CORBA Based Real-Time Object-Group Platform in Distributed Computing Environments

Su-Chong Joo[1], Sung-Kwun Oh[1], Chang-Sun Shin[1], and Jun Hwang[2]

[1] School of Electrical, Electronic and Information Engineering, Wonkwang University, Korea
{scjoo,ohsk,csshin}@wonkwang.ac.kr

[2] Department of Computer Science, Seoul Women University, Korea
hjun@swu.ac.kr

Abstract. Recently, the distributed real-time services are developing in distributed object computing environments in a way that can support a new programming paradigm of the distributed platform that requires interoperability among heterogeneous systems. These services are based on distributed middleware and object-oriented technologies. But we have the difficulties of managing of distributed objects and providing real-time objects with the timing constraints. For simultaneously solving these problems, we designed a new model, a real-time object group (RTOG) platform that can manage and group the distributed objects for reducing their own complicated managements and interfaces, and add distributed real-time requirements to real-time objects without modifying the ORB itself. The structure of the real-time object group we suggested contains several components reflecting the object grouping concepts and real-time service requirements analyzed by referring OMG CORBA specifications and real-time properties. To construct our RTOG platform, we designed the RTOG structure and the functional class diagram of components in a RTOG according to the analyzed requirements, and defined the timing constraints of real-time objects. And we explained the functional definitions and interactions among the components from the following points of view: distributed object management and real-time services. Finally, as results of the implemented our RTOG platform, we showed the real-time executing procedures via the Scheduler and the Real-Time Manager in RTOG platform charging with real-time services, and verified our findings by using the EDF algorithm to see whether real-time services can be applied to our platform.

1 Introduction and Related Works

The distributed object computing is becoming a widely accepted new programming paradigm for distributed applications that require seamless interoperability among heterogeneous environments. The Object Management Group (OMG) had developed the Common Object Request Broker Architecture (CORBA) [1,2] as a standard software specification for distributed environments. This standard software specification specifies an Interface Definition Language (IDL) for the interface descriptions of the functional behaviors of distributed components. It also specifies object services, which facilitate standard interactions with a set of capabilities (i.e. Naming, Event, etc.), and an Object Request Broker (ORB), which is the middleware that allows for the seamless interactions between distributed objects with roles of clients or servers.

Many distributed real-time applications, such as automated factory controls, avionics navigation and military target tracking system, have benefited from a standard architecture like CORBA. Though many designers of distributed applications have been adapting the CORBA specification for developing their own middleware architectures, it is currently inadequate to support requirements for distributed real-time services. For example, the IDL describes the interfaces of functional behaviors of distributed components, but it is impossible to explicitly describe the timing constraints for their real-time behaviors. Furthermore, system services provided by distributed environments offer little support for end-to-end real-time scheduling [3,6] across the environment due to not providing services as synchronized clocks and bounded message latencies.

Recently, the Real-Time Special Interest Group (RT-SIG) [4,7] organized in the OMG has being propelled to examine the current CORBA standard and determine requirements for supporting distributed real-time applications. Especially, this RT-SIG is focusing on supporting the ability to express and enforce the timing constraints by extending the current CORBA standard. The RT-SIG has produced white papers that detail the desired capabilities for a distributed computing environment to support real-time applications. The desired real-time capabilities specified in the CORBA/RT SIG white paper are classified into the following three areas: desired capabilities for the operating environment; desired capabilities for the ORB architecture and desired capabilities for the object services and facilities. But lots of researchers have been done to only improve the system performance by using real-time CORBA itself, or modified a part of CORBA compliance. But, considering the integrated real-time CORBA environments, we still have problems with embedding real-time applications into CORBA implementation. For solving the above problems, it is necessary to extend the current OMG CORBA standard so that it can meet real-time requirements and the desired capabilities of the application without changing the CORBA standard itself. Also, we used the concept of an object group that we have already studied [8–12]. This object group concept was suggested for providing the efficient and convenient management of distributed objects, and for making the individual object independent and reusable. Based on the object group concept, we designed and implemented that a RTOG platform for not only supporting the distributed real-time properties, but also for managing the distributed objects based on the current CORBA standard.

