Time Service Guranteeing in Real-Time Distributed Simulation Object Oriented Programming

Hee-Chul Kim[1], Gwang-Jun Kim[2], Moon-Hwan Kim[3], Sang-Dong Ra[4], and Chul-Soo Bae[5]
[1,2,3,4]School of Computer Engineering, Chosun University, Kwangju, Korea
Tel: +82-062-230-7757, Fax: +82-062-230-7381
[5] Dept. of Information & Communication Engineering, Kwandong University, Kangwon, Korea
Tel: +82-033-670-3411, Fax: +82-033-671-2118
e-mail : kimhc@teleoffice.co.kr, ban9628@hotmail.com, kmh@ktf.com,
sdna@mail.chosun.ac.kr, baecs@mail.kwandong.ac.kr

Abstract: The object-oriented(OO) distributed real-time(RT) programming movement started in 1990's and is growing rapidly at this turn of the century. Distributed real-time simulation is a field in its infancy but it is bounded to receive steadily growing recognition for its importance and wide applicability. The scheme is called the distributed time-triggered simulation scheme which is conceptually simple and easy to use but widely applicable. A new generation object oriented(OO) RT programming scheme is called the time-triggered message triggered object(TMO) programming scheme and it is used to make specific illustrations of the issues. The TMO structuring scheme is a general-style components structuring scheme and supports design of all types of component including hard real time objects and non real time objects within one general structure.

1. Introduction

Object oriented real time distributed computing is a rapidly growing branch of computer science and engineering. Its growth is fueled by the strong needs present in industry for the RT programming methods and tools which will bring about orders of magnitude improvement over the traditional RT programming practiced with low-level programming languages and styles.

RT simulator developments are under increasing demands[1,2,3,4]. For example, continuing advances in virtual reality application accompany increasing demands for more powerful RT simulation capabilities. Numerous other examples can also be found in the RT computing control field. Not only description but also simulation of application environments is often performed as integral steps of validating control computer system designs. RT simulators of application environments can often enable highly cost-effective testing of the control computer systems implemented. Such testing can be a lot cheaper than the testing performed in actual application environments while being much more effective than the testing based on non-RT simulators of environments.

The new generation OO RT programming scheme called the time-triggered message triggered object programming scheme[3,5,6]. In the course of developing a RT system engineering methodology based on this TMO programming scheme, a new approach to RT simulation which is conceptually simple and easy to use but widely applicable, has also been established.

In the next section, the motivations for pursuing the OO RT programming approach are reviewed and then in section 3, a brief overview is taken of the particular programming scheme. This programming scheme called the time-triggered message triggered object(TMO) programming scheme [3,6] is used on several occasions in the rest of this paper to make specific illustrations of the issues and potentials of OO RT programming..

2. An Overview of the TMO structure

In distributed RT simulation, simulator objects are distributed among multiple nodes. Synchronization of the simulation steps of distributed simulator objects in then a key challenge. In other words, a simulation step executed by every member of the distributed simulator object group must be synchronized with the corresponding simulation step executed by any other member. In other word, the simulator clock for one simulator object must commence the n-th tick neither before the (n-1)th tick by the clock for another simulator object nor after the (n+1)-th tick by the clock for another simulator object.

Therefore, every member must perform some activities necessary to stay synchronized with other members. For example, distributed nodes may exchange completion reports at the end of each simulation-step. However, this is not an efficient approach when the number of nodes used is large. The essence of the distributed time-triggered simulation approach is the following:

(1) Every node is equipped with an RT clock and executes each simulation step upon reaching of the RT clock at the predetermined value; and
(2) Every simulation step is designed to be completed within on ticking interval.

The distributed time triggered simulation approach has major advantages over other distributed simulation approaches, even if we assume that the latter approaches can be adapted somehow to enable RT simulation..

As a concrete example of a high-level OO RT distributed programming approach that has been based on the philosophy discussed in the preceding section, the time-triggered message-triggered object (TMO) programming scheme is briefly summarized in this section[2,3,4,5,6].

The TMO scheme was established in early 1990's with a concrete syntactic structure and execution semantics for economical reliable design and implementation of RT systems. The TMO scheme is a general-style component structuring scheme and supports design of all types of components including distributable objects and distributable non-RT objects within one general structure.

Calling the TMO scheme a high-level distributed programming scheme is justified by the following characteristics of the scheme :

(1) No manipulation of processes and threads:Concurrency is specified in an abstract form at the level of object methods. Since processes and threads are transparent to TMO programmers, the priorities assigned to them, if any, are not visible, either.

(2) No manipulation of hardware-dependent features in programming interactions among objects : TMO programmers are not burdened with any direct use of low-level network protocols and any direct manipulation of physical channels and physical node addresses/names.

