Scheduling of the Inter-Dependent Messages in Real-Time Communication

Li Ming
Swiss Federal Institute of Technology, Lausanne
Computer Engineering Department
EPFL-DI-LIT
CH-1015 Lausanne, Switzerland

Abstract

This paper addressed the scheduling problem of inter-dependent message. In our context, a inter-dependent message is defined as a group of messages which are dependent of one another directly or indirectly, e.g., a pair of request-response messages. A inter-dependent message can be treated as a single message. We conclude that scheduling the inter-dependent messages is quite similar to scheduling tasks with external blocking in the context of uniprocessor. We can apply the result obtained in the uniprocessor to the inter-dependent message.

1 Introduction

Scheduling real-time traffic in LAN (local area network) is now the subject of intense studies because of the advent of distributed multimedia applications. Traditionally, the rate monotonic algorithm is chosen to schedule the messages. RMS (rate monotonic scheduling) approach is a static priority one with priorities assigned according to the period. Stronider[1], later Pleinevaux[3], applied RMS approach to the 802.5 token-ring communication protocol. Agrawal et al [2] applied RMS approach to FDDI access protocol. However RMS approach is based on a number of stringent assumptions :

1. Messages are independent in that message arrivals do not depend on the initiation or the completion of transmission requests for other messages.

2. all the messages have deadline equal to period.

3. all the messages are periodic.

The restriction 2 and 3 can be relaxed by applying DMS (deadline monotonic scheduling) approach. DMS approach [8] is a static priority one with priorities assigned according to the deadline. DMS approach can easily accommodate sporadic activities (so long as the inter-arrival times can be bounded in some manner) [6]. In this paper we relax the restriction 1 by applying an extended DMS approach [5].

In real systems, messages are often inter-dependent : e.g., MMS (Manufacturing Message Specification) protocol [13] is based on the client-server model in which messages are sent in pairs, composed of a request from the client and a response from the server. The response arrivals depend on the completion of transmission of the corresponding request. Traditionally, scheduling considers messages as independent of one another; DMS or RMS approach is used. However, a direct application of DMS or RMS approach to the inter-dependent message such as a pair of request-response is not appropriate.

A inter-dependent message is defined as a group of messages which are dependent of one another directly or indirectly. Example : a set of messages a, b, c, d, e; if b depends on a and c depends on b then a, b and c is a inter-dependent message. In this paper we limit a inter-dependent message to be a pair of request-response messages. The client transmits a request and receives a response from the server. The server receives a request from the client and returns a response. We treat such a pair of request-response messages as a single message. In this case, scheduling messages that are paired is quite similar to scheduling tasks with external blocking in the context of uniprocessor. Wellings [5] has extended DMS approach to allow for external blocking, we can apply his result for the inter-dependent message.

This paper is organized as follows. The second section develops the message model. The third section

discusses scheduling of the inter-dependent messages. In the fourth section we give an application example.

2 Message Model

An independent or an inter-dependent message i can be characterized by the following parameters :

T_i — Period of message i. For a non-periodic message, it is minimal inter-arrival times.

D_i — Deadline of message i. Note that we require $D_i \leq T_i$

C_i — The transmission time of message i. For a inter-dependent message, it is the sum of transmission time of the request and the response.

R_i — The worst-case response time of message i. For a independent message, it is the time interval between instant when the message is placed into the send-queue and instant when the message arrives at the recipient node. For a inter-dependent message, it is the time interval between instant when the client places the request into the send-queue and instant when it receives the corresponding response.

X_i — The time interval between instant when the server receives the incoming request and instant when it places the outgoing response into the send-queue. For a independent message $X_i = 0$.

The relationship between these parameters is described in Fig.1. Fig.1 shows the transmission of a inter-dependent message i. Supposed that its period is T_i and its deadline is D_i.

