TMO-Based Object Group Framework for Supporting Distributed Object Management and Real-Time Services

Chang-Sun Shin, Myoung-Suk Kang, Chang-Won Jeong, and Su-Chong Joo

School of Electrical, Electronic and Information Engineering,
Wonkwang University,
Korea
{csshin,gnb,mediblue,scjoo}@wonkwang.ac.kr

Abstract. In this paper, we present a TMO-based object group framework that can support the distributed object group management and the real-time scheduling services on distributed real-time computing environments. These environments have some difficulties for managing lots of distributed objects and providing the timing constraints to real-time objects. For simultaneously solving these problems, we design a TMO object group framework that can manage as a grouping unit of the distributed TMO objects in order to reduce their own complicated managements and interfaces among individual objects without modifying the ORB itself. The TMO object as real-time object, defines the object having real-time property developed from Dream Laboratory at UC at Irvine. The TMO object group we suggested contains several components reflected the object grouping concepts and real-time service requirements analyzed by referring OMG CORBA specifications. To construct our TMO object group framework, we designed the TMO object group structure, and described the functional class diagram with representing relationships among components. We also explained the detailed functional definitions and interactions between the components from the following 2 points of views; object management service by the Dynamic Binder object for selecting an appropriate one out of objects with the same property, and the real-time scheduling service by the Scheduler object and the Real-Time Manager object. We finally verified the results produced by using the known algorithms like the Binding Priority algorithm and the EDF algorithm to see whether a distributed object management service and a real-time service can adapt on the suggested framework.

1 Introduction and Related Works

The modern computing environments have been changing toward the distributed real-time object computing environments with growing the real-time service requirements of practical applications. Though the existing real-time applications largely run as simple real-time constraints on a single processor system, nowadays most real-time applications, like avionics, widely distributed defense systems and so forth, are required to complex time constraints for distributed real-time services. That is, the distributed real-time applications might be executed by one or more logically distributed objects and required the exacting results by interacting among the distributed objects and the timely operation satisfying the real-time constraints[1,2,3].

As one of the representative researches that efficiently manage individual objects at point of view for distributed object oriented applications, the TINA-C (Telecommunications Information Networking Architecture-Consortium) defined the TINA[4]. In the TINA specification, distributed applications can be logically executed as unit of associated objects, called object group, on multiple systems. According to their specification, they have defined only the management specification of object group and the distributed functional components. Nevertheless, they have not defined the detailed specification about real-time services in a distributed environment yet.

For compensating the TINA's weakness, the OMG(Object Management Group) specified with CORBA (Common Object Request Broker Architecture) providing the standard software specification for improving the flexibility, scalability, reusability and etc via objects' implementation of the distributed environments[5], but it is impossible to support the real-time properties for the distributed real-time applications. After this, the RT-SIG(Real-Time Special Interest Group) organized by the OMG suggested the development of the CORBA specification having the real-time extensibility(CORBA/RT) for adding the real-time property to CORBA specification[6]. It made the distributed real-time environments that depended on the special system and/or the operating systems for the real-time services by modifying or extending the ORB that is the core of CORBA. In these same times, the Dream Laboratory at UCI reported the TMO(Time-triggered Message-triggered Object) scheme[9,10]. The TMO object is defined as an object having real-time property itself. The TMO object scheme is syntactically a simple and natural but semantically powerful extension of conventional object structuring approaches. However, the TMO scheme cannot support the concept of object group for real-time scheduling service and dynamically an appropriate object selection mechanism from replicated TMO objects with the same service property in an object group.

With following up the TINA and CORBA specifications, and providing both the object group management and real-time service on independent framework based on CORBA, the Real-Time Object Group(RTOG) model in the distributed environments has been researched by our researches[2,7,8]. As you referred our papers, the RTOG is developed for supporting the distributed real-time services based on the TINA's object group concepts without modifying the ORB on the standard CORBA. But this RTOG model is not enough to address problem of a dynamic object binding service among replicated objects that are called objects providing the same service property.

