A Polynomial-Time Optimal Synchronous Bandwidth Allocation Scheme for the Timed-Token MAC Protocol

Ching-Chih Han and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2122
{cchan,kgshin}@eecs.umich.edu

Abstract

Numerous methods have been proposed to integrate real-time and non-real-time services of the timed-token medium access control (MAC) protocol. One of the key issues in tailoring the timed-token MAC protocol for real-time applications is the synchronous bandwidth allocation (SBA) problem whose objective is to meet both the protocol and deadline constraints.

Several non-optimal local SBA schemes and an optimal global scheme have been proposed [1–3]. Local SBA schemes use only information available locally to each node, and are thus preferred to global schemes because of their lower network-management overhead. Unfortunately, it has been formally proved in [4] that there does not exist any optimal local SBA scheme. Chen et al. [2] proposed the only-known optimal global SBA scheme which is based on an iterative approach. However, their algorithm may not terminate theoretically. In this paper, we present an optimal global SBA scheme of polynomial-time worst-case complexity.

1 Introduction

The problem of guaranteeing the timely delivery of messages has been studied by numerous researchers, especially in the context of voice/video data transmission over a data network, and in the context of communications in embedded real-time systems. Among all the methods designed to integrate real-time and non-real-time applications, the timed-token MAC protocol has attracted considerable attention because of its bounded access time. The timed-token protocol groups messages into two classes: synchronous and asynchronous. Synchronous messages arrive at regular intervals and are usually associated with delivery deadlines. Asynchronous messages have no such time constraints. At network initialization, a protocol parameter called *Target Token Rotation Time* (TTRT) is negotiated among the nodes to specify the expected token rotation time. Each node i is assigned a portion H_i of TTRT as its *synchronous bandwidth.* The assignment of H_i is subject to the *protocol constraint* that the total bandwidth allocated for synchronous traffic over all nodes/stations should not exceed TTRT (minus various protocol-dependent overheads). Whenever a node receives the token, it transmits its synchronous messages, if any, up to H_i units of time. The node can transmit its asynchronous messages only if the time interval between the previous token arrival and the current token arrival is less than TTRT.

The work reported in this paper was supported in part by the ONR under Grants N00014-92-J-1080 and N00014-94-1-0229, and by the NSF under Grant MIP-9203895.

Many researchers studied the access time bounds and other timing properties of the timed-token protocol. In particular, Johnson *et al.* [5,6] proved that the *average* token cycle time is bounded by TTRT, and the *maximum* token cycle time is bounded by 2 $\times$ TTRT. Agrawal *et al.* [1,2] extended Johnson's result and proved that the time elapsed between k consecutive token's visits to a node is bounded by k $\times$ TTRT. They also formulated a *synchronous bandwidth allocation* (SBA) problem and attempted to calculate the synchronous bandwidth H_i that should be allocated to node i, for all i, to meet the protocol constraint and transmit all synchronous messages before their deadlines. Succinctly, H_i should be assigned so that the minimum time available for node i to transmit a synchronous message after its arrival but before its delivery deadline is greater than or equal to the worst-case message transmission time. This timing constraint in calculating H_i's is called the *deadline constraint*.

As discussed in [1], SBA schemes can be classified as local or global. A local SBA scheme uses only information available locally to a node, while a global scheme uses the parameters of all nodes' synchronous message streams in computing H_i's. The extra information on other nodes used by a global scheme may help it find better values of H_i's. However, any change in a node's message stream parameters may require the global scheme to adjust the synchronous bandwidths of *all* nodes, since all nodes use these parameters in calculating their synchronous bandwidths. By contrast, in a local scheme, if the message stream parameters of node i change, only H_i needs to be re-calculated. Thus, local schemes are preferable to global schemes from network management's perspective.

As the global SBA schemes use global information to allocate synchronous bandwidths, they are naturally expected to achieve a better performance. To our best knowledge, there are only one optimal global SBA scheme [2] and several non-optimal local SBA schemes [1,3] reported in the open literature. By an "optimal" SBA scheme, we mean an SBA scheme that finds a feasible set of H_i's subject to the protocol and deadline constraints whenever such a set exists. Unfortunately, it has been formally proved in [4] that there does not exist any optimal local SBA scheme. The only currently-known optimal global SBA scheme [2], which uses an iterative approach to find the minimum synchronous bandwidth allocations, theoretically, may not terminate. One important remaining issue is to determine if there exists any polynomial-time optimal global SBA scheme. In this paper, we propose an optimal SBA scheme of polynomial-time worst-case complexity.

The rest of the paper is organized as follows. In Section 2, we discuss the synchronous message model used for real-time applications and give a brief overview of the timed-token protocol. In Section 3, we present several timing properties for the timed-token protocol and discuss the timing requirements imposed by the messages with delivery deadlines on the protocol. In Section 4, we formulate the SBA problem and describe our polynomial-time optimal SBA scheme. We conclude the paper with Section 5.

2 Message model and MAC protocol

In this section, we first discuss the synchronous message model suitable for real-time applications. To make the paper self-contained, we also briefly review the timed-token MAC protocol used in FDDI networks and some of its timing properties. A more detailed description of the timed-token protocol and FDDI token rings can be found in [7,8].