At t=t_0, the client places the request into the send-queue. At $t = t_1$, the server receives the incoming request and process it; then the response is created. At $t = t_2$, the server places the response into the send-queue. At $t = t_3$, the transmission of the response is completed. Obviously, the time interval between t_2 and t_1 is equal to X_i. The time interval between t_3 and t_0 is equal to R_i.

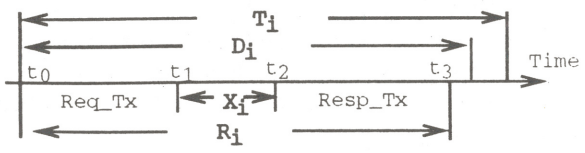

Figure 1: The transmission of a inter-dependent message

3 Scheduling the inter-dependent message

In this section, we simply review the deadline monotonic algorithm and then apply this algorithm to schedule the inter-dependent messages.

3.1 Deadline Monotonic Algorithm

Deadline monotonic priority ordering is similar in concept to rate monotonic priority ordering. Priorities assigned to tasks are inversely proportional to the length of the deadline. Thus, the task with the shortest deadline is assigned the highest priority, the longest deadline task the lowest priority. This priority ordering defaults to a rate monotonic ordering when *period = deadline*.

For DMS approach, schedulability analysis is usually completed by calculating the worst case response time of each task, and to then compare, trivially, each response time to the required deadline (i.e. test to confirm that $R \leq D$). The worst case response time of a task can be found by the following iteration [6] :

$$r_i^{n+1} = c_i + b_i + \sum_{j \in ht(i)} \lceil \frac{r_i^n}{t_j} \rceil c_j \qquad (1)$$

Where :

t_i — The period of task i

r_i — The worst-case response time of task i

c_i — The computation time of task i

b_i — The time task i is delayed by lower priority tasks due to the operation of the priority ceiling protocol [12]. It is also called as internal blocking.

$ht(i)$ — The set of tasks of higher priority than task i and that could preempt task i.

An initial value r_i^0 of 0 is suitable. The iteration is to either converge ($r_i^{n+1} = r_i^n$) or exceed a threshold value.

Wellings [5] has extended Eq. 1 to take external blocking into account. A external blocking time is defined as the time a task spends waiting for an external event occur. When a task externally blocks, the lower priority task may be executed. If there are no lower priority tasks ready to execute when a task externally blocks then the processor is idle. This is the essential difference between external blocking and internal blocking. With internal blocking the processor can not be idle — a higher priority task cannot be blocked unless a lower priority task is able to run. Eq. 1 can be modified to allow for external blocking [5]:

$$r_i^{n+1} = c_i + b_i + x_i + \sum_{j \in ht(i)} \lceil \frac{r_i^n + x_j}{t_j} \rceil c_j \qquad (2)$$

Where :

x_i — The worst case time task i can spend externally blocked in any single invocation

3.2 Scheduling the inter-dependent message

Traditionally, scheduling considers messages as independent of one another, RMS or DMS approach is used. RMS and DMS approach have been intensively studied in past years. In order to apply RMS or DMS approach to the inter-dependent message, we may treat a inter-dependent message as a single message. The priority of a inter-dependent message is assigned according to its deadline; it means that the request and the corresponding response share the same priority. One problem immediately arises : the server cannot know the priority of the request *a priori*, thus the priority of the response cannot be determined. The approach to solve this problem is to transmit the priority of the request to the server. The response inherits the priority of the request.

The transmission of an inter-dependent message i is not continuous in the sense that the transmission of the response do not start immediately after the transmission of the request is completed. There is a time interval X_i (recall that it is the time interval between instant when the server receives the incoming request and instant when this server places the outgoing response into the send-queue) during which the lower priority message may be transmitted. We say that the inter-dependent message i blocks voluntarily for a certain time interval X_i in the transmission. Thus, we see that scheduling a inter-dependent message is quite similar to scheduling a task with external blocking in the context of uniprocessor. We adopt schedulability analysis for task with external blocking and apply it to the analogous problem of scheduling the inter-dependent messages. We establish the analogy between them: the period of a message is akin to the period of a task; the deadline of a message is akin to the deadline of a task; The time interval a message blocks voluntarily in the transmission is akin to external blocking time of a task. The time interval a message is delayed by lower priority message is akin to internal blocking time of a task. Eq.2 can be modified to calculate the worst-case response time of a independent and inter-dependent message.

