely-token was compared against FDDI and FDDI-M. Finally, a method to efficiently implement the seizing of the token was discussed.

A SA scheme $\mathcal{S}$ is said to be optimal [2] for a token-protocol iff, for every other SA scheme $\mathcal{S}'$, and for every set of synchronous streams, if $\mathcal{S}$ meets the deadlines of the stream set, then $\mathcal{S}'$ also meets the deadlines of the stream set.

Finding an optimal SA scheme for FDDI remains a challenge, since none of the SA schemes proposed thus far are optimal [12]. Proving that an SA scheme is optimal is challenging to the point that a couple of early SA schemes were erroneously believed to be optimal.

In the appendix, we present an enhanced SA scheme for the timely-token that satisfies a broader set of synchronous streams than the SA scheme in

Section 5. This SA is added mostly for completeness, since the range of values over which it improves the scheme of Section 5 is not practical. We conjecture that our enhanced SA scheme is optimal for the timely-token. However, given the complexity of proving the optimality of a SA scheme, we defer evaluating the optimality of our enhanced SA scheme to future work.

References

[1] R. Grow, "A timed token protocol for local area networks", *Proc. Electro'82, Token Access Protocols*, Paper 17/3, May 1982.

[2] B. Chen, G. Agrawal, and W. Zhao, "Optimal synchronous capacity allocation for hard real-time communications with the timed token protocol", *Proc. IEEE RTSS'92*, pp. 198-207, Dec. 1992.

[3] J. Cobb, M. Lin, "The On-Time timed-token protocol", *Proc. IEEE GLOBECOM*, 2002.

[4] C.C. Han and K. G. Shin, "A polynomial-time optimal synchronous bandwidth allocation scheme for the timed-token MAC protocol", *Proc. IEEE Infocom'95*, pp. 875-882, Apr. 1995.

[5] S. Zhang and A. Burns, "An optimal synchronous bandwidth allocation scheme for guaranteeing synchronous message deadlines with the timed-token MAC protocol", *IEEE/ACM Trans. Networking*, Vol.3, pp. 729-741, Dec. 1995.

[6] K. G. Shin and Q. Zheng, "FDDI-M: A scheme to double FDDI's ability of supporting synchronous traffic", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 11, Nov. 1995.

[7] A. G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed token medium access control protocol", *IEEE Trans. on Comput.*, Vol. 43, No. 3, pp. 327-339, Mar. 1994.

[8] S. Zhang, A. Burns and A. Wellings, "An efficient and practical local synchronous bandwidth allocation scheme for the timed-token mac protocol", *Proc. IEEE Infocom*, pp 920-927, 1996

[9] S. Zhang and E.S. Lee, "Efficient global allocation of synchronous bandwidths for hard real-time communication with the timed token MAC protocol", *Proc. of the Real-Time Computing Systems and Applications Conference*, pp. 120 -127, 1999.

[10] D. Chen, V.C.S. Lee and E. Chan, "On the ability of the FDDI-M protocol to support real-time traffic", *Proc. of the Real-Time Computing Systems and Applications*, pp. 51 -57, 1998.

[11] D. Chen, E. Chan and C.H. Lee, "Timing properties of the FDDI-M medium access protocol for a class of synchronous bandwidth allocation schemes", *Proc. of the Int'l. Conference on Computer Communications and Networks*, pp. 825-832, 1998.

[12] S. Zhang and E.S. Lee, "The non-optimality of synchronous bandwidth allocation schemes for the timed token Protocol", *IEEE Communications Letters*, Vol.4, No. 3, March 2000.

Appendix

In this appendix, we present a SA scheme that is superior to the one presented in Section 5. In Section 5, it is assumed that consecutive visits of the token to the station are separated by $TTRT$ seconds. That is, it is assumed that

$$I(n) = n \cdot TTRT$$

However, as seen in Section 6, $I(n)$ actually has a smaller value when $n \geq N + 1$, namely,

$$I(n) = n \cdot TTRT - \left\lfloor \frac{n}{N+1} \right\rfloor \cdot A^* \tag{5}$$

We take advantage of this lower value of $I(n)$ for our Enhanced SA Scheme (ESAS).

As mentioned in Section 6, the value of X_i may be derived from $I(n_i)$ as follows.

$$X_i = (n_i - 1) \cdot S_i + \max[0, D_i - I(n_i) + S_i] \tag{6}$$

where n_i is an integer such that

$$I(n_i - 1) \leq D_i < I(n_i) \tag{7}$$

In order to meet the deadline of stream i, we must have

$$X_i \geq C_i \tag{8}$$

Note that the value of S_i needed to satisfy this depends on the value of n_i and $I(n_i)$. In turn, n_i and $I(n_i)$ depend on S_i, and a circular argument is created. Therefore, we cannot simply derive the value of S_i from an equation. Instead, we adopt an iteration method, similar to the one adopted in [7] and [5].

Our general strategy is therefore as follows.

