The Timed-Token Protocol for Real-Time Communications

Nicholas Malcolm and Wei Zhao
Texas A&M University

Real-time applications have stringent timing requirements. The timed-token protocol guards against timing failures by guaranteeing synchronous bandwidth regardless of the behavior of asynchronous messages.

Most real-time applications such as industrial process control depend on digital computers. Typically, sensory data is periodically transmitted over a communication network or "data highway" to a digital computer controlling the process.[1] Such systems often have stringent timing requirements, where a failure could lead to catastrophe. Further, many of these systems are distributed because the applications themselves are often physically distributed and because distributed systems share resources well and are reliable and extensible.

The key to success in such systems is the timely execution of data-processing tasks that usually reside on different nodes and communicate with one another to accomplish a common goal. End-to-end deadline guarantees are not possible without a communication network that supports the timely delivery of intertask messages.

The timed-token protocol is a token-passing protocol in which each node receives a guaranteed share of the network bandwidth.[2] Partly because of this property, the timed-token protocol has been incorporated into a number of network standards, including the IEEE 802.4 token bus,[3] the Fiber Distributed Data Interface (FDDI),[4] and the Survivable Adaptable Fiber-Optic Embedded Network (Safenet).[5] Networks based on these standards are becoming increasingly popular in new generation real-time systems. In particular, the IEEE 802.4 standard is included in the Manufacturing Automation Protocol (MAP),[6] which has been widely used in computer-integrated manufacturing and industrial applications.

Meeting message deadlines requires proper control of medium access. In the timed-token protocol, access to the communication medium is controlled by a token that is passed among the nodes in a circular fashion. Messages are segregated into two separate classes: *synchronous* and *asynchronous*.[2] Synchronous messages, used for real-time communication, can have deadline constraints and thus are given a guaranteed share of the network bandwidth. We will focus on

0018-9162/94/$4.00 © 1994 IEEE

meeting the deadlines of synchronous messages.

The idea behind the timed-token protocol is to control the token rotation time. A protocol parameter called the *target token rotation time* (TTRT) gives the expected token rotation time. Each node is assigned a portion of the TTRT, known as its *synchronous bandwidth* (H_i), which is the maximum time a node is permitted to transmit synchronous messages *every time* it receives the token. When a node receives the token, it transmits its synchronous messages, if any, for a period not longer than its allocated synchronous bandwidth. It can then transmit its asynchronous messages if the time elapsed since the previous token departure from the same node is less than the value of TTRT, that is, only if the token arrived earlier than expected.

To guarantee that synchronous-message deadlines are met, system designers must carefully choose network parameters such as the synchronous bandwidth, the TTRT, and the buffer size. The synchronous bandwidth is the most critical of the three. If the synchronous bandwidth allocated to a node is too small, the node might not have enough network access time to transmit messages before their deadlines. Conversely, large synchronous bandwidths can result in a long token rotation time, which can also cause message deadlines to be missed.

Proper selection of TTRT is also important. Let τ be the token walk time around the network. The proportion of time taken due to token walking is given by τ / TTRT. The maximum network utilization available to user applications is then $1 - \tau$ / TTRT.[7] A smaller TTRT results in less available utilization and limits network capacity. On the other hand, if TTRT is too large, the token may not arrive often enough at a node to meet message deadlines.

Each node has buffers for outgoing and incoming synchronous messages. The sizes of these buffers also affect the network's real-time performance. A buffer that is too small can result in lost messages due to buffer overflow; one that is too large wastes memory.

In the rest of this article, we discuss the methods of parameter selection and their effect on the real-time performance of the timed-token protocol.

A framework for the timed-token protocol

Our network model contains n nodes arranged in a ring. The timed-token protocol controls message transmission, and outgoing messages are queued at each node in FIFO (first in, first out) order. We assume that the network operates without any faults.

Token walk time τ includes the node-to-node delay and the token transmission time and thus represents the portion of TTRT that is not available for message transmission. Let α be the ratio of τ to TTRT, that is, $\alpha = \tau$ / TTRT; α thus represents the proportion of time that is not available for transmitting messages.