In this paper, for solving some problems mentioned above which did not studied in point of view of the object group, we suggest the TMO object group framework that can manage the TMO objects as a unit of the object group on COTS(Commercial Off-The-Shelf) middleware and provide the execution power of the guaranteed real-time services. With given pre-requirements of distributed real-time services, we defined the concepts of the TMO object and a structure of the TMO object group. We also explained the detailed functional definitions and interactions between components from the following 2 points of views; the object management service by the Dynamic Binder object for selecting an appropriate one out of objects with the same property, and the real-time scheduling service by the Scheduler object and the Real-Time Manager object. We finally verified the results executed by using the known algorithms like the Binding Priority algorithm and the EDF(Earliest Deadline First) algorithm to see whether a distributed object management service and a real-time service can adapt on the suggested framework. These algorithms might be altered to others for improving adaptation of our framework. In this paper, the viewpoint of performance is out of boundary, because we are interested in mapping a physical environment to logical one, our framework, directly at aspect of adaptability.

2 TMO Object Group

This section explain the whole of overview of the suggested TMO object group for managing individual objects in an object group and guaranteeing the real-time service in the distributed systems. The TMO object scheme[9,10] we used in paper developed from Dream Laboratory at UC at Irvine. The TMO objects, as service objects contained in an object group, are implemented by using this scheme.

2.1 TMO Object Scheme

The TMO object is defined as a real-time object having real-time property itself. This object scheme is extending the concept of existing service object. Be different from the generic object concept, TMO object has additionally an SpM(Spontaneous Method) that can be spontaneously triggered by the defined time in an object. Figure 1 is shown its structure. The TMO object contains its name, an ODS, EAC, AAC, SpMs, and SvMs as follows. As stated in [10], the role of each component is described like below.

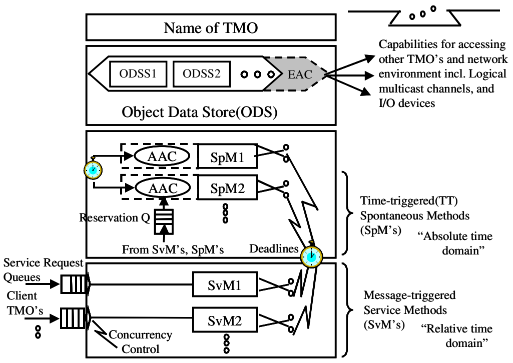


Fig. 1. Structure of TMO object scheme

① ODS(Object Data Store) : Storage of properties and states of the TMO object.

② EAC(Environment Access Capability) : List of gates to objects to providing efficient call-paths to remote object methods, logical communication channels, and I/O device interfaces.

③ AAC(Autonomous Activation Condition) : Activation condition for an SpM, which defines the time window for the execution of that SpMs.

④ SpM(Spontaneous Method) : A time triggered method which runs in real-time a periodic manner.

⑤ SvM(Service Method) : A message triggered method which responds to external service requests.

In details, an ODS is a common object data store being accessed by the SpM and the SvM, and both methods could not be accessed it simultaneously. When SpM and SvM access the ODS at the same time, the SpM's priority is higher than the SvM's one. That is, the TMO object is triggered by the BCC(Basic Concurrency Constraints). The EAC is responsible for the interface call of the communication channels and the I/O devices. The SpM and the SvM are the list of methods, which are clearly separated from the existing object. The TMO object can have several SpMs and SvMs. In the AAC(Autonomous Activation Condition) that located in the first clause of the SpM, we specify the activation time of the SpM, and implement the TMO object as the real-time object. The TMO object scheme is going on the lively research in the real-time simulation as the military or the transportation applications. But it is impossible to check the security for the object access, manage the replicated TMO objects with the same property and support the distributed scheduling service of several TMO objects globally in a given object group. We intended to solve these problems by taking advantage of the TMO object group model we suggested.

