Reduction of Useless Services with Timing Constraints

Chih-Cheng Lien Chien-Chiao Yang

Dept. of Electronic Engineering
National Taiwan Institute of Technology
Taipei 10772, Taiwan, R.O.C.

Abstract

A distributed computing system consists of objects and messages which are transmitted between objects. An object can request other objects to do some services by passing messages through a communication system. Usually, the services are done if the message transmission in the communication system is reliable. However, in some applications with timing constraints, a service may be of no value if it violates some constraints. We thus can allow the object to send the constraint within the message to avoid this useless service. In validating the satisfaction of a service for a constraint, using the current elapsed time and using the predicted elapsed time are discussed. The relative accuracy of the prediction in several types of message transmissions is also explained.

Key Words: Message, performance satisfaction, timing constraints.

1 Introduction

A distributed computing system may be viewed as many functional nodes which can be linked by a communication system and are used to perform computing work [6]. We will treat these nodes in terms of objects. Among these, some messages could be passed through the communication system in order to request the services provided by other objects [2].

In general, when one object (Source) requests a service provided by the other object, it send a message to the desired object (Destination) through a communication system. After the destination receives the message, it renders the service. A graphical representation of this sequence in which the arrows denote message transmissions is as follows:

Source → Communication system → Destination.

Thus, for message transmissions and providing a service, an object can generally have two parts: a controller and an implementor [5]. Usually, the controller sends and receives the messages. The implementor does the services allowed in the object. Hence, a message usually includes the information of its source, destination, and content. The content of a message includes some parameters to invoke a functional service.

Assume that the destination will receive the sending message and the desired function is available in the destination: the service will thus be done. However, in real-time systems, the service may be no value because of denying some timing constraints [7]. Thus, it is a useless service, and doing it is not necessary. Therefore, besiding to send and receive message, the controller can check the satisfaction of doing a request with the constraints of the desired service. Thus, in order to check the satisfaction to timing constraints, a message is extended with timing constraints, and can be denoted in a four tuple $< S, D, M, C >$; where S, D, M, C are the sources, destinations, contents of the messages, and constraints, respectively.

In the next section, using the controller to discover the useless service by checking the satisfaction of a service for the constraints in a message is discussed. Section III describes the prediction of the elapsed time for the services of a request in order to check the satisfaction of a service for a constraint. Several types of service may exist in the communication systems. For example, the destination is usually requested to reply with messages back to the source after doing the requested service. Thus, the variable accuracy of prediction in these types of service is also mentioned. Finally, in section IV, the key concept of this paper is concluded and future research is outlined.

0-8186-2755-7/92 $03.00 © 1992 IEEE

2 Sending timing constraints to reduce useless services

In this section, how to discover useless services will first be disscused. During this discovering process, we have to send the constraints in the message through a communication system. Thus, reducing services need for sending constraints is also mentioned.

2.1 Discovering and reducing useless services

From the source to the destination, a message may go through objects and yield a path, e.g., $< O_s \rightarrow O_{i1} \rightarrow \cdots \rightarrow O_{in} \rightarrow O_d >$, where O_s, O_d, and O_i are the source, destination, and internal objects in the path, respectively. In addition, the symbols $<, >$ are used to enclose the description of a path. In the above path, O_s sends a message to the communication system; further, $O_{i1}, \ldots, O_{in}$ receive and then send the message orderly; finally, O_d receives the message. Then, O_d does the desired service f_i, and makes some responses back to O_s if it is necessary. In software, we can define the constructs send(), receive(), is(D), and do() in an object to send, to receive the message, to check whether it is the destination or not, and to perform a functional service. Hence the path for servicing a request becomes <send(ME_i)$_{O_s}$, receive(ME_i)$_{O_{i1}}$, send(ME_i)$_{O_{i1}}$, ..., send(ME_i)$_{O_{in}}$, receive(ME_i)$_{O_d}$, do(f_i)$_{O_d}$ >; where subscripts of functions denote the objects which perform the functions.

Two types of service in the above path are: communication; and functional service. The communication service is to transmit messages and is implemented by constructs such as send(), receive(), is(D). Performing the desired service at the destination is the functional service. From the viewpoint of time, a timing constraint may be denied with the lack of satisfaction for the constraint in the communication service or in the functional service. Hence, the lack of satisfaction may be discovered in the source, the communication system, or the destination. Thus, in the path, from requesting the service to doing the service, the satisfaction of timing constraints should be checked. Therefore, every object in the path may be a checkpoint for the checking of satisfaction.