Our paper is organized as follows; Section 2 describes the real-time object group model. Here, we designed the real-time object group (RTOG) and defined the timing constraints of real-time objects. In Sect. 3, we explain the functional definitions and interconnections of the components included in the RTOG from the following services; distributed object management service and real-time service. In Sct.4, we depict how to implement and execute components in RTOGs on our RTOG platform. As a result, we show the executing procedures via Scheduler object and Real-Time Manager in the RTOG charging with real-time services, and verified our findings by using the EDF algorithm to see whether real-time services can be applied to our platform. Finally, we discuss our conclusions and future works.

2 Real-Time Object Group

In a real-time system, first of all, the timing constraints must be met for given real-time applications. Real-time requirements are typically derived from the practical systems interacting with the physical environment. There are two kinds of the relating approaches suggested to adapt real-time services to CORBA. One approach has been producing a CORBA specification or coding implementations that support faster performance. The other approach taken by some vendors has been embedding their CORBA implementations to the real-time operating system [3,5]. But, because both approaches are insufficient for meeting the timing constraints, the supports for the expression and system-wide enforcement of the timing constraints are also necessary. For this reason, we suggested the real-time object group platform (called RTOG platform after this).

2.1 RTOG Structure

Figure 1 below shows the RTOG structure, The RTOG we suggested consists of the following objects; Group Manager (GM), Real-Time Manager (RTM), Object Factory, Security, Scheduler, Timer, Object Information Repository, Service Objects (SOs), and Sub-object groups. Here, the RTOG is logically presented as a single view system for object management service and real-time service of SOs existing in distributed systems connected over networks physically.

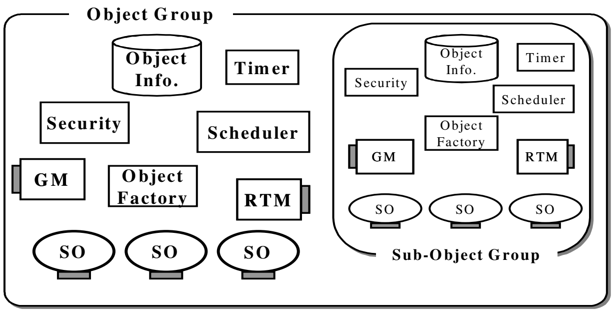


Fig. 1. RTOG structure

We suggest our RTOG for efficient management of distributed objects and the execution of services simply reflecting the real-time properties in distributed environments. And the major role of the components in the RTOG is categorized in two classes to reduce the side effect of interoperability between the object management and the real-time services. First, for supporting the object management services, our RTOG consists of the Group Manager, the Security, the Object Information Repository, and the Object Factory as components.

The Group Manager (GM) is responsible for managing all of distributed objects in group model, and interoperating between a client and target SOs. The Object Factory receives the object creation's request from the Group Manager, creates the requested new object using the information from the Object Information Repository, and then returns the reference of the created object to the Group Manager. The Security Object checks its own object group's access possibility for the request of a client in another group by searching the access control list (ACL) and the access rule. The Object Information Repository stores and maintains the information related to creation of SOs and the subgroup's Group Manager.

Secondly, the components in the RTOG taking care of client requests for real-time service are the Real-Time Manager, the Timer, and the Scheduler. The Real-Time Manager transparently supports the expression of real-time requirements and simply controls complex procedures for service calls binding between distributed objects by interoperating with the Scheduler and the Timer. This Timer has an alarm function for reporting the missing deadline to the requesting SO, without a reply message related to the missing deadline from the requested SO. Finally, the Scheduler executes a given scheduling policy. At this time, the developers can arbitrarily alter an appropriate scheduling mechanism into the Scheduler according to the real-time property of the special-purposed distributed application.