2.2 TMO Object Group Framework

The TMO object group framework proposed in this paper is a new structure that can apply the object group concept being suggested by TINA on COTS middleware. The TMO object group is represented by a logical unit that consists of a set of objects to manage an object group and a set of TMO objects to execute real-time services.

Let us explain the components and their functionalities in the TMO object group. The major roles of these components are categorized into two kinds of services; the object management service and the real-time scheduling service. For supporting the object management service, our framework contains several components, such as the Group Manager(GM) object, the Security object, the Information Repository object and the Dynamic Binder object. And for supporting the real-time scheduling service, our framework contains several components, such as the TMO objects, the Real-Time Manager(RTM) objects and the Scheduler objects. And the TMO object group may contains the replicated TMO objects with the same property and the Sub-TMO object groups as a nested inner object group. A nested object group allows encapsulation and hierarchical organization. Figure 2 shows the structure of the TMO object group framework.

From this framework, at point of view of a support of the object management service, the GM object is totally responsible for managing of all of objects being in an object group and returning the unique reference of the requesting TMO object to a client. The Security object checks access rights of an object requested by referring the access control list(ACL). The Information Repository object stores information such as service properties and their references about all of TMO objects existing in an object group. The GM object is also responsible for maintaining this information, and this will be used for selecting an arbitrary object or an appropriate one out of replicated TMO objects. In this procedure, the Information Repository object sends their references to the Dynamic Binder object. After then, the Dynamic Binder object selects an appropriate object that will be invoked by a client, after referring each system's load information and deadline. For selecting an appropriate one out of the replicated TMO objects, we will adopt the known sample algorithm like the binding priority algorithm considering system's workload and network traffic information and the request deadline as inputs to the Dynamic Binder object for calculating binding priorities of the replicated TMO objects. This algorithm assigns the binding priorities to the replicated objects individually. The GM object finally returns the reference of the selected object with the highest priority to a client. In case of binding with a non-replicated object, it is trivial.

At point of view of a support of the real-time scheduling service, the RTM object takes the calculated service deadline of the TMO object requested over the Scheduler object. This basic flow procedure occurred from clients' request to getting results is divided into 3 detailed steps; a service requesting step, a service processing step and a result returning step. Each step should be defined the timing constraints itself. Considering given timing constraints, the Scheduler object assigns the priority to the requesting tasks according to scheduling sequences. For verifying whether the Scheduler object can schedule or not in our model, we adopt the EDF scheduling algorithm, as an algorithm, to the Scheduler object for deciding the requests' priority. The Figure 3 shows the functional class diagram of the components that organized in the TMO object group using the Object Modeling Technique(OMT).