In real-time systems, we usually have minimum and maximum timing constraints. In a minimum constraint, the elapsed time for doing a request is not allowed to be less than the quantity specified in the constraint. On the other hand, the elapsed time for servicing a request must not be larger than the specified quantity in the maximum constraint. If the elapsed time violates the specification in the timing constraint, the service will not be useful.

```
receive(ME_i);
check the satisfaction for the
    constraint in ME_i;
if the constraint is satisfied
   if is(D)
         do(f_i);
         send(ME_j), if it is necessary;
   else
         send(ME_i);
else
   discard this message;
   send(ME_k), if it is necessary;
```

Fig. 1: Activities of a controller when receiving a message.

Therefore, every controller of the objects, in our approach, may have the following procedure to deal with the above activities. When it receives a message, it checks whether the elapsed time satisfies the timing constraint or not. Then, if the result of the check is satisfied, it checks whether the object is a destination or not. If it is not a destination, it transmits the message to the communication system; otherwise, it gets rid of this message. If it is a destination, the request service is allowed, then performs the service. Finally, the controller stores some responses in the object to record whether and why the service was done or not. The destination may acknowledge or prompt the result message, e.g., ME_j or ME_k, to the source if it is necessary. Thus, a general algorithm to handle the receiving message in the controller of an object is given in Fig. 1.

2.2 Discussion

From the above explanation, if we send the timing constraint within the message, we may add the communication service for the constraint and decrease the useless communication and functional services. Thus, if the elapsed time for the useless service is larger than for the communication service for the constraint, it is acceptable. Usually, the specification of a timing constraint is so simple that its load for the communication service is low [4]. In addition, this service may be discarded during the message transmission. Furthermore, this discarding may happen in a lot of messages if the communication system broadcasts a

message to a lot of objects [8]. Thus, the elapsed time of the useless functional and communication service is usually larger than of the communication service for the constraint. Therefore, it seems to be acceptable. To increase the acceptability of sending constraints within the message by detecting more useless functional services and reducing communication services for the constraint is discussed in the following.

Usually, we can use the current elapsed time or the predicted elapsed time to check the satisfaction of a service for timing constraints. In the first, as shown in Fig. 1, when every object receives a message, it uses the current elapsed time of the message transmission to check the satisfaction of a service for the constraints. This means that the current elapsed time is compared with the time specified in the constraint in order to decide whether the elapsed time still satisfies the constraint or not. In the second, we can predict the elapsed time for completing a request, then use the predicted result to check the satisfaction of a service for the constraint.

Using the current elapsed time to check the satisfaction of servicing a request for the constraints allows every object to be a checkpoint. This may increase the activities in servicing a request such that the performance of doing such a request is lowered in efficiency. However, we can use the predicted elapsed time to check the satisfaction in order to avoid too many checkpoints. This may improve the performance of doing a request and increase the acceptability of sending constraints within a message.

Fig. 2 is given to illustrate the concept of using the predicted elapsed time; where p_i are the possible checkpoints. The points p_1 and p_3 are used to check the satisfaction when sending the message to the communication system. The points p_2 and p_4 are used to check the satisfaction at receiving a message from the communication system. The point p_5 is used to check the satisfaction before doing the desired service f_i, and p_6 is used to check the satisfaction after doing the desired service. The procedure in Fig. 1 only uses the elapsed time to check the satisfaction of a service as receiving a message such that the detection of useless service is limited.

Now, for example, let p_5 be a checkpoint, then the procedure to handle a receiving message is given in Fig. 3. The bold part is different from the procedure in Fig. 1. Now, this procedure can check the useless functional service such that the acceptability of sending the constraints is improved. As another example, let p_3 be a check point, then the procedure to handle a receiving message is given as in Fig. 4. The bold

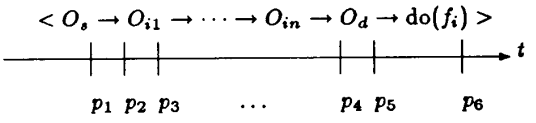

Fig. 2. Possible checkpoints for checking satisfaction.

```
receive(ME_i);
check the satisfaction for the
    constraint in ME_i;
if the result is satisfied
   if is(D) and
    the expect time of functional
    service is satisfied
        do(f_i);
        send(ME_j), if it is necessary;
   else
        send(ME_i);
else
   discard this message;
   send(ME_k), if it is necessary;
```

Fig. 3: Activities to check for useless functional service.

part is different from the procedure in Fig. 3. Consequently, in the sequence $O_{i2}, \ldots, O_{in}, O_d$, it need not send the constraint and need not check the satisfaction for the constraints if the constraint is predicted to be satisfied in this sequence at object O_{i1}. The handling procedure of $O_{i2}, \ldots, O_{in}, O_d$ after receiving a message is simplified in Fig. 5. Thus, for reducing the load of the constraints transmission and increasing the acceptability of sending the constraint, using the predicting elapsed time is preferable to using the current elapsed time.

