A Timing Constraint Search Technique for a TMO based Real-time Process

Yoon-Seok Jeong*, Tae-Wan Kim*, Sun Young Han*, and Chun-Hyon Chang*

Abstract: Finding a valid timing constraint is one of the most important issues in the real-time monitoring area. To get the valid timing constraint, a developer executes a real-time process and changes the constraint on a regular basis. This is an exhaustive and time-consuming process. To improve this approach, we propose a timing constraint search technique. This technique uses two load models and one condition proposed in this paper to support the developer in determining the valid timing constraint range in an easy and systematic manner.

Keywords: Timing Constraint Search Technique, TMO Real-Time Process Load Model, Network Load Model

1. Introduction

A real-time system should guarantee that a service request is treated within a timing constraint [5]. The given timing constraint is a required baseline to see if the real-time system operates successfully. Many studies related to the timing constraint have been made based on the assumption that the timing constraint is valid [1, 2, 5, 7, and 8]. In the view of a developer, the timing constraint is not known, however [3, 10]. Based on trial and error, the developer executes a real-time process and changes the constraint repeatedly until he finds the valid timing constraint. This is very exhaustive.

To resolve this problem, we propose a timing constraint search technique. This supports a developer in determining easily the valid timing constraint range with just one execution of a real-time process. This technique calculates the load for a real-time process and network, and then derives the valid timing constraint range for the real-time process using the BJB (Balanced Job Bound) condition.

2. Related Work

2.1 Structure of TMO

In this paper, we focus on a real-time process based on a real-time programming model, TMO (Time-triggered Message-triggered Mode) [7, 8]. The TMO model provides the concrete syntax and semantic for the reliable design of the real-time system. Fig. 1 shows the structure of the TMO.

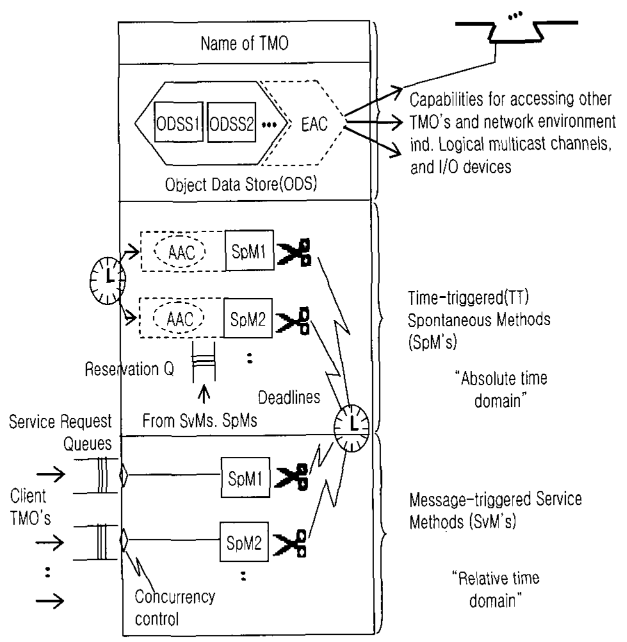


Fig. 1. The Basic Structure of the TMO

- EAC (Environment Access Capability): List of gates to objects, providing efficient call-paths to remote object methods, logical communication channels, and I/O device interfaces.
- SpM (Spontaneous Method): A time-triggered method which runs in a periodic manner.
- SvM (Service Method): A message-triggered method which responds to external service requests.
- ODS (Object Data Store): Storage of properties and states the TMO object. An ODS is being accessed by the SpM and the SvM, and could not be accessed simultaneously.

Manuscript received February 8, 2006; accepted March 3, 2006.
This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment)
Corresponding Author: Tae-Wan Kim
* Department of Computer Science and Engineering, Konkuk University, {ysjeong, twkim, syhan, chchang}@konkuk.ac.kr

2.2 Single Queue Model

A queue model is a powerful abstraction for modeling a system[4]. It is especially effective in modeling a computer communication system. In this paper, we use a queue model for modeling the TMO model-based system in the distributed environment (referred to hereinafter as the 'TMO system'). The TMO systems communicate bilaterally through the channel. An SpM of the TMO system generates and sends communication calls using the channel, while an SvM of the other system receives the calls from it. Thus, we can use a single queue model to represent the channel between two TMO systems. Fig. 2 shows the structure of a single queue model. Q is the length of services waiting for processing in the queue, and the combination of λ and μ. λ is the mean arrival rate of the services inserted in the queue. Then, μ is the mean service rate. In the TMO model, λ and μ could be understood as the mean call rate of the SpM and the mean service rate of the SvM, respectively.