(3) No specification of timing requirements in (indirect) terms other than start-windows and completion deadlines for program units (e.g., object methods) and time-windows for output actions : TMOs are devised to contain only high-level intuitive and yet precise expressions of timing requirements. Priorities are attributes often attached by the OS to low-level program abstractions such as threads and they are not natural expressions of timing requirements. Therefore, no such indirect and inaccurate styles of expressing timing requirements are associated with objects and methods.

At the same time the TMO scheme is aimed for enabling a great reduction of the designer's efforts in guaranteeing timely service capabilities of distributed computing application systems. It has been formulated from the beginning with the objective of enabling design-time guaranteeing of timely actions. The TMO incorporates several rules for execution of its components that make the analysis of the worst-case time behavior of TMOs to be systematic and relatively easy while not reducing the programming power in any way. TMO is a natural and syntactically minor but semantically powerful extension of the conventional object(s)[6].
As depicted in Fig. 1 the basic TMO structure consists of four parts :

ODS-sec = object-data-store section : list of object-data-store segments(ODSS's);
EAC-sec = environment access-capability section : list of gate objects (to be discussed later) providing efficient call-paths to remote object methods, logical communication channels, and I/O device interfaces;
SpM-sec = spontaneous-method section : list of spontaneous methods;
SvM-sec = service-method section.

The TMO model is a syntactically minor and semantically powerful extension of the conventional object model. Significant extensions are summarized below and the second and third are the most unique extensions.
(a) Distributed computing component
The TMO is a distributed computing component and thus TMO's distributed over multiple modes may interact via remote method calls. To maximize the concurrency in execution of client methods in one node and server methods in the same node or different nodes, client methods are allowed to make non-blocking types of service requests to server methods

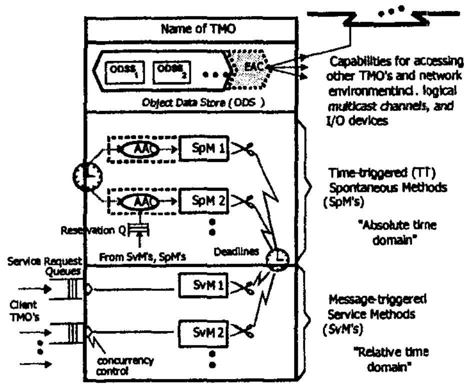

Figure 1. Structure of the TMO

(b) Clear separation between two types of methods
The TMO may contain two types of methods, time-triggered(TT-) methods (also called the spontaneous methods or SpM's) which are clearly separated form the conventional service methods(SvM's). The SpM executions are triggered upon reaching of the RT clock at specific values determined at the design time whereas the SvM executions are triggered by service request messages from clients. Moreover, actions to be taken at real times which can be determined at the design time can appear only in SpM's.
(c) Basic concurrency constraint(BCC)
This rule prevents potential conflicts between SpM's and SvM's and reduces the designers efforts in guaranteeing timely service capabilities of TMO's. Basically, activation of an SvM triggered by a message form an external client is allowed only when potentially conflicting SpM executions are not in place. An SvM is allowed to execute only if no SpM that accesses the same object data store segments(ODSS's) to be accessed by this SvM has an execution time-window that will overlap with the execution time-window of this SvM.
(d) Guaranteed completion time and deadline
As in other RT object models, the TMO incorporates deadlines and it does so in the most general form. Basically, for output actions and completion of a method of a TMO, the designer guarantees and advertises execution time-window bounded by start times and completion times. Triggering times for SpM's must be fully specified as constants during the design time. Those RT constants appears in the first clause of an SpM specification called the autonomous activation condition(AAC) section

A provision is also made for making the AAC section of an SpM contain only candidate triggering times, not actual triggering times, so that a subset of the candidate triggering times indicated in the AAC section may be dynamically chosen for actual triggering. Such a dynamic selection occurs when an SvM or SpM within the same TMO requests future executions of a specific SpM. TMO's interact via calls by client objects for service methods in server objects. The caller maybe an SpM or an SvM in the client object. The designer of each TMO provides a guarantee of timely service capabilities of the object. He/she does so by indicating the guaranteed execution time-window for every output produced by each SvM as well as by each SpM executed on requests from the SvM and the guaranteed completion time for the SvM in the specification of the SvM. Such specification of each SvM is advertised to the designers of potential clients objects. Before determining the time-window specification, the server object designer must convince himself/herself that with the object execution engine (hardware plus operating system) available, the server object can be implemented to always execute the SvM such that the output action is performed within the time-window. The BCC contributes to major reduction of these burdens imposed on the designer. Models and prototype implementations of the effective operating system(OS) support and the friendly application programmer interface(API) have been developed.