There are n streams of synchronous messages, $S_1, \ldots, S_n$, with stream S_i originating at node i. Each synchronous-message stream S_i can be characterized as $S_i = (C_i, P_i, D_i)$, where

- C_i is the maximum amount of time required to transmit a message in the stream. This includes the time to transmit both the payload data and the message headers.
- P_i is the interarrival period between messages in the stream. Let the first message in stream S_i arrive at node i at time $t_{i,1}$. The jth message in stream S_i will arrive at node i at time $t_{i,j} = t_{i,1} + (j - 1) P_i$, where $j \geq 1$.
- D_i is the *relative deadline* of messages in the stream, that is, the maximum amount of time that can elapse between a message arrival and completion of its transmission. Thus, the transmission of the jth message in stream S_i, which arrives at $t_{i,j}$, must be completed by $t_{i,j} + D_i$, which is the message's *absolute deadline*. (The terms "relative" and "absolute" will be omitted when the meaning is clear from the context.)

Throughout this article, we make no assumptions about the destination of synchronous-message streams. Several streams may be sent to one node. Alternatively, multicasting may occur in which messages from one stream are sent to several nodes.

If the parameters (C_i, P_i, D_i) are not completely deterministic, then we must use their worst-case values. For example, if the period varies between 80 milliseconds (ms) and 100 ms, we must assume a period of 80 ms. By using pseudosynchronous streams, time-constrained asynchronous messages can have their deadlines guaranteed: For each source of time-constrained asynchronous messages, we create a corresponding synchronous-message stream. We then promote the time-constrained asynchronous messages to the synchronous class and send them as synchronous messages in the corresponding synchronous-message stream.

Each synchronous-message stream places a certain load on the system. We need a measure for this load. We define the *effective utilization*, U_i, of stream S_i as follows:

$$U_i = \frac{C_i}{\min(P_i, D_i)} \tag{1}$$

Because a message of length C_i arrives every P_i time units, $U_i = C_i / P_i$ denotes the load presented by stream S_i. An exception occurs when $D_i < P_i$. A message with such a small deadline must be sent relatively urgently, even if the period is very large. Thus $U_i = C_i / D_i$ is used to reflect the load in this case.

The total effective utilization, U, of a synchronous-message set can now be defined as

$$U = \sum_{i=1}^{n} U_i \tag{2}$$

This measures the demands placed on the system by the entire synchronous-message set. To meet message deadlines, it is necessary that $U \leq 1$.

Constraints. As mentioned earlier, several parameters are crucial in guaranteeing synchronous-message deadlines. Any choice of these parameters must satisfy several constraints.

The protocol constraint. This constraint states that the total bandwidth allocated to synchronous messages must be less than the available network bandwidth, that is,

$$\sum_{i=1}^{n} H_i \leq TTRT - \tau \tag{3}$$

This constraint is necessary to ensure stable operation of the timed-token protocol.

The deadline constraint. This constraint simply states that every synchronous message must be transmitted before its (absolute) deadline. Formally, let $s_{i,j}$ be the time that the transmission of the jth message in stream S_i is completed. The deadline constraint requires that for $i = 1, \ldots, n$ and $j = 1, 2, \ldots,$

$$s_{i,j} \le t_{i,j} + D_i \qquad (4)$$

where $t_{i,j}$ is the arrival time and D_i is the (relative) deadline. Note that in the above inequality, $t_{i,j}$ and D_i are given by the application, but $s_{i,j}$ depends on the synchronous-bandwidth allocation and the choice of TTRT.

Johnson and Sevcik[8] have shown that the maximum amount of time that may pass between two consecutive token arrivals at a node can approach 2TTRT. This bound holds regardless of the behavior of asynchronous messages in the network. To satisfy the deadline constraint, a node must have at least one opportunity to send each synchronous message before the message deadline expires. Therefore, satisfaction of the deadline constraint requires that for $i = 1, \ldots, n,$

$$D_i \ge 2\text{TTRT} \qquad (5)$$

(Equation 5 is a necessary, but not a sufficient, condition for satisfying the deadline constraint. For all message deadlines to be met, the synchronous bandwidths H_i must be chosen appropriately.)

