Effect of Nodes Reordering on the Schedulability of Real-Time Messages in Timed Token Networks

Sijing Zhang, Xicheng Liu, *Member, IEEE*, and E. Stewart Lee, *Senior Member, IEEE*

***Abstract*—This letter addresses the effect of different ordering patterns of network nodes on the ability to schedule real-time messages in a timed token network where the timed token medium access control protocol is employed. It is found that for any given setting of network parameters, a set of real-time message streams which is unschedulable under one particular nodes-ordering may become schedulable under another different nodes-ordering. Some guidelines for avoiding a possible misjudgement in the schedulability due to inappropriately ordered nodes are discussed.**

***Index Terms*—FDDI networks, real-time communication, schedulability test, synchronous messages, timed token medium access control protocol, timed token networks.**

I. Introduction

THIS letter presents a complemental note on testing the schedulability of real-time traffic in a timed token network (e.g., FDDI [2]) where the timed token medium access control (MAC) protocol [1], [2] is used.

The timed token MAC protocol is capable of supporting real-time communication due to its inherent timing property of bounded medium access time. With this protocol, messages are grouped into two types: *synchronous* and *asynchronous*. Synchronous messages arrive at regular intervals and have delivery time constraints. Asynchronous messages arrive irregularly and have no delivery time constraints. During network initialization time, all nodes negotiate a protocol parameter called the *Target Token Rotation Time* (TTRT) to specify the expected token rotation time. Each node i is assigned a fraction H_i of the TTRT as its *synchronous bandwidth*, which is the maximum time the node is allowed to transmit its synchronous messages every time it receives the token. Whenever node i receives the token, it transmits its synchronous messages (if any) first for a time up to H_i. If the time elapsed since the previous token arrival at node i is less than TTRT (i.e., the token arrived earlier than expected), then node i can send its asynchronous messages to make up the difference.

One of key issues related to guaranteeing the transmission of synchronous messages before their deadlines is schedulability test (i.e., testing whether or not a considered set of synchronous message streams is schedulable under a given setting of network parameters including TTRT and H_i's assigned to all nodes). A set of synchronous message streams is said *schedulable* if every message from each of these streams can be transmitted before its deadline. Zhang *et al.* [3] proposed an optimal schedulability test for a synchronous message set with message deadlines no longer than periods based on an algorithmic method. Their method works for any timed token network running the timed token protocol.

Nodes in a timed token network can be arranged in any arbitrary order in reality. While the arbitrary ordering does not bring a concern for nonreal-time applications, it can, however, affect the network performance of supporting the timely transmission of real-time traffic for certain real-time applications. It is found that under the same setting of network parameters (i.e., TTRT and all H_i's pre-assigned to nodes remain unchanged),[1] a timed token network with a different ordering of nodes may present a quite different performance for guaranteeing real time traffic. A synchronous message set which is unschedulable (i.e., fails to be guaranteed) under one specific ordering (of network nodes) could become schedulable under another different ordering. Based on this finding, a considered synchronous message set can be concluded to be unschedulable, under a given setting of network parameters, only if this is true under every possible ordering pattern of network nodes.

The rest of this letter is organized as follows. In Section II we introduce the background knowledge under which this research is conducted. In Section III we demonstrate, by numerical examples, the effect of reordering network nodes on the schedulability of real-time traffic. In Section IV suggestions of avoiding the possible misjudgement of the schedulability due to inappropriate ordering of nodes is discussed.

II. Preliminary

A. Network and Message Models

The network consists of n nodes forming a *logical* ring. Message transmission is controlled by the timed token protocol. Let τ be the portion of TTRT unavailable for transmitting synchronous messages during one complete token rotation due to inevitable overheads involved. Note that τ may depend on the ordering of nodes (e.g., in a timed bus network).

Each node i has one stream of synchronous messages, denoted as S_i. A total of n synchronous message streams (from all n nodes) forms a synchronous message set. Each S_i is characterized by C_i, P_i and D_i where C_i is the time required to transmit a maximum-size message from S_i; P_i is the minimum message inter-arrival period and D_i is the (*relative*) deadline of messages from S_i (i.e., if a message from S_i arrives at time t, it must be transmitted by $t + D_i$).

Manuscript received April 27, 2000. The associate editor coordinating the review of this letter and approving it for publication was Dr. C. Douligeris.