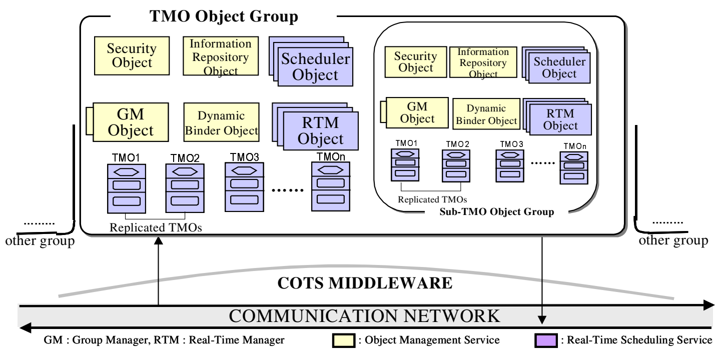


Fig. 2. Structure of the TMO object group framework

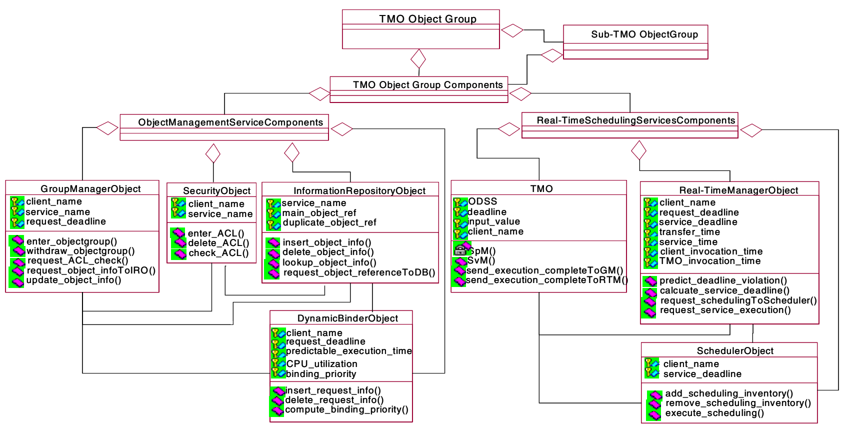


Fig. 3. Functional class diagram of components in a TMO object group

2.3 Timing Constraints

To support the object management and the real-time scheduling service at our TMO object group, the binding priority algorithm and the EDF algorithm adapting to the Dynamic Binder object and the Scheduler object respectively are requiring the timing constraints for real-time services. The basic flow procedure obtaining the result returned from a request is divided into 3 steps; the service request step, the service process step and the result returning step. Each step must have individual timing constraint and guarantee the timeliness by explicitly outlining the definitions of timing constraints. Therefore, we define five timing constraints with showing Figure 4 in our study as follows;

— An invocation time(IT) constraint specifies CIT(Client's Invocation Time), as a time that a client sends a request message to a TMO object and SIT(Service TMO's Invoked Time), as a time that a service TMO object received a request of a client.
— A service time(ST) constraint specifies the relative time for the service execution of a TMO object.
— A transfer time(TT) constraint specifies the relative time required for transferring a request from a client to an TMO object(SIT-CIT), or inversely.
— A service deadline(SD) constraint specifies the absolute time that a TMO object completes the service requested.
— A request deadline(RD) constraint specifies the absolute time that a client received a result returning from a TMO object after a client requested a TMO object. Here, a RD is the timing constraint that must be guaranteed in a distributed real-time application.

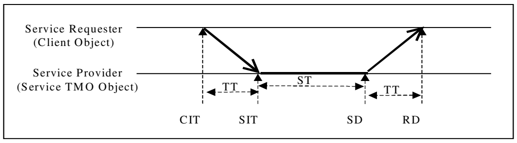


Fig. 4. Definition of timing constraint

From this figure, the timing constraint of each step can be expressed by following equations;

$$RD \geq CIT+ST + (2 \times TT) + slack\ time,$$
$$SD \leq RD - TT,\ where\ TT = SIT\text{-}CIT\ and\ ST = SD - SIT.$$

The deadline for service execution(RD) is the sum of the client's invocation time(CIT), the service time(ST), total transfer time(2×TT) and slack time. Here, the *slack time* means a constant as a factor to decide the time of the adequate RD. This service deadline(SD) must be smaller than and equal a time value that subtracts transfer time(TT) from a client's request deadline(RD).

3 Object Management Services

As we shown in Figure 2, the GM object is a representative of all of objects in a given object group. For requesting a service, a client should firstly obtain a reference of the

GM object existing in an object group via a Naming Server, and then, request the desiring TMO object's reference to the GM object. To do so, the GM object checks access rights of the requesting TMO object from the Security object, and if it is possible to be accessed, it continuously requests to the Information Repository object for getting TMO object's reference. The Information Repository object searches a TMO object's reference from the object table managing objects in a group itself, and returns it to the GM object. At this time, if the TMO objects requested are being replicated in a given object group, the GM object requests the Dynamic Binder object to obtain the reference of an appropriate TMO object by way of the Information Repository object. The Dynamic Binder object will be selected an appropriate one out of the replicated TMO objects using an arbitrary algorithm. Otherwise, that is, if the replicated TMO objects are not existed in a given object group, this algorithm will not be needed. Through these procedures, finally the GM object receives the TMO object's reference, and returns one to a client inversely. Figure 5 shows an Event Trace Diagram(ETD) for representing the whole management procedures mentioned above. We note that the binding priority algorithm is only used as a sample algorithm for implementation of the Dynamic Binder object.