The buffer constraint. This constraint states that the size of the buffers at each node must be sufficient to hold the maximum number of outgoing or incoming synchronous messages that can be queued at the node. This constraint is necessary to ensure that messages are not lost due to buffer overflow.

In the remainder of this article, we discuss how these parameters should be set and the consequent impact on system performance.

Allocating synchronous bandwidth

There are two classes of synchronous-bandwidth allocation schemes: local and global. These schemes differ in the type of information they use in allocating H_i. A local synchronous-bandwidth allocation scheme can only use information available locally to node i, including the parameters of stream S_i (C_i, P_i, and D_i), as well as TTRT and τ, which are known to all nodes. A global scheme, on the other hand, can use global information, including both local information and information regarding the parameters of synchronous-message streams originating at other nodes.

The scheme described here is simple, intuitive, well understood, and will be used in distributed real-time systems in the future.

A local scheme is preferable from a network management perspective. If the parameters of stream S_i change, then only the synchronous bandwidth H_i of node i needs to be recalculated. The synchronous bandwidths at other nodes do not need to be changed because they were calculated independently of S_i. This makes a local scheme flexible and suited for use in dynamic environments where synchronous-message streams are often initiated or terminated.

In a global scheme, if the parameters of S_i change, it may be necessary to recompute the synchronous bandwidths at all nodes. Thus, a global scheme might not be well suited to a dynamic environment. On the other hand, the extra information used by a global scheme might allow it to handle more traffic than a local scheme. However, local schemes can perform *very close* to the optimal synchronous-bandwidth allocation scheme when message deadlines are equal to message periods. Consequently, this article concentrates on local schemes, given their good performance and network management properties.

A local allocation scheme. Several local synchronous-bandwidth allocation schemes have been proposed, both for the case of $D_i = P_i$ and for the case of $D_i \ne P_i$.[9-11] These schemes all have a similar worst-case performance. In this article, we consider a simplified version of the scheme proposed in Malcolm and Zhao[10] and in Zheng and Shin.[11] With this scheme, the synchronous bandwidth for node i is allocated as

$$H_i = \frac{U_i D_i}{\left\lfloor \frac{D_i}{TTRT} - 1 \right\rfloor} \qquad (6)$$

Intuitively, this scheme follows the flow conservation principle. Between the arrival of a message and its deadline D_i time units later, node i will have at least

$$\left\lfloor \frac{D_i}{TTRT} - 1 \right\rfloor H_i$$

units of transmission time available for synchronous messages regardless of the number of asynchronous messages in the network.[10] During these D_i time units, $U_i D_i$ approximates the load on node i. Thus, the synchronous bandwidth in Equation 6 is just sufficient to handle the load on node i between the arrival of a message and its deadline.

The scheme defined in Equation 6 is simple, intuitive, and well understood. Furthermore, it has been adopted for use with the Safenet standard and thus will be used in distributed real-time systems in the future.

Schedulability testing. A message set is schedulable if it meets the deadlines of its synchronous messages, that is, if it satisfies the protocol and deadline constraints in Equations 3 and 4.

Testing the protocol constraint is relatively easy, whereas a direct test of the deadline constraints might involve a complicated and tedious timing analysis. Fortunately, this can be avoided if the bandwidths are allocated according to Equation 6. When this scheme is used, the protocol constraint condition defined in Equation 3 implies the deadline constraint condition in Equation 4. Testing the protocol constraint alone is sufficient to ensure that both constraints are satisfied.

A second method of schedulability testing is to use the measure of the *worst case achievable utilization*, widely used in real-time systems. For a synchronous-bandwidth allocation scheme, the worst case achievable utilization U^* defines an upper bound on the effective