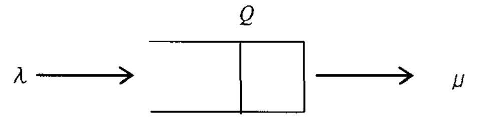


Fig. 2. The Structure of the Single Queue Model

3. Timing Constraint Search Technique

3.1 Load Model for a TMO-based Real-Time Process

Fig. 3 depicts the structure of the TMO system in the single-node environment. The single-node means one TMO system which runs all the real-time processes without communication with any other systems. But the SpM in the TMO system can call the SvM in the same system. It means the load caused by the execution of the real-time process using the single queue model can be calculated.

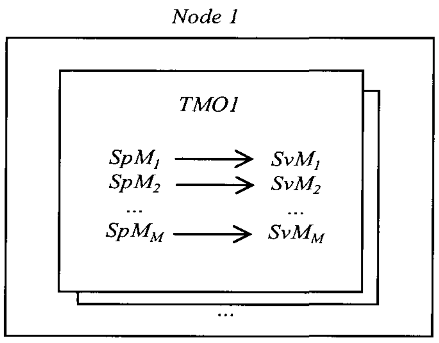


Fig. 3. A TMO System in the Single-Node

Here, we assume that there is no load caused by any other applications running in the system. Let Sp_{ij} be the j-th service time of the SpM_i, and Vp_i be the execution number in the time frame t. Then, the mean service time of the SpM_i is

$$Sp_i = \frac{1}{Vp_i}\sum_{j=1}^{Vp_i} Sp_{ij}. \quad (1)$$

Let Q_i^T be the load caused by the execution of the SpM_i. According to Reiser and Lavenberg, the response time of the SpM_i can be denoted as below.

$$Rp_i = Sp_i\left(1+Q_i^T\right). \quad (2)$$

Thus, based on the operation law, the overall response time of the TMO system is

$$R = \sum_{i=1}^{M} Rp_i Vp_i \quad (3)$$

where M is the number of the SpM. Then, the throughput of the TMO system, X, and the load caused by the SpM_i, Q_i^T are calculated by the equation (4) and (5) based on the operation law.

$$X = \frac{N}{Z+R} \quad (4)$$

$$Q_i^T = XVp_i Rp_i \quad (5)$$

Here, Z is the buffer time. It means the reserved period in which to prevent the execution of the real-time process from conflicting with the next execution. If Cp_i is the scheduled period of the SpM_i, and Pp_i is its deadline, then Z is

$$Z = \frac{1}{M}\sum_{i=1}^{M}(Pp_i - Cp_i). \quad (6)$$

Using equations (1) to (6), we can obtain the total load of the TMO system in the single-node environment finally.

$$Q^T = \sum_{i=1}^{M} Q_i^T \quad (7)$$

3.2 Network Load Model

Fig. 4 represents the structure of the TMO systems in the multi-node environment. The two systems are independent, but they are connected to each other by the channel in the network.

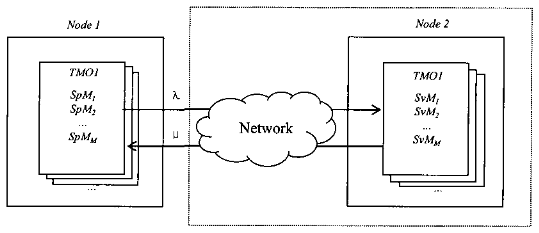


Fig. 4. A TMO System in the Multi-Node

From the viewpoint of the queue model, we assume that the dotted area in Fig. 4 is a queue. Then, λ is the mean arrival rate of the calls from system 1 to the queue. Let t and Pp_i be the time interval and the period of the SpM_i, respectively. The mean arrival rate of the calls is calculated by the equation (8).