2.1 Message model

Let n be the number of nodes in the system. Without loss of generality, we assume that there is one synchronous message stream at each node. The message stream at node i can be described by a triple (P_i, C_i, D_i), where

- P_i is the minimum inter-arrival period for the message stream at node i, i.e., if the j-th message arrives at node i at time t, then the $(j+1)$-th message will arrive at time $t + P_i$ or later, for all $j \geq 1$,
- C_i is the maximum message transmission time at node i, i.e., C_i is the time needed to transmit a maximum-size message, and
- D_i is the *relative* deadline for the message stream at node i, i.e., if a message arrives at time t, then it must be transmitted by time $t + D_i$.

The objective of an SBA scheme is to properly set the parameters of the MAC protocol so as to guarantee the delivery of each message in node i's synchronous message stream within a time period $\leq D_i$ after its arrival, as long as the message inter-arrival time is $\geq P_i$ and the message transmission time is $\leq C_i$.

2.2 MAC protocol

The key idea of the MAC protocol is to control the token rotation time. A protocol parameter called the *target token rotation time* (TTRT) is determined upon network initialization, and specifies the expected token rotation time. The TTRT is chosen to be sufficiently small so that responsiveness requirements at every node may be met.

Each node i is assigned a portion H_i of TTRT, known as its *synchronous bandwidth*, which is the maximum time a node is permitted to transmit synchronous messages every time it receives the token. The token is then forced by the protocol to circulate with sufficient speed so that all nodes receive their allocated fractions of bandwidth for transmitting synchronous traffic. This is achieved by transmitting asynchronous messages only when the token rotates sufficiently fast so that it returns to a node within the TTRT, i.e., it arrives *early*. Specifically, each node has two timers and one counter:

- The *token rotation timer* (TRT) records the time elapsed since the last token's visit (if the TRT has not yet expired). It is initialized to TTRT, and counts down (i) until it expires (i.e., reaches zero) or (ii) until the token is received and the time elapsed since its last visit is less than TTRT. In either case, TRT is reset to TTRT and continues to count down from the newly-set value.
- The *token holding timer* (THT) records the amount of time by which the token has arrived early, i.e., the amount of time which can be used to transmit asynchronous messages. It is initialized to zero, is set to the value of TRT when the token arrives early, and counts down during the transmission of asynchronous messages.
- The *late counter* (LC) records the number of times its TRT has expired since the last token's visit to the node. It is initialized to zero, is incremented

whenever TRT expires, and is reset to zero each time the node receives the token.

After the TTRT value is negotiated among the nodes during network initialization, each node i initializes its timers and counter as follows: TRT $\leftarrow$ TTRT; THT $\leftarrow$ 0; LC $\leftarrow$ 0. TRT is enabled during all ring operations and always counts down until one of the following three events occurs:

E1. TRT reaches zero: The following steps are taken: (i) TRT $\leftarrow$ TTRT, and TRT continues to count down, and (ii) LC $\leftarrow$ LC +1.

E2. The token arrives early: This is identified by LC = 0 at the time of token arrival. In this case, the following steps are taken: (i) THT $\leftarrow$ TRT, and THT counts down only during the transmission of asynchronous messages, (ii) TRT $\leftarrow$ TTRT, and TRT continues to count down, (iii) asynchronous messages, if any, are transmitted until THT expires or until all asynchronous messages are transmitted, whichever occurs first, and (iv) synchronous messages are transmitted up to H_i units of time or until all synchronous messages are transmitted, whichever occurs first.[1]

E3. The token arrives late: That is, LC $\neq 0$ at the time of token arrival. In this case, the following steps are taken: (i) LC $\leftarrow$ 0, (ii) TRT is not reset, and continues to count down, and (iii) only synchronous messages can be transmitted up to H_i units of time, and no asynchronous messages can be transmitted.

3 Protocol timing properties and real-time requirements

In this section, we discuss several interesting timing properties associated with the MAC protocol and the timing requirements imposed on the MAC protocol by the messages with delivery deadlines.

We define following notation:

- T: the TTRT of an FDDI network.
- τ: the portion of the synchronous bandwidth unavailable for transmitting synchronous messages. τ includes medium propagation delay, token transmission time, station latency, token capture delay, and various protocol-dependent overheads [6].
- $\vec{H}$: vector $(H_1, H_2, \ldots, H_n)$, where H_i is the synchronous bandwidth allocated to node i.
- X_i: the minimum time available for node i to transmit synchronous messages in an interval $(t, t+D_i]$.
- f_g, f_l: the functions which represent the global and local synchronous bandwidth allocation schemes, respectively. That is, a global allocation scheme can be represented as $\vec{H} = f_g(\vec{C}, \vec{D}, \vec{P}, T, \tau)$, where $\vec{C} = (C_1, C_2, \ldots, C_n)$, $\vec{P} = (P_1, P_2, \ldots, P_n)$, and $\vec{D} = (D_1, D_2, \ldots, D_n)$. A local allocation scheme can be represented as $H_i = f_l(C_i, D_i, P_i, T, \tau)$, for $i = 1, 2, \ldots, n$.

[1] In the MAC protocol, it is not specified which of synchronous or asynchronous traffic will be transmitted first.

Note that a node i can transmit its synchronous messages only up to its assigned synchronous bandwidth H_i, and can transmit its asynchronous messages only when the token arrives early and only up to the amount of time by which the token arrived early.

Timing properties of MAC protocol. The protocol constraint on the allocation of synchronous bandwidth states that the total bandwidth allocated to synchronous traffic among all nodes in a timed-token ring should not exceed the available portion $T-\tau$ of TTRT, i.e., $\sum_{i=1}^{n} H_i \leq T - \tau$. Violation of the protocol constraint will make the ring unstable and oscillate between "claiming" and "operational."