$$R_i^{n+1} = C_i + B_i + X_i + \sum_{j \in hm(i)} \lceil \frac{R_i^n + X_j}{T_j} \rceil C_j \qquad (3)$$

Where :

B_i — The time message i is delayed by lower priority messages.

$hm(i)$ — The set of messages of higher priority than message i and that could preempt message i

C_i, X_i, R_i and T_i in Eq.3 have been defined in section 2. These parameters except R_i can be known *a priori*. The values of B_i is MAC protocol-dependent (see next section). Calculating X_i requires an analysis of the actual program code; it is beyond our discussion. The interesting reader may read the paper [9].

In the above, we have applied DMS approach to schedule the inter-dependent messages. To support DMS or RMS approach in LAN, a priority-driven MAC protocol must be implemented. Many researches [1] [2] [3] [7] have been made in this domain. Because of the limitation of space, we do not discuss them.

4 Application

In this section we consider an application example. We suppose that LAN is a token bus network. Main service in application layer is MMS [13]. A priority driven protocol [10] is implemented on token bus network.

Table 1, 2 and 3 summarize LAN configuration parameter, MMS confirmed and unconfirmed service requirements developed for this example. Our aim is to calculate the worst-case response time of MMS service (the processing time of the request and the response at the client is not considered). MMS service is scheduled by DMS algorithm, so MMS service's priority is assigned according to its deadline.

Parameters	Value
Number of Nodes	3
Data Rate	10 M_{bps}
Maximum Packet Size	4096 bits

Table 1: `Token Bus Configuration`

To calculate the worst-case response time of a unconfirmed or confirmed service, Eq. 3 may be used.

Ser.	Length (bits)	Period (ms)	Deadline (ms)	Pri.
S_1	25000	60	45	2
S_2	34500	56	50	1

Table 2: **Unconfirmed Service Requirements**

Ser.	Req (bits)	Resp (bits)	Period (ms)	Deadline (ms)	X_i (ms)	Pri.
S_3	10000	10000	55	40	4ms	3
S_4	20000	10500	56	30	4.5ms	4
S_5	1024	3072	54	25	3.5ms	5

Table 3: **Confirmed Service Requirements**

However MAC protocol overhead such as packetization of messages must be considered. We modify Eq. 3 and get Eq. 4 :

$$R_i^{n+1} = C_i + B_i + X_i + \sum_{j \in hm(i)} \lceil \frac{R_i^n + X_j}{T_j} \rceil (C_j + OVH_j) \tag{4}$$

B_i and OVH_i in Eq. 4 can be calculated by Eq. 5 and 6 (see [11]).

If i is a inter-dependent message (confirmed MMS service) then

$$\begin{cases} B_i \leq 2((N-1)P_{max} + N \times C_{tk}) \\ OVH_j = 2C_{ctl} + (C_{enc} + N \times C_{tk})(\lceil \frac{C_{j(req)}}{P_{max} - C_{enc}} \rceil + \lceil \frac{C_{j(resp)}}{P_{max} - C_{enc}} \rceil) \end{cases} \tag{5}$$

If i is a independent message (unconfirmed MMS service) then

$$\begin{cases} B_i \leq (N-1)P_{max} + N \times C_{tk} \\ OVH_j = C_{ctl} + (C_{enc} + N \times C_{tk}) \lceil \frac{C_j}{P_{max} - C_{enc}} \rceil \end{cases} \tag{6}$$

Where :

N — The number of station.

P_{max} — Maximum packet size in units of transmission time.

C_{tk} — Time to transmit a token.