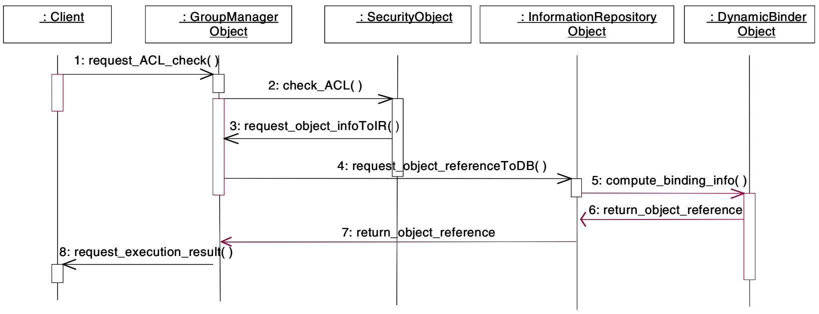

: Client
: GroupManager Object
: SecurityObject
: InformationRepository Object
: DynamicBinder Object
1: request_ACL_check( )
2: check_ACL( )
3: request_object_infoToIR( )
4: request_object_referenceToDB( )
5: compute_binding_info( )
6: return_object_reference
7: return_object_reference
8: request_execution_result( )

Fig. 5. The ETD for object management service

3.1 Dynamic Selection and Binding Service from Replicated TMO Objects

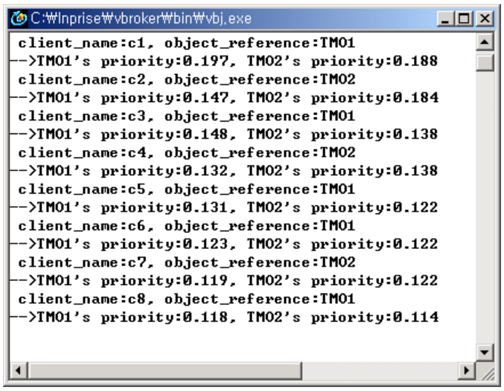
When two or more TMO objects with the same service property, called replicated target object, exist in an object group for supporting service, the least one out of replicated TMO objects should be selected and this selected object have to bind to the client object. For this reason, the Dynamic Binder object is implemented as an algorithm for selecting an appropriate one among replicated TMO objects. Here, we explain an example algorithm, called the binding priority algorithm, which describes with calculating each object's binding priority and then selecting an appropriate object's reference. As input parameters of this algorithm, we use load information of systems that objects are located on and the request deadline information of client's task. You do not care of adopting another algorithm as an alternative. In this section, we only use the binding priority algorithm for verifying whether the Dynamic Binder object in our object group may operate correctly, but not showing the improved performance of our framework itself. Let us consider the calculation of binding-priority. The $binding\_priority_k$ for the client's $request_k$ can calculate by the following expression[1,11].

$$binding\_priority_k = \frac{1}{\sum_{i=0}^{k} request\_deadline_i} + \frac{c}{CPU\_utilization}$$

where
request_deadline : client's request deadline, CPU_utilization: CPU utilization rate, c : rate constant(0.01)

From above expression we have defined, we can obtain the binding priorities of all of replicated TMO objects with the same property. In the dynamic object selection and binding procedure, we just allocate client's request to the TMO object with the highest binding-priority out of replicated ones. That is, the client will be received the reference of the selected object for binding between a client object and a server object. To verify the dynamic selection and binding service in our object management services, we show an appropriate example as follows;