$$\lambda = \frac{1}{M}\sum_{i=1}^{M}\frac{t}{Pp_i} \tag{8}$$

Then, we can also calculate λ of the TMO system. Using equation (1), the mean service time of the TMO system is denoted as follows.

$$\mu = \frac{1}{M}\sum_{i=1}^{M}\frac{t}{Sp_i} \tag{9}$$

Note that μ could be influenced by the network state and the performance of the TMO system 2. By considering these influences, equations (2) and (9) are used as follows.

$$\mu = \frac{1}{M}\sum_{i=1}^{M}\frac{t}{\left(Rp_i/(1+Q)\right)} \tag{10}$$

Q is the load caused by the network and the TMO system 2. Thus, according to the general response law, Q must be divided into the network load, Q^N, and the load made by the system 2, Q^T. Here, Q can be denoted as below.

$$Q = Q^N + Q^T. \tag{11}$$

Now, the queue area should be narrowed down to calculate Q^N and Q^T. Fig. 5 can be modeled using the M/M/1 queue because λ and μ follows the exponential distribution.

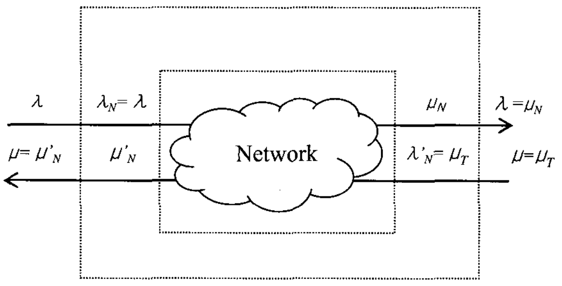


Fig. 5. A Network based on a Single Queue Model

Using M/M/1 queue model, the length of Q is given by equation (12).

$$Q = E[n] = \rho + \frac{\rho^2}{2(1-\rho)} \tag{12}$$

Here, ρ means the relation between λ and μ, and can be derived by the equation below.

$$\rho = \frac{\lambda}{\mu} \tag{13}$$

As shown in Fig. 5, let λ and μ from the system 1 to 2 be λ_N and μ_N (called hereinafter 'input direction'). In contrast, let λ and μ from the system 2 to 1 be λ'_N and μ'_N (called hereinafter 'output direction'). ρ in the input direction can be denoted as follows.

$$\rho_N = \frac{\lambda_N}{\mu_N} = \frac{\lambda}{\mu_N} \tag{14}$$

Using the equation (12) and (14), the load in the input direction is given by equation (15).

$$Q_{in}^N = E_{in}[n] = \rho_N + \frac{\rho_N^{\,2}}{2(1-\rho_N)} \tag{15}$$

On the other hand, the load in the output direction can be derived by equations (16) and (17).

$$\rho'_N = \frac{\lambda'_N}{\mu'_N} = \frac{\mu_T}{\mu'_N} \tag{16}$$

$$Q_{out}^N = E_{out}[n] = \rho'_N + \frac{\rho'^{\,2}_N}{2(1-\rho'_N)} \tag{17}$$

In Fig. 5, we divided the network load into the input and output directions because λ and μ are different. In conclusion, using equations (15) and (17), the equation (11) should be converted as follows.

$$Q = (Q_{in}^N + Q_{out}^N) + Q^T \tag{18}$$

Finally, we should calculate Q^T. Q^T is the load caused by the execution of the SvMs in system 2. Before calculating Q^T, we assume that the load made by SpM_i and the one by SvM_i are the same. This assumption will be true because the SvM takes most of the time of the SpM's execution. Based on this assumption, we can use the load model for the TMO real-time process to calculate Q^T. Using that model, we calculate the load of the SvM_i, Q_i^T, and Q^T. If we obtain Q^T, the total load in the multi-node environment, Q can finally be determined.