The TMO model is effective not only in the multiple-level abstraction of RT (computer) control systems under design but also in the accurate representation and simulation of the application environments. In fact, it enables uniform structuring of control computer systems and application environment simulators and this presents considerable potential benefits to the system engineers.

3. Deadline specification and service time guarantee

As mentioned in the overview of the TMO scheme, the designer of each RT object can provide a guarantee of the timely service capabilities of the object by indicating the execution time-window for every output produced by each service method in the specification of the service method advertised to the designers of potential client objects. Actually the execution time-window associated with every output form every TT method is also a part of the guarantee.

An output action of a service method ma be one of the following

(1) An updating of a portion of the ODS;
(2) Sending a message to either another RT object (which may or may not be the client) or a device shared by multiple objects;
(3) Placing a reservation into the reservation queue for a certain TT method that will in turn take its own output actions.

The specification of each service method which is provided to the designers of potential client RT objects, must contain at least the following:

(1) An input specification that consists of
   (1a) the types of input parameters that the server object can accept and
   (1b) the maximum request acceptance rate, i.e., the maximum rate at which the server object can receive service requests form client objects;
(2) An output specification that indicates the maximum delay (not the exact output time) and the nature of the output value for every output produced by the service method.

If service requests form client object arrive at a server object at a rate exceeding the maximum acceptance rate indicated in the input specification for the server object, then the server may return exception signals to the client objects. The system designer who checks an interconnection of RT objects can prevent such "overflow" occurrences through a careful analysis. The designer should ensure that the aggregate arrival rate of service request at each server object does not exceed the maximum acceptance rate during any period of system operation. In order to satisfy greater service demands presented by the client objects, the system designer can increase the number of server objects or use more powerful execution engines in running server objects. Before determining the maximum delay specification, the server object designer must consider the following.

(1) The worst-case delay form the arrival of a service request form a client object to the initiation of the corresponding service method by the server object;
(2) The worst-case execution time for the service method from its initiation to each of its output actions.

On the other hand, a client RT object imposes a deadline on the cooperating distributed object execution engines for producing the intended computational effects including the execution of the called service method and the arrival of the return results at the client object If this deadline is violated, the execution engine for the client object invokes an appropriate exception handling function.

The specifications of the TT methods which may be executed on requests from the service method must also be provided to the designers of the client objects which may call the service method. The specification of such a TT method must contain at least the time triggering specification and the output specification. There is no input specification. The output specification indicates, for every output expected from the execution of the TT method, the exact time at or by which it will be produced and the nature of every value carried in the output action.

4. Simulation design with TMO structure

The attractive basic design style facilitated by the TMO structuring is to produce a network of TMO's meeting the application requirements in a top-down multi-step fashion. For each environment object represented by a state descriptor in the Mini Theater TMO, there is a spontaneous method (SpM) for periodically updating the state descriptor. Conceptually the SpM's in the Mini Theater TMO are activated continuously and each of their executions is completed instantly. The SpM's can then represent continuous state changes that occur naturally in the environment objects. The natural parallelism that exists among the environment objects can also be precisely represented by use of multiple SpM's which may be activated simultaneously.

The service methods (SvM's) in the Mini Theater TMO are provided as an interface for the clients outside the Mini Theater. The only conceivable clients here are the enemy which send RV's into the Mini Theater to enter the Mini Theater. Entry of an RV into the Mini Theater is represented by and enemys call for the SvM "Accept RV". Both the enemy and the external forces are represented by a TMO called the Alien TMO.

So far, the Mini Theater TMO in Fig. 2 has been *interpreted* as a mere description of the application environment. However, if the activation frequency of each SpM is chosen such that it can be supported by an object execution engine, then the resulting Mini Theater TMO becomes a simulation model. The behavior of the application environment is represented by this simulation model somewhat less accurately than by the earlier description model based on continuous activation of SpM's. In general, the accuracy of a TMO structured simulation is a function of the chosen activation frequencies of SpM's. Upon receiving the customers order, the system engineer will first decide on the set of sensors and actuators to be deployed in the Mini Theater. After the set of sensors and actuators is determined, the Mini Theater TMO in Fig. 2 is expanded to incorporate all the components enclosed by square brackets.