2.2 Timing Constraints

The basic flow of procedures to obtain the result returned from a client's request is broken into three steps; the service request step, the service process step and the result return step. Each step has the timing constraints and must guarantee the timeliness by explicitly outlining the timing constraint's definition. To do this support, we define the five timing constraints 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 SO and SIT(Service Object's Invoked Time), as a time that a SO received a request message of a client.
- A service time (ST) constraint specifies the relative time for the service execution of a SO.
- A transfer time (TT) constraint specifies the relative time required for transferring a request from a client to a SO (SIT-CIT), or inversely.
- A service deadline (SD) constraint specifies the completed service time of a SO as an absolute time.
- A request deadline (RD) constraint specifies the absolute time that a client received the result returning from a SO after a client requested a SO. Here, a RD is the timing constraint that must be guaranteed in a distributed real-time application.

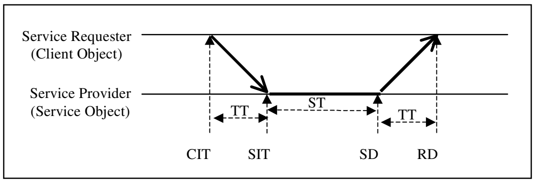


Fig. 2. The definition of timing constraints

Figure 2 shows the definition of the timing constraints. From these timing constraints, we can redefine RD and SD as following expressions;

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

The deadline for service execution (SD) is the sum of the client's invocation time(CIT), the service execution time (ST), transfer time (2*TT) and *slack time*. Here, the slack time is for a constant as a factor for deciding the time of the adequate RD. This service deadline (SD) must be smaller than and equal a time value which subtracts transfer time (TT) from a client's request deadline (RD) for guaranteeing real-time service. In our RTOG platform, we will use the timing constraints mentioned above for applying an Early Deadline First (EDF) algorithm.

3 Interacting with Components in RTOGs

In this section, we describe the interacting procedures between the components in a RTOG or RTOGs on distributed platform from the following points of view; distributed object management and real-time services. And we show interacting procedures and operations via an Event Trace Diagram (ETD).

3.1 Supporting Object Management Services in RTOGs

The Group Manager is responsible for the whole management of the components of the RTOG. The Group Manager can create and withdraw SOs or Sub-object groups in the RTOG itself, and also receive the reference of a SO requested by clients. In this paper, among these object management functions, we will discuss only the procedures for obtaining a SO's reference requested from a client in order to save space. A client must execute the object management procedures firstly before supporting real-time service. These procedures are consisted of following several steps. A client in an Object Group obtains the reference of the Group Manager in another Object Group involving the desired SO via a Name Server. A client requests the SO's reference from the Group Manager using the Object Group's reference obtained before. The Group Manager invokes the Security to decide whether a client can access a SO. After the Security checks the access control list, it returns the access right of the requested a SO to the Group manager. If a client has the access right for a target SO, the Group Manager requests the creation of it from the Object Factory. The Object Factory creates it by accessing information about the requested SO from the Object Information Repository, and then the Object Factory returns the created SO's reference to the Group Manager. Finally, the Group Manager returns the SO's reference from the Object Factory to the client. Figure 3 shows an Event Trace Diagram (ETD) for procedures explained above.

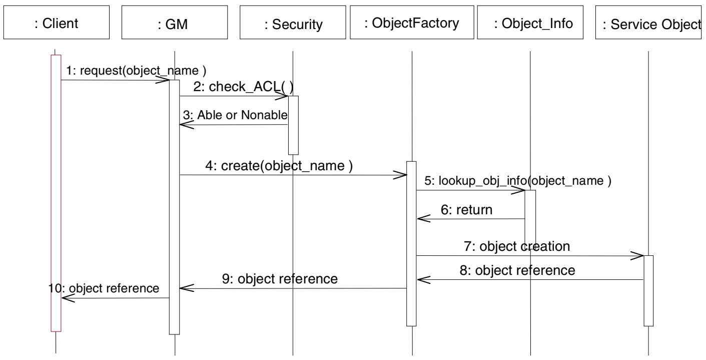


Fig. 3. The ETD for obtaining the Service Object (SO)'s reference

3.2 Supporting Real-Time Services in RTOGs

A client requests the real-time service to the SO with a reference obtained via the object management procedures. Here, as we previously mentioned, the service request of a client is necessary to involve the expression, that is, the RI (Real-time Information structure), about the timing constraints because of the service request with real-time property. In addition, we assume that the Real-Time Manager (RTM) in each object group maintains the global time information using global time service. But, we do not focus on the detailed global time service in our study.