In our framework environment, Let us assume that a sequence of 8 clients' requests continue to arrive to the GM object in an object group for invoking replicated TMO objects(TMO1 or TMO2) with the same property and they have already had their client names, request deadlines, CPU utilizations of systems in which TMO1 and TMO2 are located, and binding-priorities for taking TMO1 or TMO2 services, that are the client's status information. Here, we will not numerically describe the client's status information, like request-deadline(RD), CPU-utilization, and so on. The binding-priority will be decided through competitions of clients waiting in Ready Queues of TMO1 and TMO2 using the clients' status information. After finishing the dynamic selection and binding service, we can get the results that client c1, c3, c5, c6 and c8 are bound to TMO1, and c2, c4 and c7 are bound to TMO2. Here, if each CPU's utilization may change due to system overloads, the binding priorities should be also changed. Figure 6 shows the binding priorities and the selected binding references as results executed by the Dynamic Binder object.



```
 client_name:c1, object_reference:TMO1
-->TMO1's priority:0.197, TMO2's priority:0.188
 client_name:c2, object_reference:TMO2
-->TMO1's priority:0.147, TMO2's priority:0.184
 client_name:c3, object_reference:TMO1
-->TMO1's priority:0.148, TMO2's priority:0.138
 client_name:c4, object_reference:TMO2
-->TMO1's priority:0.132, TMO2's priority:0.138
 client_name:c5, object_reference:TMO1
-->TMO1's priority:0.131, TMO2's priority:0.122
 client_name:c6, object_reference:TMO1
-->TMO1's priority:0.123, TMO2's priority:0.122
 client_name:c7, object_reference:TMO2
-->TMO1's priority:0.119, TMO2's priority:0.122
 client_name:c8, object_reference:TMO1
-->TMO1's priority:0.118, TMO2's priority:0.114
```

Fig. 6. Results executed by the Dynamic Binder object

The each component in the TMO object group was designed to object-based technology, and implemented by using VisiBroker running on Windows 2K and Linux for independently supporting heterogeneous distributed computing environments.

4 Real-Time Services

For supporting real-time service from the TMO object group framework, we use the client's status information described above section. On coming client TMOs' requests into an TMO object group including the desiring target TMO object, each client TMO object transmits its own status information to the Real-Time Manager(RTM) object. After calculating the service deadline(SD) which a target TMO object should be served for a client, the RTM send the calculated SD and the client's status information toward the Scheduler object. The Scheduler object schedules the client's task, i.e. client request by using the Earliest Deadline First(EDF) algorithm. In this section, we only use the EDF algorithm for verifying whether the Scheduler object may be scheduled correctly, but not interesting in the improved performance of our framework itself. The below Figure 7 showed the ETD described interactions among the relevant objects for supporting the real-time scheduling service in the TMO object group.

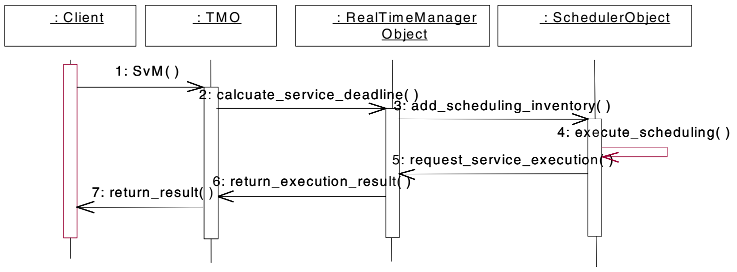
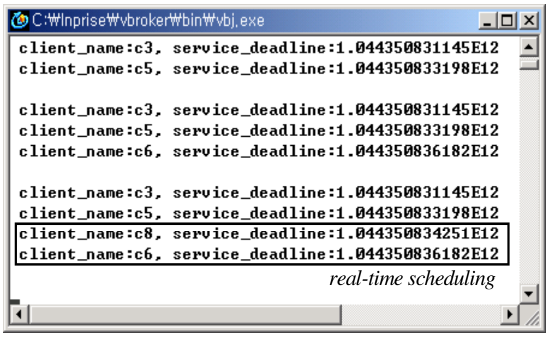