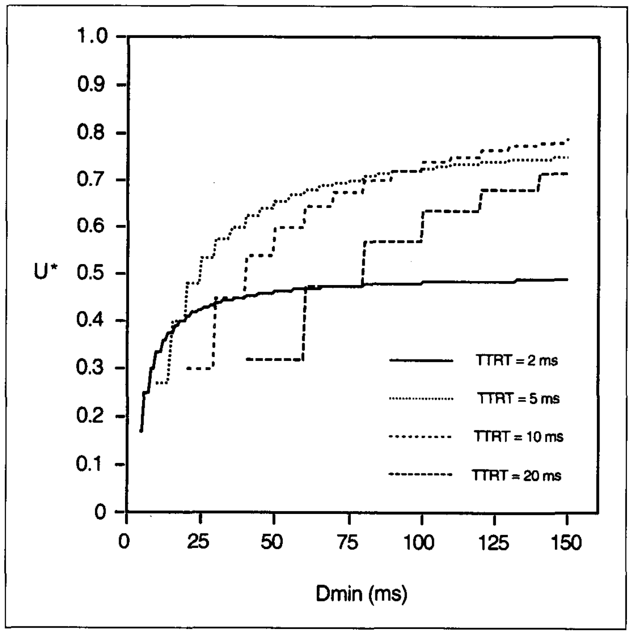

Figure 1. Worst case achievable utilization (U^*) versus minimum deadline ($D_{\min}$).

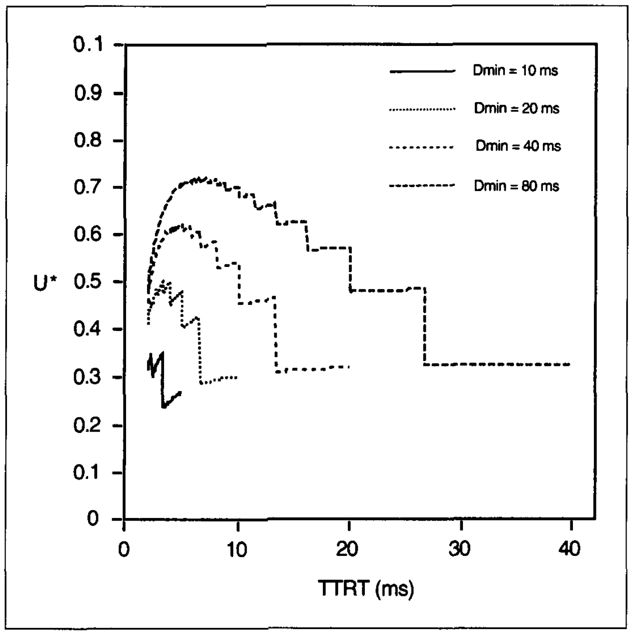

Figure 2. Worst case achievable utilization (U^*) versus TTRT.

utilization of a message set: If the effective utilization is no more than the upper bound, both the protocol and the deadline constraints are always satisfied. For the scheme defined in Equation 6, U^* is

$$U^* = \frac{\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor - 1}{\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor + 1}(1-\alpha) \tag{7}$$

where $D_{\min}$ is the minimum deadline. For any message set, if its effective utilization (Equation 2) is less than U^*, both the protocol and deadline constraints are guaranteed to be satisfied.

In practice, using this measure can simplify network management considerably. The parameters of a synchronous-message set can be modified freely while still maintaining schedulability, provided that the effective utilization remains less than U^*.

Figure 1 shows the effect of $D_{\min}$ and TTRT on the worst case achievable utilization given in Equation 7. To obtain the figure, we plotted Equation 7, with τ as 1 ms, a typical value for an FDDI network. Three observations can be made from Figure 1 and Equation 7:

(1) For a fixed value of TTRT, U^* increases when $D_{\min}$ increases. Equation 7 shows that when $D_{\min}$ approaches infinity, U^* approaches $(1 - \alpha) = (1 - \tau / \text{TTRT})$. That is, as the deadlines become larger, the worst case achievable utilization approaches the network's available utilization.

(2) For a system in which all relative deadlines are equal to the corresponding message periods ($D_i = P_i$), we can achieve a worst case achievable utilization of 1/3 $(1 - \alpha)$.[9] That result can be seen as a special case of Equation 7: When $D_{\min} = 2\text{TTRT}$, we have

$$\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor = 2$$

and $U^* = 1/3\,(1 - \alpha)$

(3) TTRT clearly has an impact on U^*. Figure 1 shows that when $D_{\min}$ = 50 ms, TTRT = 5 ms gives a higher U^* than the other plotted values of TTRT. When $D_{\min}$ = 125 ms, TTRT = 10 ms gives a higher U^* than the other plotted values of TTRT. This observation provides motivation for maximizing U^* by properly selecting TTRT once $D_{\min}$ is given.