The authors are with the Centre for Communications Systems Research, University of Cambridge, Cambridge CB2 3DS, U.K. (e-mail: sz207@ccsr.cam.ac.uk; nl225@ccsr.cam.ac.uk; esl1@ccsr.cam.ac.uk).

Publisher Item Identifier S 1089-7798(00)11521-2.

[1]Here we assume that real-time messages (i.e., synchronous messages) originated from a node and the synchronous bandwidth pre-assigned to that node remain unchanged with that same node.

B. Requirements

To guarantee synchronous message deadlines, network parameters must be carefully selected such that they satisfy the following two constraints [3]:

- *Protocol Constraint*: The sum total of the synchronous bandwidths allocated to all nodes does not exceed the available portion of TTRT, i.e., $\sum_{j=1}^{n} H_j \leq \mathrm{TTRT} - \tau$;
- *Deadline Constraint*: Every synchronous message must be transmitted before its deadline. Let R_i be the *worst-case* message response time for a message from S_i (defined as the *longest* possible time interval between the arrival of the message and the completion of its transmission), then the deadline constraint is satisfied *if* and *only if* $R_i \leq D_i$.

Given network parameters, an ordering of network nodes is *feasible* if, under such ordering, both the protocol and deadline constraints hold for the considered synchronous message set. A synchronous message set is *schedulable* under a particular ordering of nodes if the ordering is feasible.

As will be clear, given an assignment of network parameters, the different ordering of network nodes may present different performance for guaranteeing synchronous traffic. So, a synchronous message set is schedulable only if there is at least one feasible ordering among all possible orderings.

III. Effect of Nodes Reordering

In this section we demonstrate, by a numerical example, the effect of different ordering patterns of network nodes on the schedulability of real-time traffic. To simplify the demonstration, let us consider a timed token network (say, FDDI) of three nodes, under two different nodes-ordering patterns as shown in Fig. 1. The considered synchronous message set M with $P_i = D_i$ ($i = 1, 2, 3$) and the synchronous bandwidths (H_i's) produced by the *Normalized Proportional Allocation* (NPA) scheme [4] are listed in Table I. Also, for simplicity, assume $\tau = 0$ and $\mathrm{TTRT} = 50$.

Zhang *et al.* [3] proposed an algorithm to capture the exact value of R_i, which can be used to form an optimal test on schedulability of a synchronous message set in a timed token network of any particular (fixed) ordering. Due to space limitation we shall not describe here their algorithm for searching the value of R_i. Instead we illustrate how to derive the value of R_i in an intuitive way which mirrors, in nature, their method adopted in [3].

Now let us test the schedulability of message set M under two different nodes-ordering patterns (i.e., Ordering A and Ordering B) given in Fig. 1. Clearly, the protocol constraint is met. We now consider how to derive the value of R_i required for testing the deadline constraint.

The worst-case situation for transmission of a synchronous message from S_i, which is used in [3] for derivation of R_i, happens with some time instant t_0 at which the token arrives at node i, subject to the following conditions: i) Before time t_0 there is no traffic on the ring, so the token rotation time is τ and the token arrives at node i, at time t_0, earlier than expected by $(\mathrm{TTRT} - \tau)$; ii) At time t_0 there is not a synchronous message available to be sent in node i, so the *transmission right* is internally passed onto asynchronous traffic within the same node i.

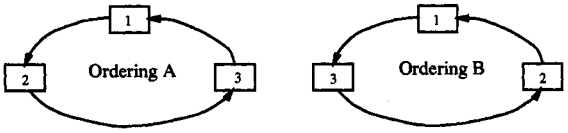

Fig. 1. Two different ordering patterns of the network (e.g., FDDI).

TABLE I
SYNCHRONOUS MESSAGE SET M AND THE H_i'S

S_i	C_i	P_i ($D_i = P_i$)	H_i produced by NPA
S_1	8.6	100	8.6
S_2	15	140	10.7
S_3	40	130	30.7

But, just at that moment a message from S_i arrives; iii) After t_0 every node has enough asynchronous messages (to send) to use up all available asynchronous bandwidth; iv) At time t_0, a message from every stream S_j ($j = 1, 2, \cdots, n; j \neq i$) arrives in node j and another message from the same stream S_j keeps arriving every P_j units of time.