Under the timed-token protocol constraint, the following well-known result for the MAC protocol is formally proved in [5,6].

Theorem 1: (Johnson and Sevcik's Theorem) For the timed-token protocol, the worst-case token rotation time — the time elapsed between any two consecutive token's visits to a node — is bounded by $T + \sum_{j=1}^{n} H_j + \tau \leq 2 \cdot T$. □

A more general result has also been obtained by Agrawal *et al.* [1,9]:

Corollary 1: For the timed-token protocol, the time elapsed between any $c+1$ consecutive token's visits to a node is bounded by $c \cdot T + \sum_{j=1}^{n} H_j + \tau \leq (c+1) \cdot T$. □

A proof of the above theorem and an example showing that the bound is tight can be found in [9]. A simpler proof can also be found in [4], in which a more general result is also derived.

The deadline constraint. Every synchronous message for real-time applications must be transmitted to meet its delivery deadline. That is, the minimum amount of time, X_i, available for node i to transmit its synchronous messages in an interval $(t, t+D_i]$ should be no less than the required maximum message transmission time. Using Corollary 1, Agrawal *et al.* [1,2] derived a lower bound for the time available for a node to transmit its synchronous messages within a given time period D_i as follows.

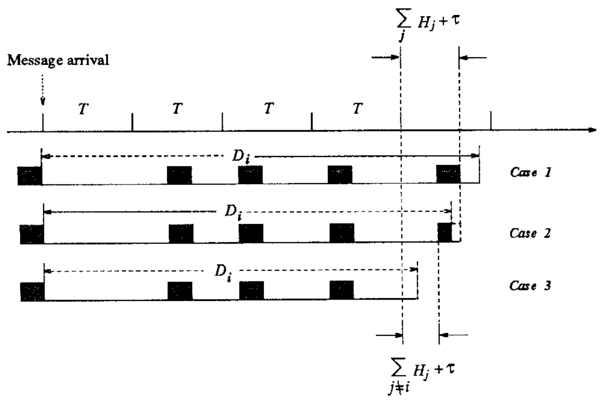

Figure 1: Worst-case token visit scenarios.

Theorem 2: Assume that at time t, a synchronous message with deadline D_i arrives at node i ($1 \le i \le n$). Then, the minimum amount of time, X_i, available for node i to transmit this synchronous message before its deadline is given by

$$X_i(\vec{H}) = (q_i-1)\cdot H_i + \max(0, \min(r_i - (\sum_{\substack{j=1,\dots,n\\ j\neq i}} H_j + \tau), H_i)). \tag{3.1}$$

where $q_i = \lfloor D_i/T \rfloor$ and $r_i = D_i - q_i \cdot T$. □

Note that Eq. (3.1) can be re-written as

$$X_i(\vec{H}) = \begin{cases} (1)\ q_i \cdot H_i, \text{ if } r_i \ge \sum_j H_j + \tau, \\ (2)\ (q_i-1)\cdot H_i + r_i - \sum_{j\neq i} H_j - \tau, \\ \quad \text{if } \sum_{j\neq i} H_j + \tau < r_i < \sum_j H_j + \tau, \\ (3)\ (q_i-1)\cdot H_i, \text{ if } r_i \le \sum_{j\neq i} H_j + \tau. \end{cases} \tag{3.2}$$

Also, note that the time available for a node to transmit a synchronous message before its deadline becomes *minimal* when the message arrives *right after* the token's departure. Figure 1 depicts the scenarios where Case 1, 2, or 3 may arise. Since the time elapsed between any $c+1$ consecutive token's visits is bounded by $c \cdot T + \sum_{j=1}^{n} H_j + \tau$,

Case 1. if $r_i \ge \sum_j H_j + \tau$, then $D_i \ge q_i \cdot T + \sum_j H_j + \tau$, and D_i can "accommodate" the q_i-th token's visit since the message arrival.

Case 2. if $\sum_{j\neq i} H_j + \tau < r_i < \sum_j H_j + \tau$, then D_i can accommodate the first $(q_i - 1)$ token's visits and part of the q_i-th token's visit.

Case 3. if $r_i \le \sum_{j\neq i} H_j + \tau$, then, in the worst case, $D_i = q_i \cdot T + r_i \le q_i \cdot T + \sum_{j\neq i} H_j + \tau$, and D_i cannot accommodate the q_i-th token's visit since the message arrival. However, $D_i \ge q_i \cdot T \ge (q_i - 1)\cdot T + \sum_j H_j + \tau$, and D_i can accommodate the first $(q_i - 1)$ token's visits since the message arrival.

For $D_i \le P_i$, the timing requirements of synchronous messages impose the deadline constraint that $X_i(\vec{H}) \ge C_i$, for $i = 1, 2, \dots, n$.

4 SBA problem and solution algorithm

In the following discussion, we assume that $D_i \le P_i$, for all i. In such a case, f_g (f_l) is independent of $\vec{P}$ (P_i), and hence $\vec{P}$ (P_i) can be dropped from the argument list of f_g (f_l).

We first give a formal mathematical formulation of the SBA problem and then describe the proposed algorithm.