: Client
: TMO
: RealTimeManager Object
: SchedulerObject
1: SvM( )
2: calcuate_service_deadline( )
3: add_scheduling_inventory( )
4: execute_scheduling( )
5: request_service_execution( )
6: return_execution_result( )
7: return_result( )

Fig. 7. The ETD for real-time scheduling service

According to real-time scheduling procedures, the Scheduler object have to immediately execute a task of a client's request whenever the TMO object stays in the idle state. Otherwise, the successive client's requests have to be piled on the Scheduler object's ready queue and waited for taking the guaranteed service until the executing task is finished. The Scheduler object is implemented to the EDF algorithm with non-preemptive property. According to this algorithm, the highest priority task is defined the client's request with minimum service deadline. The task execution priority list will be renewed whenever tasks are arrived or finished. The following expression describes making a condition deciding a task priority(TP), given two tasks' service deadlines[12].

$$\textit{if } SD_i < SD_{i-1} \textit{ then } TP_i > TP_{i-1}$$

Let us consider the same framework environment given at chapter 3. After finishing the dynamic TMO object selection and binding service, like Figure 6. Client c1, c3, c5, c6 and c8 are assigned to TMO1, and client c2, c4 and c7 are assigned to TMO2. Here, to verify the real-time scheduling service on our object group framework, we show the all of scheduling procedures of TMO1 that are requested by c1, c3, c5, c6 and c8. For real-time scheduling, we considered the service deadline(SD) which subtracted the transfer time(TT) from client's request deadline(RD) . That is, the RTM object calculates the service deadline(SD) from following expression; SD = RD-TT. The RTM object invokes the Scheduler object to decide the priority level. As a result, the Figure 8 is shown a sequence of their executions on window screen, as scheduling execution results(c1, c3, c5, c8 and c6) of the Scheduler object on our framework, when TMO1 is requested by the coming sequent clients(c1, c3, c5, c6 and c8).



```
client_name:c3, service_deadline:1.044350831145E12
client_name:c5, service_deadline:1.044350833198E12

client_name:c3, service_deadline:1.044350831145E12
client_name:c5, service_deadline:1.044350833198E12
client_name:c6, service_deadline:1.044350836182E12

client_name:c3, service_deadline:1.044350831145E12
client_name:c5, service_deadline:1.044350833198E12
client_name:c8, service_deadline:1.044350834251E12
client_name:c6, service_deadline:1.044350836182E12
```

Fig. 8. Scheduling results of the Scheduler object for the target object TMO1

5 Conclusions

The TMO object group framework proposed in this paper is a logical distributed structure that can provide the object group management and the TMO-based real-time services on COTS middleware without restricting the real-time CORBA or operating systems. For achieving this framework, we described whole of overview of the TMO object group, such as the concepts of the TMO object, structuring our model, designing the functions, interactions among the components in the object group, and implementing the Dynamic Binder object and the Scheduler object for supporting dynamic binding and real-time scheduling services. We also described the ETD for conveniently showing procedures of the object group management and real-time service in our framework. After then, in order to verify executions of the framework we constructed, we adopted the known algorithms as implementation of these objects described above, while implementing Dynamic Binder object and Scheduler object respectively. The algorithms we used are the binding priority algorithm for dynamic binding service and the EDF algorithm for real-time scheduling service. For the reason we used them, we only use these algorithms for verifying whether our framework may operate correctly, but not showing the improved performance of the framework itself. With the execution results obtained from our framework using above algorithms, we showed that our framework can be being supported not only the dynamic selection and binding service of replicated or non-replicated TMO objects from client's request, but also real-time scheduling service for an arbitrary TMO object requested from clients.