Based on the above worst-case scenario for transmission of a message from S_i and the protocol rules regulating normal ring operations (briefly stated in Section I), We can mimic (simulate) the worst case system behavior for transmitting a message from S_1, S_2 or S_3 by a tracing table, Tables II, III or IV respectively in our example. For a clear tracing, assume that $t_0 = 0$ and the tracing process starts from time $t = t_0 = 0$. Let $T_s(i)$ and $T_a(i)$ represent the time spent at node i in transmitting synchronous and asynchronous messages respectively.

Due to space limitation we explain below only the tracing process for S_2 under Ordering A. In Table III (see the left part under "Ordering A"), at time $t = t_0 = 0$ (Row 1) the token arrives at node 2 earlier than expected by $(\mathrm{TTRT} - \tau) = (50 - 0) = 50$ [by i) above], $T_s(2) = 0$ [by ii)] and $T_a(2) = \mathrm{TTRT} - \tau = 50 - 0 = 50$ [by i) and iii)]. After transmitting its asynchronous messages, node 2 passes the token to its downstream neighbor, i.e., node 3. Since $\tau = 0$, the token arrives at node 3 (Row 2) at time $t = 50$. By iv), a whole message from S_3 should be available when the token arrives at node 3 (at time $t = 50$), thus $T_s(3) = H_3 = 30.7$ (since $H_3 < C_3 = 40$), i.e., node 3 uses up H_3 to transmit the first frame of the whole message (of the length $C_3 = 40$); $T_a(3) = 0$ (since the token arrives in node 3 late at time $t = 50$). Then the token is passed to the next node, i.e., node 1, and it arrives at node 1 at time $80.7 (= 50 + 30.7 + 0)$. Similar to the token's visit to the upstream neighbor of node 1, i.e., node 3 (row 2), in node 1 (row 3) $T_s(1) = 8.6$ (the whole message from S_1, since $C_1 = H_1 = 8.6$); $T_a(1) = 0$. Similarly, we can continue the tracing process for the next few successive token visits to nodes, as shown in the remaining few rows of the table. Note that the token arrives at node 1 *early* at time $t = 109.3$ (Row 6) because the time elapsed since the token's last arrival at node 1 (at $t = 80.7$) is $109.3 - 80.7 = 28.6$, less than $\mathrm{TTRT} = 50$. Thus, $T_s(1) = 8.6$ (another message from S_1 arrived at $t = 100$ because $P_1 = 100$); $T_a(1) = 50 - 28.6 = 21.4$. Thus, the token arrives at node 2 (the last row) at $t = 139 (=$

TABLE II
THE WORST-CASE SYSTEM BEHAVIOR FOR STREAM S_1

Under Ordering A				Under Ordering B			
i	Time t	$T_s(i)$	$T_a(i)$	i	Time t	$T_s(i)$	$T_a(i)$
1	0 (t_0)	0	50	1	0 (t_0)	0	50
2	50	10.7	0	3	50	30.7	0
3	60.7	30.7	0	2	80.7	10.7	0
1	91.4	**8.6**		1	91.4	**8.6**	

TABLE III
THE WORST-CASE SYSTEM BEHAVIOR FOR STREAM S_2

Under Ordering A				Under Ordering B			
i	Time t	$T_s(i)$	$T_a(i)$	i	Time t	$T_s(i)$	$T_a(i)$
2	0 (t_0)	0	50	2	0 (t_0)	0	50
3	50	30.7	0	1	50	8.6	0
1	80.7	8.6	0	3	58.6	30.7	0
2	89.3	10.7	0	2	89.3	10.7	0
3	100	9.3	0	1	100	8.6	0
1	109.3	8.6	21.4	3	108.6	9.3	0
2	139.3	**4.3**		2	117.9	**4.3**	

TABLE IV
THE WORST-CASE SYSTEM BEHAVIOR FOR STREAM S_3

Under Ordering A				Under Ordering B			
i	Time t	$T_s(i)$	$T_a(i)$	i	Time t	$T_s(i)$	$T_a(i)$
3	0 (t_0)	0	50	3	0 (t_0)	0	50
1	50	8.6	0	2	50	10.7	0
2	58.6	10.7	0	1	60.7	8.6	0
3	69.3	30.7	0	3	69.3	30.7	0
1	100	8.6	0	2	100	4.3	0
2	108.6	4.3	0	1	104.3	8.6	6.4
3	112.9	**9.3**		3	119.3	**9.3**	

$109.3 + 8.6 + 21.4$). Since node 2 has transmitted part of the whole message on the previous token visit (at time $t = 89.3$), node 2 only needs to transmit the remaining part of the synchronous message, i.e., $T_s(2) = 15 - 10.7 = 4.3$. So, the transmission of the whole message from S_2 completes, in the worst case, at $t = 139.3 + 4.3 = 143.6$, i.e., 143.6 units of time after the message's arrival, longer than the required deadline of 140 (since $D_2 = P_2 = 140$). Thus, messages from S_2 will miss their deadlines in the worst case, i.e., message set M is unschedulable under Ordering A.