When a client invokes real-time service to a SO in another object group, it passes the deadline information to the RTM in its own object group. This deadline is for an absolute time until the result message is returned. The RTM in a client's object group stores the current global time and the client's request deadline, that is, requested by a client, in RI. The RTM sets a client's deadline to the Timer in its own object group. If this deadline expires, the Timer notifies the missed deadline to a client. After a sequence of procedures mentioned above in an object group including a client is completed, a client requests the real-time service to SO in another object group. At this time, the RI information having the timing constraints stored by the RTM is simultaneously sent with the service call to a SO. A SO passes the RI received to the RTM in its own object group. The RTM calculates the service deadline with consideration to the transfer time, and passes the calculated service deadline to the Scheduler. The Scheduler schedules and decides the priority for the service execution of the client. According to the decided priority, the RTM sets the calculated service deadline on the Timer, and this Timer can also predict and notify the SO with the missed service deadline to a client. With setting the Timer, the RTM sends a client's information to a SO. After a SO performs the services, it returns the executed results to a client, inversely, and it reports the completion of the client's request to the RTM in its own object group. Then the RTM in a SO's object group disarms the Timer. Also, the RTM in a client's object group disarms Timer by getting the result from a SO. Figure 4 shows the Event Trace Diagram for the real-time service's procedures described above.

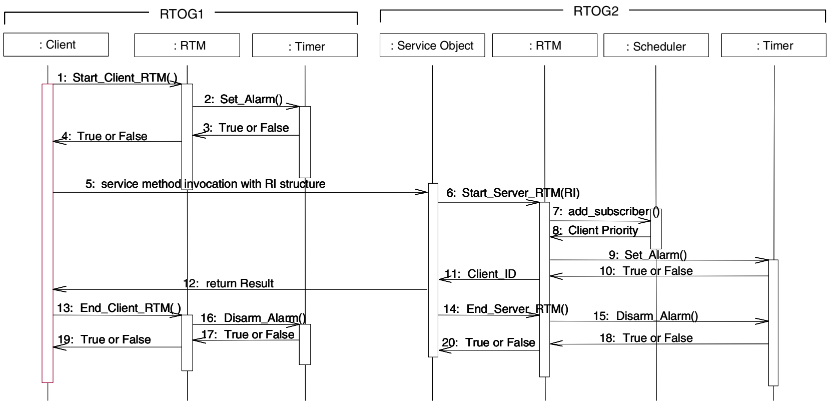


Fig. 4. The ETD for the support of real-time service

To illustrate how the components in our RTOG model work together for supporting a real-time service, the Fig. 5 is shown the expression and the real-time processing procedures along with given timing constraints in RI parameter, as an example of a typical interactions between Client1 and Service Object1 (SO1) by requesting real-time service.