Problem 1: (The SBA Problem) Given the number of nodes (or synchronous message streams), n, the maximum message transmission time vector, $\vec{C} = (C_1, C_2, \dots, C_n)$, the deadline vector, $\vec{D} = (D_1, D_2, \dots, D_n)$, and the negotiated TTRT, T, the objective of a global SBA scheme is to find an algorithm that realizes the function f_g:

$$\vec{H} = (H_1, H_2, \dots, H_n) = f_g(\vec{C}, \vec{D}, T, \tau), \tag{4.1}$$

and the objective of a local SBA scheme is to find an algorithm that realizes the function f_l:

$$H_i = f_l(C_i, D_i, T, \tau), \text{ for } i = 1, 2, \dots, n, \tag{4.2}$$

subject to

$$\textbf{protocol constraint:} \quad \sum_{i=1}^{n} H_i \le T - \tau, \tag{4.3}$$

$$\textbf{deadline constraint:} \quad X_i(\vec{H}) \ge C_i, \tag{4.4}$$

where $X_i(\vec{H})$ is defined in Eq. (3.1). □

A *feasible* solution for the above SBA problem is a vector $\vec{H}$ that satisfies both the protocol and deadline constraints. An *optimal* global (local) SBA scheme is the one that realizes the function f_g (f_l) whenever such a solution exists.

As mentioned in Section 1, whether or not there exists any polynomial-time optimal global SBA scheme remains unknown since the only previously-known optimal global SBA scheme (proposed by Chen *et al.* [2]), theoretically, may not terminate. For convenience of reference, we include their algorithm in Figure 2 and some of the results they derived below. Note that their algorithm assumes that $q_i \ge 2$, for all i. For convenience of discussion, we also make the same assumption. (Interested readers are referred to [10] for the details of how to relax this assumption.)

Let Π be the set of solutions for Eq. (4.4), i.e.,

$$\Pi = \{\vec{H} \mid X_i(\vec{H}) \ge C_i, \text{ for all } i\}. \tag{4.5}$$

Also, for two given vectors $\vec{H}'$ and $\vec{H}''$, we define

```
Procedure Min_H;
1. For $i = 1$ to $n$ do $H_i := \frac{C_i}{q_i}$;
2. Repeat
3.     For $i = 1$ to $n$ do calculate $X_i$ as defined in Eq. (3.1);
4.     For $i = 1$ to $n$ do {
5.         $b_i := C_i - X_i$;
6.         If $b_i > 0$ then $H_i := H_i + \frac{b_i}{q_i - 1}$ };
7. Until none of $b_i$'s are larger than zero;

Allocation Scheme MCA
1. Call procedure Min_H to obtain $\vec{H}^*$;
2. If $\sum_{i=1}^{n} H_i^* \le T - \tau$ then return(success, $\vec{H}^*$)
3. else return(fail, nil)
```

Figure 2: The optimal SBA scheme proposed in [2].

- $\vec{H}' = \vec{H}''$ if and only if $H_i' = H_i''$, for all i,
- $\vec{H}' \le \vec{H}''$ if and only if $H_i' \le H_i''$, for all i, and
- $\vec{H}' < \vec{H}''$ if and only if $\vec{H}' \le \vec{H}''$ and $\vec{H}' \ne \vec{H}''$.

Chen *et al.* proved the following theorem (Theorem 6.1 in [2]).

Theorem 3: The set Π has the following properties:

- Π is not empty, i.e., Eq. (4.4) is solvable,
- there is a minimal element $\vec{H}^*$ in Π, i.e., for any $\vec{H} \in \Pi$, $\vec{H}^* \le \vec{H}$, for all i, and
- $\frac{C_i}{q_i} \le H_i^* \le \frac{C_i}{q_i - 1}$, for all i. □

Their optimal SBA scheme, called MCA (Figure 2), uses a procedure, called Min_H, to find the minimal element $\vec{H}^*$ in Π and then checks if $\vec{H}^*$ satisfies the protocol constraint. If Procedure Min_H can always find $\vec{H}^*$, MCA is an optimal SBA scheme since $\vec{H}^*$ minimizes $\sum_{i=1}^{n} H_i$ in the protocol constraint Eq. (4.3) among all $\vec{H} \in \Pi$. However, although they proved that the value of $\vec{H}$ calculated in the Repeat–Until loop of Procedure Min_H converges to $\vec{H}^*$, Procedure Min_H is *not* guaranteed to terminate. For example, let $T = 30$, $\tau = 0$, $q_i = 6$, $r_i = 24$, and $C_i = 30$, for $i = 1, 2, \ldots, 5$. Let $b_i^{(k)}$ denote the value of b_i at the k-th iteration of the loop in their algorithm, then $b_i^{(k)} = (\frac{4}{5})^{k-1} > 0$, for all i. Moreover, even if we use a traditional engineering approach to stop the algorithm at a certain point, such as forcing the algorithm to terminate when all b_i's are smaller than a certain threshold, the values of H_i's thus found are still unusable since some of them are not large enough to satisfy the deadline constraint ($b_i > 0$). To remove this deficiency, we propose another algorithm for finding the minimal element $\vec{H}^* \in \Pi$, which is guaranteed not only to terminate but also to terminate in polynomial time. Before describing the algorithm, we first study the deadline constraint in more detail.