One can conduct a similar tracing process for stream S_2 under Ordering B as shown on the right half of Table III, and for streams S_1 and S_3 under Ordering A and Ordering B, as shown in Tables II and IV. As shown in the right half of Tables II–IV we have that $R_1 = 91.4 + 8.6 = 100 = D_1$; $R_2 = 117.9 + 4.3 = 122.2 < D_2 = 140$; $R_3 = 119.3 + 9.3 = 128.6 < D_3 = 130$. That is, the message set M is schedulable under Ordering B.

The above simple example reveals an fact that a synchronous message set which fails to be guaranteed under one nodes-ordering may become guaranteed under another different nodes-ordering in a timed token network.

IV. DISCUSSION AND FINAL REMARKS

As the sequence of nodes arranged in a timed token network can affect the schedulability of real-time traffic, network nodes need to be arranged cautiously in order to avoid, in reality, a possible misjudgement in schedulability of real-time traffic caused by improper nodes-ordering. The following suggestions may help a better reordering of network nodes in practice for real-time applications.

It is easy to check that for a timed token network of n nodes, there are $(n-1)!$ different nodes-ordering patterns. With the linear algorithm proposed in [3] for searching the exact value of R_i, it is easy to design and run an algorithm to automatically search for all those *feasible* nodes-ordering patterns, if any, or to check the schedulability under any desired specific nodes-ordering. Given network parameters, a considered synchronous message set is deemed unschedulable only if none of $(n-1)!$ possible patterns is feasible.

Theoretically, reallocating application jobs/tasks (which generate real-time traffic) to different nodes can have an equivalent effect to reordering network nodes. The former is preferable in practice if such re-allocation (say, by exchanging some concerned real-time applications among several geographically-separated homogeneous workstations) is possible, because this may lead to an economical solution. The physical migration of nodes and the associated re-cabling can be costly and should be avoided if possible.

Also, in many circumstances, the physical migration of nodes can be avoided using the notion of *logical ring* inherent with the original idea of *timed token* [1]. Nodes in a timed token network can form a logical ring in spite of whether the formed ring is a physical ring network (e.g., FDDI) or not (e.g., IEEE 802.4 token bus). The sequence of nodes arranged in such a logical ring can be arbitrary but must be predefined. Bearing the *logical ring* in mind one can reorder nodes by simply predefining (after finding) a *feasible* order of the token's visits to nodes, rather than physically changing the location and interconnection of nodes. Although larger overheads could be incurred in passing the token following a pre-defined order of nodes, such an option which may bring an effective solution for certain real-time applications remains worth-trying (as the increment of overheads may be small or acceptable). For some real-time applications, the real-time traffic produced may never become schedulable if by only improving a synchronous bandwidth allocation scheme rather than reordering the nodes at the cost of a possibly higher overhead.

We have demonstrated that an improper ordering of nodes may cause actually-schedulable real-time traffic to be misjudged as unschedulable. Some possible solutions are discussed above. This letter presents an important note that complements the previous research on testing the schedulability of real-time traffic in a timed token network.

REFERENCES

[1] R. M. Grow, "A timed token protocol for local area networks," in *Proc. Electro'82, Token Access Protocols*, May 1982, Paper 17/3.
[2] *Fiber Distributed Data Interface (FDDI)—Token Ring Media Access Control (MAC)*, ANSI Standard X3.139, 1987.
[3] S. Zhang, A. Burns, A. Mehaoua, E. S. Lee, and H. Yang, "Testing the schedulability of synchronous traffic for the timed token medium access control protocol," Real-Time Systems, to be published.
[4] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed token medium access control protocol," *IEEE Transactions on Computers*, vol. 43, no. 3, pp. 327–339, March 1994.