Selecting TTRT

Recall that TTRT determines the expected value of the token rotation time. In contrast to the synchronous bandwidths of individual nodes, TTRT is common to all the nodes and should be kept constant during runtime. Since TTRT affects U^*, we would like to choose TTRT so that U^* is maximized, increasing the amount of real-time traffic that can be supported by the network. The optimal value of TTRT is given by[10]

$$TTRT = \frac{D_{\min}}{\left\lceil \frac{-3+\sqrt{9+\frac{8D_{\min}}{\tau}}}{2} \right\rceil} \tag{8}$$

The impact of an appropriate selection of TTRT is further evident from Figure 2, which uses Equation 8 to show U^* versus TTRT for several different values of $D_{\min}$. As with Figure 1, τ is taken to be 1 ms. Figure 2 yields the following observations:

(1) The curves in Figure 2 verify the prediction of the optimal TTRT value given by Equation 8; for example, when $D_{\min}$ = 40 ms, the optimal value of TTRT is 5 ms. The curve clearly indicates that at TTRT = 5 ms the worst case

achievable utilization is maximized. Similar observations can be made for the other cases.

(2) As indicated in Equation 8, the optimal TTRT is a function of $D_{\min}$, coinciding with the values depicted in Figure 1. Generally, as $D_{\min}$ increases, the optimal TTRT increases. For example, the optimal values of TTRT are approximately 2.5 ms for $D_{\min}$ = 10 ms, 4 ms for $D_{\min}$ = 20 ms, 5 ms for $D_{\min}$ = 40 ms, and 6.67 ms for $D_{\min}$ = 80 ms.

(3) The choice of TTRT has a large effect on U^*. Consider the case of $D_{\min}$ = 40 ms shown in Figure 2. If TTRT is too small (say TTRT = 2 ms), U^* can be as low as 45 percent. If TTRT is too large (say TTRT = 15 ms), U^* can be as low as 31 percent. However, when we use the optimal value of TTRT (TTRT = 5 ms), U^* is 62 percent. This is an improvement of 17 percent and 31 percent, respectively, over the previous two cases.

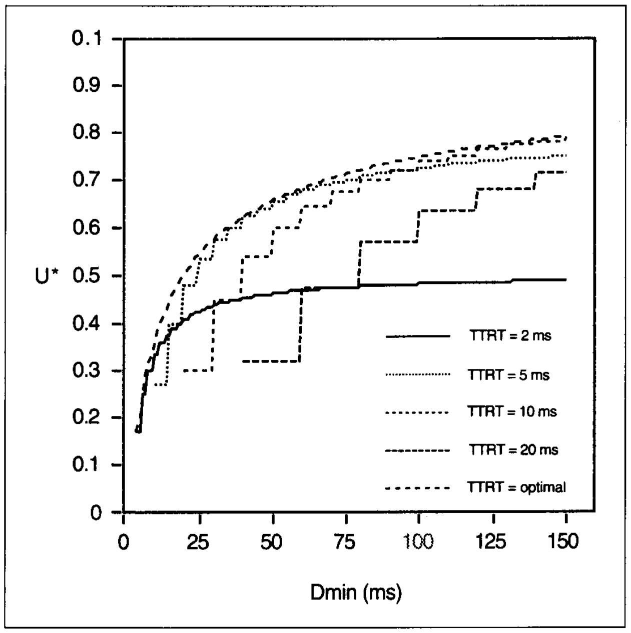

Figure 3. Worst case achievable utilization (U^*) versus minimum deadline ($D_{\min}$) with optimal TTRT.

Figure 3 shows U^* versus $D_{\min}$ when an optimal value of TTRT is used. For ease of comparison, Figure 3 also shows the results of Figure 1.

Buffer requirements

Messages in a network can be lost if there is insufficient buffer space at either the sending or the receiving node. There are two buffers for synchronous messages on each node. The send buffer holds messages waiting to be transmitted to other nodes; the receive buffer holds messages that have been received from other nodes and are waiting to be processed by the host.