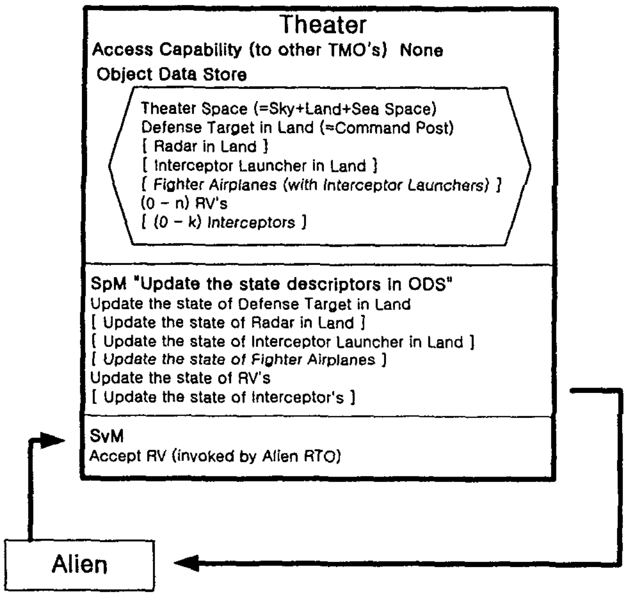
Theater
Access Capability (to other TMO's) None
Object Data Store
Theater Space (=Sky+Land+Sea Space)
Defense Target in Land (=Command Post)
[Radar in Land]
[Interceptor Launcher in Land]
[Fighter Airplanes (with Interceptor Launchers)]
(0 – n) RV's
[(0 – k) Interceptors]
SpM "Update the state descriptors in ODS"
Update the state of Defense Target in Land
[Update the state of Radar in Land]
[Update the state of Interceptor Launcher in Land]
[Update the state of Fighter Airplanes]
Update the state of RV's
[Update the state of Interceptor's]
SvM
Accept RV (invoked by Alien RTO)
Alien

Fig. 2. High-level specification of the *Mini Theater TMO.*

The ODS now contains the selected sensors (e.g., Radar in Land) and actuators (i.e., Interceptor Launcher in Land with Interceptors). The radar and another interceptor launcher on the fighter airplane are not shown in the ODS of the Mini Theater TMO but these environment objects are described in the corresponding parts of the state descriptors for the command ship and the fighter airplane, respectively.

The Mini Theater Space component in the ODS of the Mini Theater TMO not only provides geographical information about the Mini Theater but also maintains the position information of every moving object in the Mini Theater. This information is used to determine the occurrences of collisions among objects and to recognize the departure of any object from the Mini Theater space to the outside.

As the system engineer refines the single TMO representation of the Mini Theater, a component in the ODS of the Mini Theater TMO may be taken out of the Mini Theater TMO to form a new TMO. such separation of the command post from the Mini Theater TMO. Therefore, the Command Post TMO and the Command Ship TMO are born. When the new TMO's are created, the SvM's that serve as front-end interfaces of those new TMO's and the call links from the earlier born TMO's to the new TMO's should also be created. As a result, the Mini Theater TMO becomes a network of three TMO's. The two new TMO's may describe or simulate the command post and the command ship more accurately than the Mini Theater TMO in Fig. 2 did.

5. Conclusion

The essence of the GG design paradigm advocated in this and earlier publications by the author, which is also the goal of the TMO structuring scheme, is to realize RT computing in a general manner not alienating the main-stream computing industry and yet enabling the system engineer to confidently produce certifiable RTCS's for safety-critical applications. This author believes that time is ripe for vigorously pursuing this idealistic approach.

Although the potential of the TMO scheme has been amply demonstrated, much further research efforts are needed to make the TMO structuring technology easily accessible to common practitioners. Further development of TMO support middleware, especially those running on new-generation RT kernels and multiprocessor hardware, is a sensible topic for future research. Tools assisting the TMO designer in the process of determining the response time to be guaranteed are among the most important research topics.

References

[1] A. Attoui and M. Schneider, "An object-oriented model for parallel and reactive systems", *Proc. IEEE CS 12th Real-Time Systems Symp.*, pp. 84-93, 1991

[2] K. H. Kim et al., "A timeliness-guaranteed Kernel model DREAM kernel and implementation techniques", *Proc. Workshop on Real-Time Computing Systems and Applications (RTCSA 95)*, Tokyo, Japan, pp. 80-87.Oct. 1995

[3] K. H. Kim and C. Subbaraman, "Fault-tolerant real-time objects", *Commun. ACM pp.* 75-82. 1997

[4] K. H. Kim, C. Subbaraman, and L. Bacellar, "Support for RTO.k Object Structured Programming in C++", *Control Engineering Practice* 5 pp. 983-991, 1997

[5] K. H. Kim, "Object Structures for Real-Time Systems and Simulators", *IEEE Computer 30.*, pp.62-70,1997

[6] G.J.K, S.D.Na and C.S.Bae, " Time Service Guarantee in Real-Time Distributed Object Oriented Programming of TMO", *Proc.ICIM'01*,pp.215-219,Nov.2001

[7] H. Kopetz and K. H. Kim, "Temporal uncertainties in interactions among real-time objects", *Proc. IEEE CS 9th Symp. On Reliable Distributed Systems.*, pp. 165-174,Oct. 1990