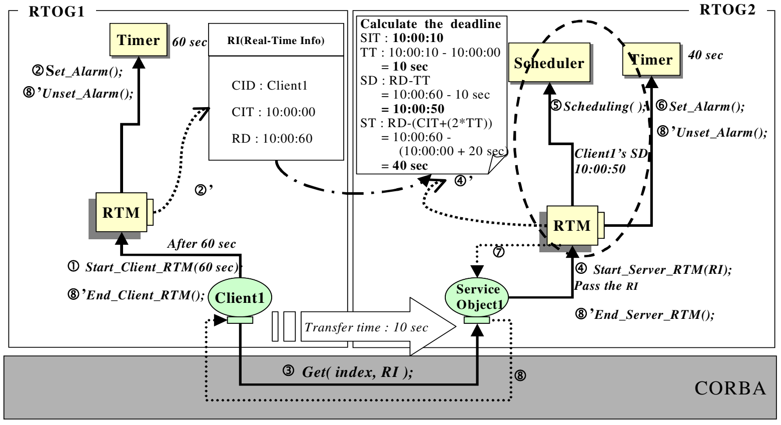


Fig. 5. The example of real-time service in RTOGs

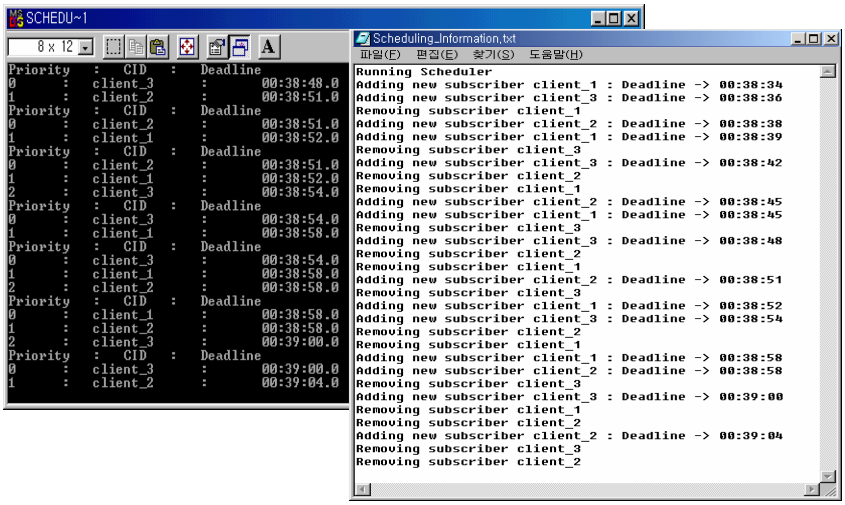
We explain the procedure of real-time service shown in Fig. 5 and time-based numerical analysis from our model using the timing constraints described Sect. 2.2. Here, we assume that for convenient calculations, a unit of time is the second.

- Client1 sends the deadline information (CID, CIT, RD) to the RTM in RTOG1.
- The RTM in RTOG1 sets 60sec on the Timer and stores global time information(CID : Client1, CIT : 10:00:00, client's RD : 10:00:60)

- Client1 requests the real-time service with RI to Service Object1 (SO1) in RTOG2.
- SO1 passes the received RI to the RTM in RTOG2. And the RTM calculates the service deadline (SD) from following expression; SD = RD- TT; Here, when let the transfer time (TT) be 10sec, the calculated service deadline will be 10:00:50 sec. Also, the service time (ST) is for 40 sec.
- The RTM in RTOG2 invokes the Scheduler to decide the priority of a Client1,
- And simultaneously, the RTM sets 40sec on the Timer.
- The RTM sends a Client1's information to a SO1.
- SO1 executes the service requested from Client1 and returns the executing result to Client1. If SO1 finishes the service within the SD (10:00:50), the RTM in RTOG2 disarms the Timer. Also, if Client1 receives the returned result of service within the RD (10:00:60) the RTM in RTOG1 disarms the Timer. If SO1 does not finish the service within the SD or if Client1 does not receive the returned result of service within the RD, the Timers in each object group alarm the missing deadline to Client1.

4 The Execution of RTOG Platform

The components of RTOG were designed and implemented to IDL for independently supporting heterogeneous distributed computing environments, and implemented by using Visi-broker and Visual C++ 6.0 running on Windows. In this section, we will show only a procedure of real-time service. The supports of distributed object group management service have already published as researching papers [9,11,12]. Hence, we will not explain this part.