We consider the send buffer first. When $D_i \le P_i$, a message must be transmitted within the period it arrives, so the send buffer needs to accommodate only one message.

In the case of $D_i > P_i$, a message may wait as long as D_i time units without violating its deadline constraint. At node i, messages from stream S_i arrive every P_i time units, requesting transmission. During the D_i time units after a message arrives

$$\left\lfloor \frac{D_i}{P_i} \right\rfloor$$

more messages will arrive. Thus, the send buffer may have to accommodate

$$\left\lfloor \frac{D_i}{P_i} \right\rfloor + 1$$

messages. A very large deadline might become a problem in practice due to excessive buffer requirements; in a voice transmission, for example, deadlines can be ten to a hundred times larger than the period.

Fortunately, when the synchronous bandwidths are allocated according to Equation 6, this problem can be eliminated because the waiting time of a message from stream S_i is bounded by $\min(D_i, P_i + 2\text{TTRT})$.[12] Thus, the maximum number of messages from S_i that can be queued at node i is no more than

$$\left\lceil \frac{\min(D_i, P_i+2TTRT)}{P_i} \right\rceil$$

This allows us to deduce the required size of the send buffer. Let BS_i denote the send buffer size at node i. The send buffer will never overflow if

$$BS_i \ge \left\lceil \frac{\min(D_i, P_i+2TTRT)}{P_i} \right\rceil B_i \qquad (9)$$

where B_i is the number of bytes in a message of stream S_i.

Interestingly, if $D_i \ge P_i + 2\text{TTRT}$, then

$$BS_i \ge \left\lceil \frac{P_i+2TTRT}{P_i} \right\rceil B_i \qquad (10)$$

That is, the send buffer requirements for stream S_i are independent of the deadline D_i. As mentioned earlier, one would suppose that increasing a stream's deadline would increase the buffer requirements. When we use the scheme in Equation 6, however, increasing the deadline once it is sufficiently large does not affect the buffer requirements.

Now consider the receive buffer. Suppose that node j is the destination node of messages from nodes $j_1, \ldots, j_k$. This means that messages from streams $S_{j_1}, \ldots, S_{j_k}$ are being sent to node j. The size of the receive buffer at node j depends not only on the message traffic from these streams but also on the speed at which the host of node j is able to process incoming messages from other nodes. The host at node j has to be able to keep pace with the incoming messages; otherwise, the receive buffer at node j can overflow. We assume that the host at node j can process a message from stream S_{j_i} within P_{j_i} time units of its arrival.

Assuming that the host is fast enough, the number of messages from S_i that can be queued at the destination node is bounded by

$$\left\lceil \frac{\min(D_i, P_i+2TTRT)}{P_i} \right\rceil + 1$$

With this bound, we can derive the space requirements for the receive buffer at node j. Let BR_j denote the receive buffer size at node j. The receive buffer will never overflow if

Table 1. Sample synchronous-message streams.

Stream	Origin	Size	C_i	P_i	D_i	Destination
S_1	Node 1	0.25 Mbits	2.5 ms	40 ms	32 ms	Node 3
S_2	Node 2	0.5 Mbits	5 ms	20 ms	40 ms	Node 3
S_3	Node 3	0.5 Mbits	5 ms	50 ms	50 ms	Node 1

Table 2. Buffer sizes at the nodes.

	Send Buffer	Receive Buffer
Node 1	0.25 Mbits	2 * 0.5 Mbits = 1.0 Mbits
Node 2	2 * 0.5 Mbits = 1.0 Mbit	0
Node 3	0.5 Mbits	2 * 0.25 Mbits + 3 * 0.5 Mbits = 2.0 Mbits

$$BR_j \geq \sum_{i=1}^{k} \left\lceil \frac{\min\left(D_{j_i}, P_{j_i}+2TTRT\right)}{P_{j_i}} + 1 \right\rceil B_{j_i} \tag{11}$$

As in the case of the send buffer, we can observe from Equation 11 that, if the deadlines are large, the required size of the receive buffer will not grow as the deadlines increase.