- First, we choose $S_i = 0$ for all i.
- Then, we derive n_i and $I(n_i)$ for each i.
- Next, assuming that n_i and $I(n_i)$ remain constant, we compute for each i a new value of S_i that is the minimum value needed to ensure $X_i \geq C_i$.
- Since each S_i has increased, we obtain new values of n_i and $I(n_i)$.
- These new values in turn may require each value of S_i to be increased, and thus new values of n_i and $I(n_i)$ are needed. Thus, the iteration continues.

The above iteration should stop when one of the following hold.

(i) For each i, the value of S_i need not increase to satisfy $X_i \geq C_i$.

(ii) The protocol constraint of Definition 1 is violated.

We next describe the approach in more detail. First, assuming n_i and $I(n_i)$ remain constant, the minimum value needed for S_i follows from Equation 6, and is as follows.

$$\begin{aligned} S_{min}(n_i, I(n_i), C_i) &= \begin{cases} \frac{C_i}{n_i - 1} & \textbf{if } \delta_i \cdot (n_i - 1) \geq C_i \\ \delta_i \cdot (n_i - 1) + \frac{C_i - \delta_i \cdot (n_i - 1)}{n_i} & \textbf{otherwise} \end{cases} \\ \delta_i &= I(n_i) - D_i \end{aligned}$$

Above, δ_i is the amount of time that the token is unavailable for synchronous bandwidth during its n_ith arrival to station i.

We next proceed to give the SA algorithm in more detail.

Enhanced SA Scheme:

1. for each i, let $S_i := 0$.
2. for each i, obtain n_i and $I(n_i)$ according to Equations 5 and 7.
3. if for each i, $S_i \geq S_{min}(n_i, I(n_i), C_i)$, then stop, a solution has been found.
4. for each i, let $S_i := S_{min}(n_i, I(n_i), C_i)$.
5. If the protocol constraint is violated, stop, no solution was found.
6. Go to step 2.

Next we provide a theorem to justify the mechanism to compute S_i above.

Theorem 6 *Consider a synchronous allocation to every station such that the deadlines of all synchronous streams are satisfied. Let S'_i be the synchronous allocation to station i. Assume the following hold for this synchronous allocation.*

$$I'(x) = x \cdot TTRT - \left\lfloor \frac{x}{N+1} \right\rfloor \cdot A^{*\prime} \text{ for all } x$$
$$I'(n'_i - 1) \leq D_i < I'(n'_i) \text{ for all } i$$
$$S_{min}(n'_i, I'(n'_i), C_i) \leq S'_i \text{ for all } i$$

Assume that after step number 2 in the ESAS, for all i,

$$S_i \leq S'_i$$

Then this continues to hold after step four.

Proof: First, note that if $\sum S_i$ increases, then for any x, $I(x)$ must remain equal or increase. This is because as $\sum S_i$ increases, A^* decreases, and $I(x)$ remains equal or increases. Therefore, since we are given that $\sum S_i \leq \sum S'_i$, then for all x,

$$I(x) \leq I'(x)$$

We next compare $S_{min}(n_i, I(n_i), C_i)$ and $S_{min}(n'_i, I(n'_i), C_i)$. Note that since $I(x) \leq I'(x)$ for all x, it must be the case that $n_i \leq n'_i$.

Assume first that $n_i = n'$ and $I(n_i) \leq I'(n'_i)$. In this case, from the definition of S_{min}, we have

$$S_{min}(n_i, I(n_i), C_i) \leq S_{min}(n'_i, I'(n'_i), C_i)$$

Assume on the other hand that $n_i < n'_i$. In this case, again, from the definition of S_{min}, we have

$$S_{min}(n_i, I(n_i), C_i) \leq S_{min}(n'_i, I'(n'_i), C_i)$$

After step four in the algorithm, $S_i = S_{min}(n_i, I(n_i), C_i)$, and we are given that $S_{min}(n'_i, I'(n'_i), C_i) \leq S'_i$ for all i. Thus, after step four, for all i,

$$S_i \leq S'_i$$

■

Corollary 1 *For a given set of synchronous flows and $TTRT$ value, if there exists a synchronous bandwidth allocation that satisfies Equations (5), (6), and (8), then the ESAS also obtains a synchronous bandwidth allocation that satisfies these equations.*

Proof: In ESAS, the value of S_i never increases beyond the S_i value of the solution that satisfies Equations (5), (6), and (8). Thus, since the minimum S_i value always increases in ESAS, ESAS will converge to a solution. ■

Note that the corollary restricts itself to those solutions satisfying Equations (5), (6), and (8). This is because we do not claim that the ESAS above is optimal for the timely-token. Proving that a SA allocation scheme is optimal has been known to be challenging to the extent that for FDDI no optimal SA scheme is known, and earlier SA schemes were erroneously considered to be optimal.

The ESAS provide in this appendix is provided primarily for completeness. The scheme in Section 5 will suffice for most cases. ESAS is superior only when $n_i \geq N + 1$, and since N is likely to be large, this would apply only for very large values of D_i.