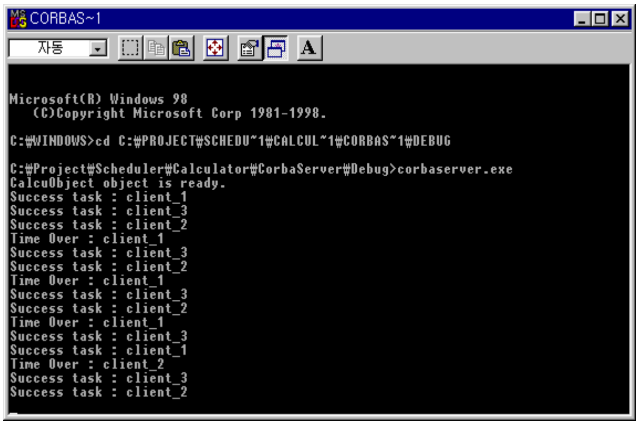
```
Priority  :   CID   :   Deadline
0    :   client_3   :        00:38:48.0
1    :   client_2   :        00:38:51.0
Priority  :   CID   :   Deadline
0    :   client_2   :        00:38:51.0
1    :   client_1   :        00:38:52.0
Priority  :   CID   :   Deadline
0    :   client_2   :        00:38:51.0
1    :   client_1   :        00:38:52.0
2    :   client_3   :        00:38:54.0
Priority  :   CID   :   Deadline
0    :   client_3   :        00:38:54.0
1    :   client_1   :        00:38:58.0
Priority  :   CID   :   Deadline
0    :   client_3   :        00:38:54.0
1    :   client_1   :        00:38:58.0
2    :   client_2   :        00:38:58.0
Priority  :   CID   :   Deadline
0    :   client_1   :        00:38:58.0
1    :   client_2   :        00:38:58.0
2    :   client_3   :        00:39:00.0
Priority  :   CID   :   Deadline
0    :   client_3   :        00:39:00.0
1    :   client_2   :        00:39:04.0
```

```
Running Scheduler
Adding new subscriber client_1 : Deadline -> 00:38:34
Adding new subscriber client_3 : Deadline -> 00:38:36
Removing subscriber client_1
Adding new subscriber client_2 : Deadline -> 00:38:38
Adding new subscriber client_1 : Deadline -> 00:38:39
Removing subscriber client_3
Adding new subscriber client_3 : Deadline -> 00:38:42
Removing subscriber client_2
Removing subscriber client_1
Adding new subscriber client_2 : Deadline -> 00:38:45
Adding new subscriber client_1 : Deadline -> 00:38:45
Removing subscriber client_3
Adding new subscriber client_3 : Deadline -> 00:38:48
Removing subscriber client_2
Removing subscriber client_1
Adding new subscriber client_2 : Deadline -> 00:38:51
Removing subscriber client_3
Adding new subscriber client_1 : Deadline -> 00:38:52
Adding new subscriber client_3 : Deadline -> 00:38:54
Removing subscriber client_2
Removing subscriber client_1
Adding new subscriber client_1 : Deadline -> 00:38:58
Adding new subscriber client_2 : Deadline -> 00:38:58
Removing subscriber client_3
Adding new subscriber client_3 : Deadline -> 00:39:00
Removing subscriber client_1
Removing subscriber client_2
Adding new subscriber client_2 : Deadline -> 00:39:04
Removing subscriber client_3
Removing subscriber client_2
```

Fig. 6. The execution of the Scheduler in a RTOG

To verify the real-time execution procedures of the implemented RTOG platform, Figs. 6 and 7 below are shown the real-time scheduling procedures on the Scheduler and succeeding/missing procedures of a request deadline given from a client on the RTM, when an arbitrary SO is requested by a number of clients over our platform.

In Fig. 6, we display that the priorities of a client have been changed according to an arbitrary given scheduling policy of the Scheduler, as an example. Here we adopted the EDF algorithm as an example of the scheduling policy of the Scheduler. In this executing procedure, the service request of a client having the shorter deadline has the higher priority by EDF policy. We can show two cases deleting clients' requests from the scheduling list in the execution of the Scheduler. One is when a client's request does not complete within a given deadline from a SO's service; deadline missing. The other is when a client's request does complete normally; deadline success. In Fig. 7, we show as the executing procedures on the Real-Time Manager. The deadline success rates of each client's request are 2 per 5 (40%) in a client-1, 4 per 5 (80%) in a client-2, and 5 per 5 (100%) in a client-3, respectively. These results may differ according to each given client's deadline and scheduler policy. In this paper, we only used the EDF algorithm for verifying whether the Scheduler in our model could be operated well correctly, but not showing the improved performance, like decrease of deadline missing rate, of our model itself. Through this verification, we will develop a more flexible prototypical platform that can adopt various real-time scheduling policies to the Scheduler in our model according to given real-time application's properties.



```
Microsoft(R) Windows 98
   (C)Copyright Microsoft Corp 1981-1998.

C:₩WINDOWS>cd C:₩PROJECT₩SCHEDU~1₩CALCUL~1₩CORBAS~1₩DEBUG

C:₩Project₩Scheduler₩Calculator₩CorbaServer₩Debug>corbaserver.exe
CalcuObject object is ready.
Success task : client_1
Success task : client_3
Success task : client_2
Time Over : client_1
Success task : client_3
Success task : client_2
Time Over : client_1
Success task : client_3
Success task : client_2
Time Over : client_1
Success task : client_3
Success task : client_1
Time Over : client_2
Success task : client_3
Success task : client_2
```