3.3 Condition for Searching the Timing Constraint Range

Using Q derived in sections 3.1 and 3.2, we can determine the performance range of the real-time process. According to the operation law,

$$Q = RX \tag{19}$$

and based on equation (11), equation (19) can be changed into equation (20) or (21).

$$(Q^T + Q^N) = RX \qquad (20)$$

$$R = \frac{(Q^T + Q^N)}{X} \qquad (21)$$

Here, we will derive the performance range using the BJB from the bottleneck analysis. The BJB constitutes the upper and lower boundaries of the performance (i.e. response time) [4]. Let D and D_{max} be the total demand time and the maximum demand time of the real-time process. Then, the lower boundary of R, R_b is denoted as follows.

$$R_b = \max\left\{ ND_{\max} - Z, D + (N-1)D_{avg}\frac{D}{D+Z} \right\} \qquad (22)$$

On the other hand, the upper boundary of R, R_u is given by the equation (23).

$$R_u = D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z} \qquad (23)$$

Thus, R should be in the range between R_u and R_b. If R satisfies the condition (24), it is a valid performance; in other words, a valid execution time.

$$R_b \le R \le R_u \qquad (24)$$

In conclusion, a developer can be sure that he gets the valid timing constraint if he sets it to R between R_u and R_b.

4. Experiments

For the experiments, we used the LTMOS middleware [6, 9], and we wrote the TMO applications and installed them on the two systems.

Figs. 6 to 8 show the results of the experiments on the TMO based real-time process. For the experiments, we measured the queue length and the response time as increasing the number of TMO applications.

Fig. 6 depicts the queue length obtained from the execution of SpM1 and SpM2 in the multi-node. In fact, the queue length in the multi-node is greater than that in the single node. In addition, the queue length in the multi-node increases steeply while that in the single-node rises gently. This can be explained as the result of the influence of the network load.

Fig. 7 shows the response time of SpM1 and SpM2 in the multi-node. We find that the response time follows the load pattern shown in Fig. 6. Fig. 8 shows the BJB to determine the valid range of the response time. The response time exists between the lower and the upper value of BJB. In other words, if a developer sets the timing constraint to the value between the lower boundary and the upper boundary of the BJB, he can be sure that it will be the valid timing constraint.

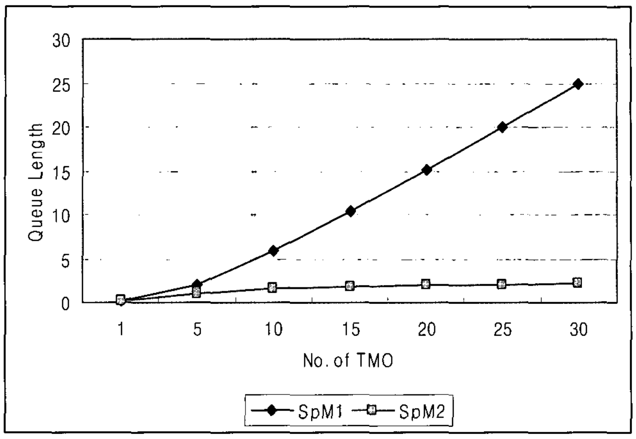


Fig. 6. Queue Length (Multi-Node)