C_{enc} — Time to transmit a packet header and trailer.

$C_{j(req)}$ — Time to transmit the request part of a confirmed service.

$C_{j(resp)}$ — Time to transmit the response part of a confirmed service.

C_{ctl} — Time to transmit a *control packet.*

By Eq. 4, 5 and 6, we can calculate the worst-case response time of each MMS service. Example : for a confirmed service S_3,

$$OVH_3 = 2C_{tk} + (3C_{tk} + C_{enc})(\lceil \frac{C_{3(req)}}{P_{max} - C_{enc}} \rceil + \lceil \frac{C_{3(resp)}}{P_{max} - C_{enc}} \rceil)$$

$$B_3 \leq 2(2P_{max} + 3C_{tk})$$

$$R_3^{n+1} = C_3 + B_3 + OVH_3 + X_3 + \sum_{j \in hm(3)} \lceil \frac{R_3^n + X_j}{T_j} \rceil C_j$$

$$R_3^0 = 0; \quad R_3^1 = 12.05ms; \quad R_3^2 = 12.05ms;$$

$$because : R_3^2 = R_3^1 \quad therefore : R_3 = 12.05ms;$$

The result for other service is shown in Table 4

Ser.	Worst-case response time
S_1	10.14ms
S_2	14.10ms
S_3	12.05ms
S_4	9.88ms
S_5	5.65ms

Table 4: **Worst-Case Response Time**

5 Conclusion

In this paper we study the problem of scheduling of the inter-dependent message. We treat a inter-dependent message as a single message. We conclude that scheduling the inter-dependent messages is quite similar to scheduling tasks with external blocking in the context of uniprocessor. We adopt schedulability analysis for task with external blocking and apply it to the analogous problem of scheduling of the inter-dependent messages. From our best knowledge, no paper addressed the scheduling problem of the inter-dependent messages.

References

[1] Strosnider, J., Marchok, T., Lehoczky, J. "Advanced real-time scheduling using the IEEE 802.5 token ring", *Proc. of IEEE Real-Time Systems Symposium* Dec. 1988. pp. 42-52

[2] Agrawal, G., Chen, B., Zhao, W., and Davari, S., "Architecture Impact of FDDI Network on Scheduling Hard Real Time Traffic", *Workshop on Architectural Aspects of Real Time Systems* (December 1991)

[3] P.Pleinevaux "An Improved Hard Real-Time Scheduling for the IEEE 802.5", *The Journal of Real-Time Systems*, 4, 1992, pp. 99-122

[4] Lehoczky, J.P and Sha, L. "Performance of Real-Time Bus Scheduling Algorithms", *ACM Performance Evaluation Review, Special Issue*, Vol. 14,No. 1, May 1986

[5] A.Wellings, M.Richardson, A.Burns, N.Audsley, K.Tindell , "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling", Department of Computer Science, University of York.

[6] Ken Tindell, "Analysis of Hard Real-Time Communications", Department of Computer Science, University of York.

[7] Shirish, "Scheduling Real-Time Traffic in packet switched Networks", *A Dissertation at Carnegie Mellon University* 1992..

[8] Neil C. Audsley, "Deadline Monotonic Scheduling", Department of Computer Science, University of York.

[9] Neil C. Audsley, A.Burns, "Real-Time System Scheduling", Department of Computer Science, University of York.

[10] M.Li, "A priority-based protocol for real-time token passing networks", *12th Conf. on European Fibre Optic Commu. and Networks*, Heidelberg, June 21-24, 1994

[11] M.Li, "Real-Time MAC Protocol in local area network", Internal Technique Report, EPFL-DI-LIT 1994

[12] R. Rajkumar, L. Sha and J.P. Lehoczky, "Real-Time Synchronization Protocols for Multiprocessors", Department of Computer Science, Carnegie-Mellon University (April 1988).

[13] ISO/IEC 9506-1 *"Manufacturing Message Specification"* 1900-10-15 First edition