An example

The following example illustrates this methodology for testing the schedulability of a set of synchronous messages. Consider a network with the three synchronous-message streams shown in Table 1. Suppose that TTRT = 8 ms and τ = 1 ms. These are typical values for TTRT and τ in a FDDI network. We then have $\alpha = \tau$ / TTRT = 0.125. From Equation 6, the synchronous bandwidths are allocated as

$$H_1 = 0.83 \text{ ms} \tag{12}$$

$$H_2 = 2.5 \text{ ms} \tag{13}$$

$$H_3 = 1.0 \text{ ms} \tag{14}$$

One way to determine schedulability is to verify that the protocol constraint is satisfied. That is, we confirm that the total synchronous bandwidth allocated is less than the available network bandwidth. Because $H_1 + H_2 + H_3 = 4.33$ ms $\leq$ TTRT $- \tau$ = 8 ms $-$ 1 ms = 7 ms, both the protocol and the deadline constraints are satisfied. The deadlines of all messages will be satisfied as long as the network operates normally.

We can also test schedulability by calculating U^*. From Equation 7, we have

$$U^* = \frac{\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor - 1}{\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor + 1}(1-\alpha) \tag{15}$$

$$= 0.525 \tag{16}$$

Now, the effective utilization of the message set is

$$U = U_1 + U_2 + U_3 \tag{17}$$

$$= 0.43 \tag{18}$$

Thus $U < U^*$, so the message set is schedulable. $U^* - U = 0.095$ gives the safety margin of the system. The utilization of the message set can increase by 9.5 percent without violating the protocol or the deadline constraints.

The above calculation assumes that TTRT = 8 ms. If TTRT is selected using Equation 8, U^* can be maximized. From Equation 8 we have

$$TTRT = \frac{D_{\min}}{\left\lceil \frac{-3+\sqrt{9+\frac{8D_{\min}}{\tau}}}{2} \right\rceil} \tag{19}$$

$$= 4.57 \text{ ms} \tag{20}$$

This value of TTRT would give $\alpha = \tau$ / TTRT = 0.22 and

$$U^* = \frac{\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor - 1}{\left\lfloor \frac{D_{\min}}{TTRT} \right\rfloor + 1}(1-\alpha) \tag{21}$$

$$= 0.585 \tag{22}$$

Thus, given the current load of 43 percent, an additional 15 percent load can be added without violating the protocol and deadline constraints. Hence, using the optimal value of TTRT, the system can accommodate more real-time traffic.

Finally, Table 2 shows the required buffer sizes, using the formulas in Equations 9 and 11. The receive buffer at node three must allow for messages from both stream S_1 and from stream S_2.

Our methodology for using the timed-token protocol in real-time communication has several advantages. Because the synchronous-bandwidth allocation method uses only information local to a node, we can create or modify synchronous-message streams locally without reinitializing the network. Furthermore, our parameter selection methods are compatible with current standards of the timed-token protocol such as IEEE 802.4 and FDDI, and the schedulability tests are simple to implement and computationally efficient. Finally, the method is valid regardless of the behavior of asynchronous messages, because it assumes the worst possible impact of asynchronous traffic. ■

Acknowledgments

The work described in this paper was supported in part by NSF under grant NCR-9210583, by ONR under grant N00014-92-J-4031, and by Texas A&M University under an Engineering Excellence Grant.

References

1. J. Borer, *Microprocessors in Process Control*, Elsevier Applied Science Publishers Ltd., Essex, England, 1991.
2. R.M. Grow, "A Timed-Token Protocol For Local Area Networks," *Proc. Electro/82, Token Access Protocols*, Electronic Conventions, Inc., El Segundo, Calif., Paper 17/3, May 1982.
3. *IEEE Standard 802.4: Token-Passing Bus Access Method and Physical Layer Specifications*, IEEE CS Press, Los Alamitos, Calif., Order No. EZ938, 1985.
4. *ISO Int'l Standard 9314-2, Information Processing Systems: Fibre Distributed Data Interface (FDDI); Part 2: Token Ring Media Access Control (MAC)*, Int'l Standards Organization, Geneva, 1989.
5. *Survivable Adaptable Fiber-Optic Embedded Network*, MIL-STD-2204, US Dept. of Defense, Washington, D.C., Sept. 1992.