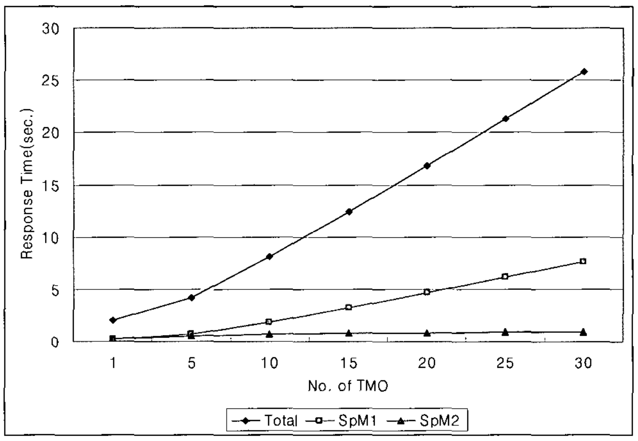


Fig. 7. Response Time

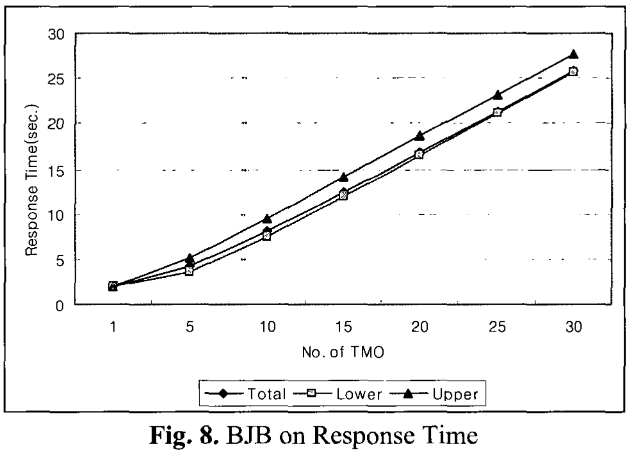


Fig. 8. BJB on Response Time

5. Conclusion

In this paper, we proposed a timing constraint search technique to determine the valid range of the timing constraint. For this technique, we proposed two load models to derive the valid boundary of the timing constraint of the real-time process: the load model for the TMO based real-time process is designed to calculate the load caused by the real-time process, while the one for the network is made to find the network load in the multi-node. Using these models, we could set the conditions based on the BJB for determining the valid range of the timing constraint. This technique is more convenient and systematic compared to the conventional trial and error approach.

Based on the results of this paper, we will implement an analysis tool for the TMO systems which can determine the recommended valid timing constraint.

Reference

[1] S. A. Brandt, "Performance Analysis of Soft Real-Time Systems", 1997 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), 1997.

[2] S.A. Brandt, "Performance Analysis of Dynamic Soft Real-time Systems", Proceedings of the IEEE Performance, Computing, and Communications, 2001.

[3] M. V. Devarakonda, R. K. Iyer, "Predictability of Process Resource Usage: A Measurement-Based Study on UNIX", IEEE Transactions on Software Engineering, 1989.

[4] R. Jain, "the Art of Computer Systems Performance Analysis", John Wiley & Sons, Inc., 1991.

[5] Yoon-Seok Jeong, Tae Wan Kim, Chun Hyon Chang, "Design and Implementation of a Run-time TMO Monitor on LTMOS", Proceedings of the Embedded Systems and Applications, 2003.

[6] J. G. Kim, S. Y. Cho, LTMOS: "An Execution engine for TMO-Based Real-Time Distributed Objects", Proceedings of the PDPTA, 2000.

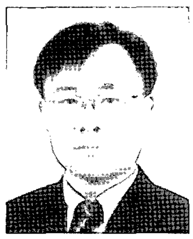
[7] K. Kim, et al., "A Timeliness-Guaranteed Kernel Model: DREAM Kernel and Implementation Techniques", RTCSA, 1995.

[8] K.H. Kim, H. A. Kopetz, "Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", Proceedings of the 18th IEEE Computer Software & Applications Conference, 1994.

[9] S.H. Park, "LTMOS(LinuxTMO System)'s Manual", HUFS, 2000.

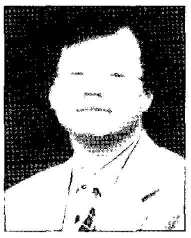
[10] D. Rosu et al., "On Adaptive Resource Allocation for Complex Real-time Applications", Proceedings of the 18th IEEE Real-Time Systems Symposium, 1997.

Yoon-Seok Jeong
He received a M.S. degree in Computer Science and Engineering from Konkuk University in 2000. He has been on a Ph.D. course in Computer Science and Engineering at Konkuk University since 2000. His research interests include Real-time Monitoring in Distributed Environments, Real-time Middleware, and Dynamic Analysis.



Tae-Wan Kim
He received a M.S. degree in Computer Science and Engineering from Konkuk University in 1996. He worked at Hyundai Heavy Industry Co., as a researcher from 1996 to 2001. He has been a lecture professor with the Department of Computer Science and Engineering, Konkuk University, since 2004. His research interests are in the area of Programming Language, Real-time Programming, Monitoring and Diagnosing Systems for Industrial Devices.

Sun Young Han
He received M.S. and Ph.D. degrees in Computer Science from KAIST in 1979 and 1988 respectively. Since 1984, he has been a professor with the Department of Computer Science and Engineering, Konkuk University. His research interests include Computer Networks, Mobile IPv6, and Real-time Communication Systems.



Chun-Hyon Chang
He received M.S. and Ph.D. degrees in Computer Science from KAIST in 1979 and 1985, respectively. Since 1984, he has been a professor with the Department of Computer Science and Engineering, Konkuk University. His research interests include Programming Languages and Compilers, and Real-time Programming and Systems.