In future, for applying this framework to practical fields, we have a plan to develop a prototypical framework that can variously adopt new dynamic binding and real-time scheduling strategies to Dynamic Binder object and Scheduler object in an object group. After then we will verify the execution power of the distributed real-time application on the TMO object group framework via various simulations for improving real-time services and convenient distributed object management services.

Acknowledgements. The authors wish to acknowledge helpful discussions of the TMO Programming Scheme with Professor Kane Kim in UC at Irvine DREAM Lab. This work reported here was supported in part by Research funds of Wonkwang University and the Brain Korea 21-Project of KRF, 2003.

References

1. M. Takemoto: Fault-Tolerant Object on Network-wide Distributed Object-Oriented Systems for Future Telecommunications Applications. In IEEE PRFTS (1997) 139–146
2. W.J. Lee, C.W. Jeong, M.H. Kim, and S.C. Joo: Design and Implementation of An Object Group in Distributed Computing Environments. Journal of Electronics & Computer Science, Vol. 2, No. 1 (2000)
3. E.D. Jensen, C.D. Locky, and H. Tokuda: A Time-Driven Scheduling Model for Real-Time Operating Systems. In Proc. 6th IEEE Real-Time System Symposium (1985) 112–122
4. L. Kristiansen, P.Farley, R.Minetti, M. Mampaey, P.F. Hansen, and C.A. Licciardi: TINA Service Architecture and Specifications. http://www.tinac.com/specifications
5. Object Management Group: The Common Object Request Broker: Architecture and Specification 2.2. http://www.omg.org/corba/corbaCB.htm (1998)
6. OMG Real-time Platform SIG: Real-time CORBA A White Paper-Issue 1.0. http://www.omg.org/real time/real-time_whitepapers.html (1996)
7. C.S. Shin, M.H. Kim, Y.S. Jeong, S.K. Han, and S.C. Joo: Construction of CORBA Based Object Group Platform for Distributed Real-Time Services. In Proc. 7th IEEE Int'l Workshop on Object-oriented Real-time Dependable Systems (WORDS'02) (2002) 229–302
8. C.S. Shin, M.S. Kang, Y.S. Jeong, S.K. Han, and S.C Joo: TMO-Based Object Group Model for Distributed Real-Time Services. In Proc. IASTED Int'l Conference Networks, Parallel and Distributed Processing, and Applications(NPDPA'02) (2002) 178–183
9. K.H. Kim: Object-Oriented Real-Time Distributed Programming and Support Middleware. In Proc. 7th Int'l Conf. on Parallel & Distributed System (2000) 10–20
10. K.H. Kim, Seok-Joong Kang, and Yuqing Li: GUI Approach to Generation of Code-Frameworks of TMO. In Proc. 7th IEEE Int'l Workshop on Object-oriented Real-time Dependable Systems(WORDS'02) (2002) 17–25
11. V. Kalogeraki, P.M. Melliar-Smith, and L.E. Moser: Dynamic Scheduling for Soft Real-Time Distributed Object Systems. In Proc. IEEE 3rd Int'l Symp. on Object-Oriented Real-Time Distributed Computing (2000) 114–121
12. John A. Stankovic, Marco Spuri, Krithi Ramamrithm, Giorgio C. Buttazzo: Deadline Scheduling for Real-Time Systems. Kluwer Academic Publishers (2002) 31
13. G.M. Shin, M.H. Kim, and S.C. Joo: Distributed Objects Grouping and Management for Supporting Real-Time in CORBA Environments. Journal of The Korea Information Processing Society, Vol. 6, No. 5 (1999)
14. B.T Jun, M. Kim, and S.C Joo: The Construction of QoS Integration Platform for Real-Time Negotiation and Adaptation Stream in Distributed Object Computing Environments. Journal of The Korea Information Processing Society, Vol. 7, No. 11S (2000)