Though the maximum improvement in the load of the constraint transmission is to predict the satisfaction of a service for the constraint in the source object, sometimes it is not practical. For example, the source may not be available for doing this prediction. This is widely possible in a distributed system. Also, if every object has the ability to predict, too many copies of the same function increase the workload of the objects in the distributed computing system. Finally, the accuracy of the result of the prediction may be not acceptable in some software requirements. Thus,

```
receive(ME_i);
check the satisfaction for the
    constraint in ME_i;
if the result is satisfied
   if is(D) and the expect time of
        functional service is satisfied
          do(f_i);
          send(ME_j), if it is necessary;
   else if the predicted elapsed
        time is satisfied
          send(ME_i);
else
   discard this message;
   send(ME_k), if it is necessary;
```

Fig. 4: Activities to check for useless service before sending a message.

```
receive(ME_i);
if is(D)
    do(f_i);
    send(ME_j), if it is necessary;
else
    send(ME_i);
```

Fig. 5: The most simplified activities of receiving a message.

the sending constraint is necessary in the above cases.

Even though the communication service for constraints may deny the requirements in a real system, we can include this service during system development. Then, the constraint is passed to a communication system in order to search for the approximate solution of the system. For example, if many objects provide the requested service, based upon the responses we can enclose an area of message transmission (one message transmission or broadcast) to tune timing constraint. One of concern, we can relax the constraint to enclose more objects which provide the requested service in order to improve the functional reliability. On the other hand, a more restricted constraint encourages improvement of performance and reducing the workload of message transmission. Finally, before the delivery of this system, the implementation of sending timing constraints can be pick up and out. Thus, our approach using constraints in a communication system may be applied to system development at least.

3 Predicting the elapsed time for servicing a request

In the last section, we concluded that using the predicted elapsed time is preferable in checking the satisfaction of servicing a request for the constraints rather than using the current elapsed time. Thus, we introduce shortly the techniques for predicting the elapsed time about doing services involved by a request in this section. Also, the levels of the accuracy of the prediction in some types of communication service are explained.

3.1 Introducing the technique for the prediction

We can use the expected elapsed time for the services involved in a request to predict the elapsed time for doing the request [9]. The expected elapsed time of a communication system can be predicted from the performance of the communication system [8]. On the other hand, the expected elapsed time for the functional service can be predicted using the efficiency of the implementation of this function [1]. Then, the elapsed time of doing a request service f_i can be generally expressed and predicted by the following polynomial,

$$
\begin{aligned}
&T(\text{request}) \\
&= T(\text{communication service}) \\
&\quad + T(\text{functional service}) \\
&= T(ME_i \text{ in } O_s \rightarrow O_{i1} \rightarrow \cdots \rightarrow O_{in} \rightarrow O_d) \\
&\quad + T(f_i) \\
&= T(\text{send}(ME_i)_{O_s}, \ldots, \text{receive}(ME_i)_{O_d}) \\
&\quad + T(f_i) \\
&= T(\text{send}(ME_i)) + \ldots + T(\text{receive}(ME_i)) \\
&\quad + T(f_i) \qquad (1)
\end{aligned}
$$

In (1), the elapsed time caused by message transmission in transmission line and by interrupts among the functions is discarded.

Particularly, if the elapsed time of sending and receiving a message and checking the destination in the objects, respectively, are the same and $n = 3$, (1) becomes the following,

$$4 \times (T(\text{send}()) + T(\text{receive}()) + T(is(D))) + T(f_i).$$

Now, if the quantities of the elapsed time for doing send(), receive(), is(), and f_i are in numerical form, the above polynomial can be represented in numerical form. Then, the quantity can be used to compare with the quantity specified in a constraint in order to check the satisfaction of the elapsed time for the constraint.

3.2 Prediction in several types of the communication service

In general, the communication and functional services are involved in doing a request. The prediction in the functional service can be done by studying the efficiency of programs in a uniprocessor environment. This is widely known in computer science, e.g., [1], and we do not need further comment. On the other hand, there are usually many types of services in communication systems. The accuracy of the prediction in these types of services may be varied. A short discussion of this is undertaken in the following.