Fig. 7. The execution of the real-time manager in a RTOG

5 Conclusions

The distributed object computing technology is becoming a new program paradigm for distributed applications requiring interoperability between heterogeneous systems. Especially, when developing a distributed real-time software, we have to consider the timing constraints for distributed real-time services. Therefore, for the needs of a new paradigms and trends, we suggest the real-time object group (RTOG) platform for providing object group concepts and real-time constraints. This platform supports real-time service at the distributed application level and object management service at the CORBA level, respectively. At the development of this platform, we first surveyed and analyzed studies relating to real-time service models for distributed object-oriented systems. From lots of researches based on the above environment, the existing researches have been attempting to only improve the performance of systems by using real-time CORBA, or modifying the part of CORBA compliance. Up to now, this kind of a new platform agreed to our goal has not ever tried to be developed yet.

Hence, we designed a new model of a real-time object group platform that can support the real-time requirement without modifying the ORB. In this paper, we defined the roles of the components of a RTOG, and described functions of each component and their relationships of interoperability between the components including in an object group. To verify the executing procedures of the implemented real-time object group platform, we simulated the processing procedures for the real-time service execution requested by several clients in our platform. As final results, we showed that our real-time object group platform is able not only to manage distributed objects and/or object groups, but also to guarantee the real-time requirements of distributed real-time applications. In this paper, we only used the EDF algorithm for verifying whether the Scheduler in our model could be scheduled correctly, but not showing the improved performance of our model itself.

In future, using this RTOG model, we will develop more flexible prototypical platform that can variously adopt new real-time scheduling policies to the Scheduler in our model according to given real-time application's properties and support general-purpose real-time services, not special-purpose real-time services. And then we want to verify the execution power of the distributed real-time application on our platform via various simulations for improving real-time service.

Acknowledgements. 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. OMG Real-time Platform SIG: Real-time CORBA A White Paper-Issue 1.0. http://www.omg.org/realtime/real-time_whitepapers.html (1996)
2. OMG: The Common Object Request Broker: Architecture and Specification revision 2.2. http://www.omg.org/corba/corbaCB.htm (1998)
3. Victor Fay Wolfe. (eds.): Expressing and Enforcing Timing Constraints in a Dynamic Real-Time CORBA System. http://www.cs.uri.edu/rtsorac/publication.html (1997)
4. OMG TC: Realtime CORBA Extensions: Joint Initial Submission. OMG TC, Document orbos/98-01-09 (1998)
5. Nguyen Duy Hoa: Distributed Object Computing with TINA and CORBA. Technical Report Nr. 97/7. http://nenya.ms.mff.cuni.cz/thegroup (1997)
6. Scheduling in Real-Time System. http://www.realtime-os.com/sched_o3.html (1996)
7. Andrew S. Tanenbaum: Distributed Operating Systems. Prentice Hall (1995)
8. S.C. Joo: A Study on Object Group Modeling and Its Performance Evaluation in Distributed Processing Environments. Final Researching Report. ETRI (1997)

9. G.M. Shin, M.H. Kim, 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) 1241–1252
10. B.T. Jun, M.H Kim, 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) 3651–3667
11. M.H. Kim, S.C Joo: Construction of CORBA Object-Group Model for Distributed Real-Time Service. Journal of The Korea Information Science Society, Vol. 7, No. 6 (2001) 602–613
12. M.K. Kim, C.W Jeong, C.S Shin, S.C Joo: Design and Implementation of Distributed Object Management Model. In Proceedings of IASTED International Conference-Parallel and Distributed Computing and Systems. Anaheim CA, U.S.A (2001) 222–227