As discussed in Section 3, there are three possible regions for the values of $X_i(\vec{H})$ (Figure 1). For convenience of discussion, we say that $X_i(\vec{H})$ (or, simply, X_i or H_i, if $\vec{H}$ is unambiguous in the context) is in Region I, II, or III, if $r_i \ge \sum_j H_j + \tau$, $\sum_{j \ne i} H_j + \tau < r_i < \sum_j H_j + \tau$, or $r_i \le \sum_{j \ne i} H_j + \tau$, respectively. It is easy to see that $X_i(\vec{H}^*) = C_i$, for all i, since if $X_i(\vec{H}^*) > C_i$ we can find another vector $\vec{H}'$ with $H_i' = H_i^* - \epsilon$, $H_j' = H_j^*$, for $j \ne i$, and ϵ a very small positive number, which also satisfies the deadline constraint, and hence contradicts that $\vec{H}^*$ is the minimal element in Π (this property will be used later). Therefore, we can conclude that if H_i^* is in Region I, II, or III, then H_i^* equals $\frac{C_i}{q_i}$, $\frac{C_i - (r_i - \sum_{j \ne i} H_j^* - \tau)}{q_i - 1}$, or $\frac{C_i}{q_i - 1}$, respectively. Moreover, if we know, for each i, which region H_i^* falls in, then the values of H_i^*'s can be easily found by solving the following system of n linear equations with n variables:

$$x_i = \begin{cases} \frac{C_i}{q_i}, & \text{if } H_i^* \text{ is in Region I,} \\ \frac{C_i - (r_i - \sum_{j \ne i} x_j - \tau)}{q_i - 1}, & \text{if } H_i^* \text{ is in Region II,} \\ \frac{C_i}{q_i - 1}, & \text{if } H_i^* \text{ is in Region III,} \end{cases} \tag{4.6}$$

for $i = 1, 2, \ldots, n$. However, since each of H_i^*'s may fall in one of the three regions, if we try to find the vector $\vec{H}^*$ by guessing the region that each H_i^* falls in, in the worst case, we need to solve the system of linear equations (Eq. (4.6)) 3^n times (and check if they indeed fall in the regions we guessed). This means that the SBA problem can be solved by an algorithm that is guaranteed to terminate in exponential time. Now, there is still one important theoretical question left: Does any polynomial-time optimal global scheme exist for the SBA problem? We answer this question positively by proposing an algorithm, Procedure PT-Min_H (Figure 3), which can find the minimal element $\vec{H}^*$ in Π in polynomial time. For convenience of discussion, we will call $\frac{C_i}{q_i}$, $\frac{C_i - (r_i - \sum_{j \ne i} H_j^* - \tau)}{q_i - 1}$, and $\frac{C_i}{q_i - 1}$ Formulas I, II, and III, respectively.

Procedure PT-Min_H works as follows. In Step 1, we first "assume" that H_i^* is in Region I, and hence set $H_i := \frac{C_i}{q_i}$, for all i (note that $\frac{C_i}{q_i}$ is the minimum possible value of H_i^*), and set $F_1 := \{1, 2, \ldots, n\}$, and $F_2 := F_3 := \emptyset$. During the execution of the algorithm, F_i, $i = 1, 2, 3$, is the set of indices of H_i's whose current values are calculated using Formulas I, II, and III, respectively. However, the current values of H_i's may not really fall in the regions as we expected. Therefore, in Step 2, we calculate the correct region that each H_i falls into.

```
Procedure PT-Min_H
Step 1. For $i = 1$ to $n$ do $H_i := \frac{C_i}{q_i}$.
    Let $F_1 := \{1, 2, \ldots, n\}$, and $F_2 := F_3 := \emptyset$.
Step 2. Partition $\{1, 2, \ldots, n\}$ into three subsets $R_1, R_2$, and $R_3$, where $R_1 = \{i \mid r_i \geq \sum_j H_j + \tau\}$, $R_2 = \{i \mid \sum_{j \neq i} H_j + \tau < r_i < \sum_j H_j + \tau\}$, and $R_3 = \{i \mid r_i \leq \sum_{j \neq i} H_j + \tau\}$.
Step 3. If $F_1 = R_1$ then $\vec{H}^*$ has been found and the algorithm terminates.
Step 4. Let $R := F_1 - R_1$, and $S := R \cup F_2$. Let $b_i := C_i - X_i(\vec{H})$, $a_i := q_i - 1$, and $\delta_i = \frac{C_i}{q_i - 1} - H_i$, for $i \in S$. Solve the following linear programming (LP) formulation:
```

$$\text{maximize} \quad z = \sum_{i \in S} x_i \tag{4.7}$$

$$\text{subject to} \quad x_i \leq \frac{b_i}{a_i} + \frac{\sum_{j \in S, j \neq i} x_j}{a_i}, \tag{4.8}$$

$$x_i \leq \delta_i, \text{ and} \tag{4.9}$$

$$x_i \geq 0, \tag{4.10}$$

```
    for all $i \in S$.
    Let $(x_i^*)$, $i \in S$, be the solution of the above LP.
    Reset $H_i := H_i + x_i^*$, for all $i \in S$.
    Reset $F_1 := R_1$, $F_2 := \{i \in S \mid x_i^* < \delta_i\}$, and $F_3 := F_3 \cup \{i \in S \mid x_i^* = \delta_i\}$.
    Go to Step 2.
```

Figure 3: Polynomial-time algorithm for finding the minimal element $\vec{H}^*$ in Π.

During the execution of the algorithm, R_i, $i = 1, 2, 3$, is the set of indices of H_i's whose current values fall in Regions I, II, and III, respectively. If the formula used in calculating H_i matches the region that H_i really falls in, then the formula we used to calculate the value of H_i is correct. If this is true for all i then it means that all the H_i's have been calculated using the correct formulas, and hence the algorithm terminates (Step 3); otherwise, some of the H_i's are calculated using wrong formulas, i.e., some of the H_i's should have been calculated using Formula II or III, but were calculated using Formula I.