If a request needs a response from the server after doing the service, we need a bidirectional service of the communication. If the request only initiates a service and does not need a response after doing the service, then the unidirectional service is enough. The transmission of these services may go through a communication link directly or assist with a mailbox indirectly. The service provided by a communication link may be connectionless or connection-oriented. In the connection-oriented service, a virtual circuit is built; in the connectionless service, no circuit is held. The following explains the accuracy of the prediction in their types of service.

Let the symbolic terms T_c, T_f, T_m, T_r be used to denote the elapsed time for sending messages in communication system, for doing functional service, for processing messages in a mailbox, and for performing the services involved in a request, respectively. Usually, due to the techniques applied to the prediction, T_c, T_f, T_m, T_r have some inaccuracies [3].

In the bidirectional service, the source sends the request to the destination through the communication system. After performing the requested service in the destination, it responds with some message back to the source through the communication system. In the unidirectional service, the source only sends the request to the destination, and does not require responses. After the request is performed in the destination, the request is completely serviced. Using $T_r(\text{type})$ to denote the prediction of a type of service, we get the following equations.

$$
\begin{aligned}
T_r(\text{bidirection}) &= T_c + T_f + T_c, \\
T_r(\text{unidirection}) &= T_c + T_f.
\end{aligned}
$$

Because one T_c has few inaccuracy factors than two T_c have, the prediction of T_r in the unidirectional service is more accurate than in the bidirectional service.

In the following, the unidirectional service is defaulted. Now, we look at the elapsed time using the mailbox or not.

$$
\begin{aligned}
T_r(\text{mailbox}) &= T_c + T_m + T_f, \\
T_r(\text{no mailbox}) &= T_c + T_f.
\end{aligned}
$$

Since T_m has some inaccuracies, the prediction in the service with the mailbox is less accurate than without the mailbox.

In the connection-oriented service, a virtual circuit is built such that message transmission is more controllable than in the connectionless service. Thus, the prediction in the connection-oriented service is more reliable than in the connectionless service. That is, the prediction is more accurate in the circuit rather than in no circuit.

4 Conclusion

This paper recognizes that some services for a request initialized from an object may be useless if they deny the constraints. Thus, sending the constraints within the message is proposed in order to reduce the

useless services. This approach can be used to develop the system if the load of the communication service for the constraints is less than the load of doing the useless services. It also assists time-critical systems development with investigating the satisfaction of time-critical services for the constraint.

Sometimes the workload for transmitting the constraints can be reduced using the techniques of prediction. Especially, if the prediction is accurate and reliable in an object, the workload for constraints transmission in other objects can be reduced. Furthermore, we have examined the prediction in several types of the communication service, including unidirectional, bidirectional, mailbox-based, connectionless, and connect-oriented services. Thus, a software developer can compare with the relative accuracy of their prediction in applying this proposed approach.

The information contained in a message for requesting a service is different within various communication systems and applications. In addition, whether the communication service of constraints is acceptable may depend upon the real communication system. Thus, further work is needed to practice and compare this approach with real systems and applications. Moreover, we used the expected elapsed time to make the prediction. However, in real distributed computing systems, many factors affect the elapsed time for performing the services of a request. For example, a message may be blocked, duplicated, retransmitted, and objects may be reallocated [8]. Thus, to compensate for the complexity of the prediction in a distributed computing system, simulation analysis of the elapsed time can be done in the future.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1974.

[2] R. S. Chin and S. T. Chanson, "Distributed object-based programming systems," *ACM Computing Surveys*, vol. 23, pp. 91–124, March 1991.

[3] E. D. Lazowska, J. Zahorijan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

[4] C. C. Lien and C. C. Yang, "Rule-based performance evaluation," in *Int' Computer Symposium*, (Taiwan), pp. 791–797, Nov. 1990.

[5] N. H. Minsky, "The imposition of protocols over open distributed systems," *IEEE Trans. Software Eng.*, vol. 17, pp. 183–195, Feb. 1991.

[6] M. Sloman and J. Kramer, *Distributed Systems and Computer Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[7] J. A. Stankovic, "Misconceptions about real-time computing: A serious problems for next-generation systems," *IEEE Computer*, vol. 21, no. 10, pp. 10–19, 1988.

[8] A. S. Tanenbaum, *Computer Networks.* Englewood Cliffs, NJ: Prentice-Hall, 2nd ed., 1988.

[9] S. S. Yau, C. C. Yang, and S. M. Shatz, "An approach to distributed computing system software design," *IEEE Trans. Software Eng.*, vol. SE-7, pp. 427–436, July 1981.