6. L.J. McGuffin, L.O. Reid, and S.R. Sparks, "MAP/TOP in CIM Distributed Computing," *IEEE Network*, Vol. 2, No. 3, May 1988, pp. 23-31.

7. J.N. Ulm, "A Timed-Token Ring Local Area Network and Its Performance Characteristics, *Proc. Conf. on Local Computer Networks*, IEEE CS Press, Los Alamitos, Calif., Order No. 82CH1800-2 (microfiche only), 1982, pp. 50-56.

8. K.C. Sevcik and M.J. Johnson, "Cycle Time Properties of the FDDI Token Ring Protocol," *IEEE Trans. on Software Eng.*, Vol. 13, No. 3, Mar. 1987, pp. 376-385.

9. G. Agrawal, B. Chen, and W. Zhao, "Local-Synchronous-Capacity Allocation Schemes for Guaranteeing Message Deadlines with the Timed-Token Protocol," *Proc. Infocom*, IEEE CS Press, Los Alamitos, Calif., Order No. 3580-02T, 1993, pp. 186-193.

10. N. Malcolm and W. Zhao, "Guaranteeing Synchronous Messages With Arbitrary Deadline Constraints in an FDDI Network," *Proc. of the Conf. on Local Computer Networks*, IEEE CS Press, Los Alamitos, Calif., Order No. 4510-02T, 1993, pp. 186-195.

11. Q. Zheng and K.G. Shin, "Synchronous Bandwidth Allocation in FDDI Networks," *Proc. ACM Multimedia 93: First Int'l Conf. on Multimedia*, ACM, New York, 1993.

12. N. Malcolm, *Hard Real-Time Communication in High-Speed Networks*, doctoral dissertation, Dept. of Computer Science, Texas A&M Univ., in progress.

Wei Zhao is an associate professor in the Department of Computer Science at Texas A&M University. His research interests include scheduling algorithms, communications protocols, distributed real-time systems, concurrence control in database systems, resource management in operating systems, and their applications.

Currently, Zhao serves as an editor for the *IEEE Transactions on Computers* and as vice chairman of the 14th IEEE International Conference on Distributed Computing Systems (ICDCS) to be held in June 1994. In 1993, he was made a Texas Engineering Experiment Station Fellow by the College of Engineering at Texas A&M University.

Zhao received the Diploma in physics from Shaanxi Normal University, Xian, China, in 1977, and the MS and PhD degrees in computer science from the University of Massachusetts, Amherst, in 1983 and 1986.

Nicholas Malcolm is a doctoral candidate in the Department of Computer Science at Texas A&M University. His research interests include distributed systems, real-time systems, computer networks, and software engineering. In 1989, he received a first-class honours degree from the University of Adelaide. He received an MSc degree in computer science from the University of Calgary in 1991.

Readers can contact the authors at the Department of Computer Science, Texas A&M University, College Station, TX 77843-3112; or e-mail {malcolm, zhao}@cs.tamu.edu.

Advanced Research Projects Agency
Computing Systems Technology Office

High Performance Computing and Communications Symposium

HPC

March 15-18, 1994
Radisson Mark Plaza Hotel
Alexandria, Virginia

The Federal High Performance Computing and Communications (HPCC) Program is creating the tools and technologies that are enabling a revolution throughout science and engineering with broad impact on all of society. Recent accomplishments of ARPA's high performance computing programs and new directions within this inter-agency program will be highlighted, as these will provide the foundation for National-Scale Information Infrastructure. Sessions are being organized on:

High Performance Computing Systems
Scalable Software Technology
High Performance Computing Networking
Grand Challenges of Science and Engineering
Information Infrastructure Technologies and National Challenge Applications

SatoTravel has been designated the travel management company for the HPCC Symposium. Questions concerning registration, hotel, and travel may be emailed or faxed to hpcc-symp@arpa.mil (703) 696-2202. For telephone inquiries, call (800) 347-6338.