At the beginning of Step 4, $R = F_1 - R_1$ is the set of indices of H_i's whose current values are calculated using Formula I but actually falls in Region II or III, and whose corresponding b_i's ($b_i = C_i - X_i(\vec{H})$) are larger than 0. That is, $b_i > 0$ is the "deficiency" of H_i (the difference between the maximum message transmission time C_i and the minimum available transmission time $X_i(\vec{H})$), for each $i \in R$. It means that to satisfy the deadline constraint, the synchronous bandwidth H_i of node i, $i \in R$, should be increased to compensate the deficiency. However, the increase of the H_i's with $i \in R$ will incur non-zero deficiency for the H_i's with $i \in F_2$ and also for some of the H_i's with $i \in F_1 - R$. Since we are not sure which H_i's with $i \in F_1 - R$ will incur non-zero deficiency we will only take the H_i's with $i \in F_2$ into consideration and temporarily leave the H_i's with $i \in F_1 - R$ fixed at the value $\frac{C_i}{q_i}$. (Note that since $\frac{C_i}{q_i - 1}$ is the maximum possible value of H_i^*, those H_i's with $i \in F_3$ no longer need to be changed, and hence are also fixed at $\frac{C_i}{q_i - 1}$.)

It is easy to see that for each $i \in S = R \cup F_2$, the increase, x_i, in H_i makes the increase in $X_i(\vec{H})$ by $a_i \cdot x_i$. Therefore, H_i should be increased by (at least) $\frac{b_i}{a_i}$ to compensate the deficiency b_i. However, due to the increase, x_j, of some other H_j with $j \in S$ and $j \neq i$, $X_i(\vec{H})$ will be decreased by the same amount x_j, and hence increases the deficiency of X_i by x_j. If we let x_i be the amount of bandwidth increase for H_i, for each $i \in S$, then the actual deficiency of X_i will be $b_i + \sum_{j \in S, j \neq i} x_j$. Therefore, we expect to increase the bandwidth H_i by an amount $\frac{b_i}{a_i} + \frac{\sum_{j \in S, j \neq i} x_j}{a_i}$. However, if the value of $\frac{b_i}{a_i} + \frac{\sum_{j \in S, j \neq i} x_j}{a_i}$ becomes larger than $\delta_i = \frac{C_i}{q_i - 1} - H_i$ then X_i will move from Region I or II to Region III, and hence the synchronous bandwidth H_i need not be increased to a value larger than $\frac{C_i}{q_i - 1}$. That is, the increase x_i of H_i never needs to be greater than $\min(\delta_i, \frac{b_i}{a_i} + \frac{\sum_{j \in S, j \neq i} x_j}{a_i})$. Therefore, we have the two inequalities Eq. (4.8) and (4.9) in Step 4. We then solve the linear programming (LP) problem with the $3 \cdot |S|$ constraints (including the nonnegativity constraints Eq. (4.10)) and the objective function (maximize) $z = \sum_{i \in S} x_i$. We will show later that the LP defined in Step 4 has exactly one optimal solution (x_i^*) with $i \in S$, which satisfies either $0 < x_i^* = \delta_i \leq \frac{b_i}{a_i} + \frac{\sum_{j \in S, j \neq i} x_j}{a_i}$ or $0 < x_i^* = \frac{b_i}{a_i} + \frac{\sum_{j \in S, j \neq i} x_j}{a_i} < \delta_i$. Note that the H_i's with $i \in R_1$ are exactly those H_i's whose values are fixed at $\frac{C_i}{q_i}$ (Formula I). Therefore, we reset $F_1 := R_1$. Similarly, those H_i's with $i \in S$ and $x_i^* < \delta_i$ are exactly those H_i's whose new values are calculated according to Formula II, and those H_i's with $i \in S$ and $x_i^* = \delta_i$, and with $i \in F_3$ are exactly those H_i's whose (new) values are calculated or fixed at $\frac{C_i}{q_i - 1}$ (Formula III). Therefore, we reset F_2 and F_3 accordingly. Now, due to the increase of those H_j's with $j \in S$, some of the H_i's with $i \in R_1$ may now move from Region I to Region II or III, and hence the calculation of this kind of H_i's may still use wrong formulas. So, we need to go back to Step 2 and check if further changes are needed.

Before proving the correctness and giving the time complexity of Procedure PT-Min_H, we need the following theorem and lemma.

Theorem 4: Consider the following linear programming formulation

$$\text{maximize} \quad z = \sum_{i=1}^{k} c_i \cdot x_i \tag{4.11}$$

$$\text{subject to} \quad x_i \le \frac{b_i}{a_i} + \frac{\sum_{j\neq i} x_j}{a_i}, \tag{4.12}$$

$$x_i \le \delta_i, \text{ and} \tag{4.13}$$

$$x_i \ge 0, \tag{4.14}$$

for $i = 1, 2, \ldots, k$. If $a_i > 0$, $b_i \ge 0$, $\delta_i > 0$, $c_i > 0$, for all i, and there exists at least one index j such that $b_j > 0$, then

(a) the optimal value of the objective function z exists and is bounded,

(b) $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_k^*)$ satisfies the following condition

$$\text{either} \quad x_i^* = \delta_i \le \frac{b_i}{a_i} + \frac{\sum_{j\neq i} x_j^*}{a_i} \tag{4.15}$$

$$\text{or} \quad x_i^* = \frac{b_i}{a_i} + \frac{\sum_{j\neq i} x_j^*}{a_i} < \delta_i \tag{4.16}$$

for each $i = 1, 2, \ldots, k$, where $\mathbf{x}^*$ is an optimal solution for the LP,

(c) any vector $\mathbf{y} = (y_1, y_2, \ldots, y_k)$ satisfying Eq. (4.15)–(4.16) must have all positive components, i.e., $y_i > 0$, $\forall\, i$, and

(d) there is a *unique* vector satisfying Eq. (4.15)–(4.16), i.e., $\mathbf{x}^*$ is the only vector satisfying Eq. (4.15)–(4.16). □

The proof of Theorem 4 is omitted due to space limitations. See [10] for details. Note that the uniqueness of the solution of Eq. (4.15)–(4.16) is the key point of the optimality of our algorithm, since it means that the new values of H_i's found in each iteration of Step 4 won't be too large.

Before any further discussion, we need the following notation. Let F_i^- and F_i^+ denote the values of F_i before and after the execution of Step 4 in an iteration of the loop from Step 2 to Step 4, respectively, and let $\vec{H}^-$ and $\vec{H}^+$ be similarly defined. Let $V \in \{R_i\text{'s}, R, S, b_i\text{'s}, \delta_i\text{'s}\}$ denote the value of the corresponding variable in the current iteration of Step 4, and let V^+ denote the value of the corresponding variable in the next iteration of Step 4, i.e., $R_1^+ = \{i \mid r_i \ge \sum_j H_j^+ + \tau\}$, $R_2^+ = \{i \mid \sum_{j\neq i} H_j^+ + \tau < r_i < \sum_j H_j^+ + \tau\}$ $R_3^+ = \{i \mid r_i \le \sum_{j\neq i} H_j^+ + \tau\}$, $R^+ = F_1^+ - R_1^+$, $S^+ = R^+ \cup F_2^+$, $b_i^+ = C_i - X_i(\vec{H}^+)$, and $\delta_i^+ = \frac{C_i}{q_i - 1} - H_i^+$.

As mentioned earlier, F_1, F_2, and F_3, are the sets of indices of H_i's whose current values are calculated according to Formulas I, II, and III, respectively. If H_i is calculated by Formula I and the values of q_i and r_i are changed to $q_i + 1$ and 0, respectively, then the deadline constraint for node i is guaranteed to be satisfied. A similar statement also holds for H_i's that are calculated by Formula III. Thus, for $\# \in \{-, +\}$ we define

- $q_i^\# = q_i + 1$, $r_i^\# = 0$, for $i \in F_1^\#$,
- $q_i^\# = q_i$, $r_i^\# = r_i$, for $i \in F_2^\#$,
- $q_i^\# = q_i$, $r_i^\# = 0$, for $i \in F_3^\#$,
- $X_i^\#(\vec{H}) = (q_i^\# - 1) \cdot H_i + \max(0, \min(r_i^\# - (\sum_{j\neq i} H_j + \tau), H_i))$, and
- $\Pi^\# = \{\vec{H} \mid X_i^\#(\vec{H}) \ge C_i, \text{ for all } i\}$.

We will also use the following assumptions in Lemma 1 and Theorem 5.

A1. $\vec{H}^- \le \vec{H}^*$,

A2. $H_i^- = \frac{C_i}{q_i}$, for $i \in F_1^-$, $\frac{C_i}{q_i} < H_i^- = \frac{C_i - (r_i - \sum_{j\neq i} H_j^* - \tau)}{q_i - 1} < \frac{C_i}{q_i - 1}$, for $i \in F_2^-$, and $H_i^- = \frac{C_i}{q_i - 1}$, for $i \in F_3^-$,

A3. $b_i = 0$, for all $i \in F_2^-$, $b_i > 0$, for all $i \in R = F_1^- - R_1$, and $\delta_i > 0$, for all $i \in S$, and

A4. $\vec{H}^-$ is the minimal element in Π^-.

The following lemma states that assumptions A1–A4 are loop invariants.

Lemma 1: If assumptions A1–A4 are true and assume R is nonempty,then

(a) $\vec{H}^- < \vec{H}^+ \le \vec{H}^*$,

(b) $H_i^+ = \frac{C_i}{q_i}$, for $i \in F_1^+$, $\frac{C_i}{q_i} < H_i^+ = \frac{C_i - (r_i - \sum_{j\neq i} H_j^* - \tau)}{q_i - 1} < \frac{C_i}{q_i - 1}$, for $i \in F_2^+$, and $H_i^+ = \frac{C_i}{q_i - 1}$, for $i \in F_3^+$,

(c) $b_i^+ = 0$, for all $i \in F_2^+$, and $b_i^+ > 0$, for all $i \in R^+ = F_1^+ - R_1^+$, and $\delta_i^+ > 0$, for all $i \in S^+$, and $R_1^+ = \{i \mid r_i \ge \sum_j H_j^+ + \tau\}$, and

(d) $\vec{H}^+$ is the minimal element in Π^+. □

The proof of the lemma is omitted due to space limitations. See [10] for a detailed account. We now prove the correctness and give the time complexity of Procedure PT-Min_H in the following theorem.

Theorem 5: Let $\vec{H}^{(k)}$ be the value of $\vec{H}$ before the execution of the k-th iteration of Step 3 in the loop from Step 2 to Step 4. We have

(a) assumptions A1–A4 are true for all iterations of the loop from Step 2 to Step 4; in particular, $\vec{H}^{(k)} \leq \vec{H}^*$, for all k,

(b) procedure PT-Min_H terminates, and at termination (assuming after the l-th iteration of Step 3) $\vec{H}^{(l)} = \vec{H}^*$, and

(c) the time complexity of Procedure PT-Min_H is at most $O(nM)$, where M is the time complexity for solving an LP with $3n$ constraints and n variables.

Proof: We prove (a) by induction on k. The first time when the algorithm goes to Step 3, $H_i^{(1)} = \frac{C_i}{q_i} \leq H_i^*$, for all i, $F_1 = \{1, 2, \ldots, n\}$, and $F_2 = F_3 = \emptyset$. It is easy to check that assumptions A1, A2, and A4 are true (note that assumption A3 is meaningful only if the algorithm goes to Step 4). If the algorithm terminates at the first iteration of Step 3, then for all i, $r_i \geq \sum_j H_j^{(1)} + \tau$ and $X_i(\vec{H}^{(1)}) = C_i$. Therefore, $\vec{H}^{(1)} = \vec{H}^*$. If the algorithm goes to Step 4, it means that $R = F_1 - R_1 \neq \emptyset$, and $r_i < \sum_j H_j^{(1)} + \tau$, for $i \in R$. Therefore, $b_i = C_i - X_i(\vec{H}^{(1)}) > 0$ (note that $F_2 = \emptyset$). Hence, assumption A3 is true.

Assume (a) is true at the beginning of the k-th iteration of Step 4. By Lemma 1, (a) is still true after the execution of Step 4. Therefore, by the logic of induction proof, (a) is true.

Next, since for each iteration, the size of F_1 will be decreased by $|R| = |F_1 - R_1|$, and if $|R| = 0$ the algorithm will terminate. Therefore, the algorithm is guaranteed to terminate after (at most) n iterations of Step 4, and when the algorithm terminates at the l-th iteration of Step 3, $R = \emptyset$, which means that the current value of H_i is calculated by Formula I, II, or III (i.e., $i \in F_i$, $i = 1, 2, 3$,) if and only if H_i is in Region I, II, or, III (i.e., $i \in R_i$, $i = 1, 2, 3$), respectively. Therefore, $X_i(\vec{H}^{(l)}) = C_i$, for all i, and hence, $\vec{H}^{(l)} = \vec{H}^*$. Thus, (b) is proved.

It is easy to see that the time complexity of the algorithm is dominated by the LP described in Step 4, which has at most $3n$ constraints and n variables. Since there are at most n iterations of Step 4, we need to solve the LP at most n times. Therefore, the time complexity of the algorithm is given as in (c). □

Note that although the famous simplex method for solving LP has an exponential-time worst-case complexity,[2] LP has been proved to be polynomial-time solvable. Therefore, Procedure PT-Min_H is a polynomial-time algorithm.

[2]However, in practice, the simplex method performs exceedingly well.

5 Conclusion

We have presented an optimal global SBA scheme which is guaranteed to find a feasible solution in polynomial time for allocating synchronous bandwidths when such an allocation exists. In the only previously-known optimal global SBA scheme proposed in [2], although the computation is proved to converge to the optimal solution, it may not terminate in polynomial time, or, theoretically, may not terminate at all. Moreover, even though a traditional engineering approach can be applied in their SBA scheme to stop the algorithm at a certain point, the synchronous bandwidths found in their algorithm are still unusable. The algorithm described in this paper is, to our best knowledge, the first polynomial-time optimal global SBA scheme.

References

[1] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed token medium access control protocol," *IEEE Trans. on Computers*, vol. 43, pp. 327–350, March 1994.

[2] B. Chen, G. Agrawal, and W. Zhao, "Optimal synchronous capacity allocation for hard real-time communications with the timed token protocol," in *Proc. of the 13th Real-Time Systems Symposium*, (Phoenix, Arizona), pp. 198–207, December 1992.

[3] Q. Zheng and K. G. Shin, "Synchronous bandwidth allocation in FDDI networks," in *Proc. of ACM Multimedia'93*, pp. 31–38, August 1993.

[4] C.-C. Han, K. G. Shin, and C.-J. Hou, "On non-existence of optimal local synchronous bandwidth allocation schemes for the timed-token MAC protocol," in *Proc. of IEEE Int'l Phoenix Conf. on Computers and Communications*, March 1995.

[5] M. J. Johnson, "Proof that timing requirements of the FDDI token ring protocol are satisfied," *IEEE Trans. on Commun.*, vol. COM-35, pp. 620–625, June 1987.

[6] K. C. Sevcik and M. J. Johnson, "Cycle time properties of the FDDI token ring protocol," *IEEE Trans. on Software Eng.*, vol. SE-13, pp. 376–385, March 1987.

[7] "Fiber Distributed Data Interface (FDDI) — Token Ring Media Access Control (MAC)." American National Standard, ANSI X3.139, 1987.

[8] R. Jain, "FDDI: Current issues and future plans," *IEEE Commun. Magazine*, pp. 98–105, Sept. 1993.

[9] B. Chen and W. Zhao, "Properties of the timed token protocol," Tech. Rep. 92-038, Department of Computer Science, Texas A&M University, October 1992.

[10] C.-C. Han, K. G. Shin, and C.-J. Hou, "Synchronous bandwidth allocation for real-time communications with the timed-token MAC protocol." submitted for publication.