Real-Time CORBA Development at MITRE, NRaD, Tri-Pacific and URI*

Gregory Cooper cooper@cs.uri.edu University of Rhode Island, USA
Lisa Cingiser DiPippo cingiser@cs.uri.edu University of Rhode Island, USA
Levon Esibov esibov@cs.uri.edu University of Rhode Island, USA
Roman Ginis ginis@linus.mitre.org MITRE Corporation, Bedford, MA USA
Russell Johnston russ@nosc.mil U.S. Navy NRaD, San Diego, CA USA
Peter Kortman peter@tripac.com Tri-Pacific Software, Alameda, CA USA
Peter Krupp pck@mitre.org MITRE Corporation, Bedford, MA USA
John Mauer mauer@mitre.org MITRE Corporation, Bedford, MA USA
Michael Squadrito squadrit@cs.uri.edu University of Rhode Island and MITRE, USA
Bhavani Thuraisingham thura@mitre.org MITRE Corporation, Bedford, MA USA
Steven Wohlever wohlever@mitre.org MITRE Corporation, Bedford, MA USA
Victor Fay Wolfe wolfe@cs.uri.edu University of Rhode Island and MITRE, USA

Abstract

This paper describes Real-Time CORBA results and work in progress stemming from a collaborative effort involving researchers at MITRE, NRaD, Tri-Pacific Software and The University of Rhode Island.

1 Introduction

This paper is a summary of the Real-Time CORBA development being performed by a team from MITRE Corporation, the US Navy NRaD facility, The University of Rhode Island (URI), and Tri-Pacific Software with support from Iona Technologies. Our team members from each of these organizations are leaders in the Object Management Group's Real-Time Special Interest Group (SIG), the group that is establishing the standard for Real-Time (RT) CORBA. In addition, our team has performed collaborative research and development of techniques to implement real-time capabilities within the CORBA framework.

This paper presents a brief overview of RT CORBA in Section 2. Section 3 then summarizes two parts of our project - a "static" RT CORBA system implemented on the widely-available ILU ORB from Xerox Parc, and a "dynamic" RT CORBA system implemented as an extension to the commercial Orbix system from Iona Technologies. Section 4 then describes our work in progress, which features an extension to the commercial PERTS real-time analysis tool for analyzing real-time behavior of RT CORBA systems, an "open ORB" approach to allow configuring middleware to meet RT needs, and applications of RT CORBA to military programs.

*This work is partially supported by the U.S. Office of Naval Research grant N000149610401.

2 Real-Time CORBA

RT CORBA is being defined by the OMG RT SIG. The essence of its definition is:

> *RT CORBA deals with the expression and enforcement of real-time constraints on end-to-end execution in a CORBA system.*

There are two main categories of RT CORBA requirements: requirements on the operating environment (operating systems and networks); and requirements on the CORBA run-time system. Requirements on the operating environment include: synchronized clocks, bounded message delay, priority-based scheduling, and priority inheritance in synchronization primitives. Requirements on the CORBA run-time system include: transmittal of real-time and quality-of-service requirements with CORBA requests, scheduling based on a global uniform notion of priority, priority queueing at all CORBA services, real-time events that are prioritized and carry the time of their occurrence, priority inheritance in CORBA synchronization, and real-time exceptions. The requirements are fully described in the RT SIG's whitepaper [4] and summarized in the RT CORBA overview presented at the IEEE Real-Time Applications Symposium in June of 1997 [8].

In September 1997, the first Request For Proposals (RFP) to establish the standard for fixed priority RT scheduling was approved by the RT SIG and other governing parties of the OMG. The RFP can be found on the OMG's Web site at `www.omg.org`. The RFP for dynamic RT scheduling is close to completion and is expected to be released within six months. Based on the expected responses to the RFPs from companies that have actual products, the RT CORBA standard will be established by the OMG.

3 Our RT CORBA Development

Our team took two approaches to developing techniques for RT CORBA systems: a fixed priority system implemented using the ILU ORB from Xerox Parc, which was done primarily at MITRE; and a dynamic scheduling system implemented using Orbix from Iona, which was done primarily at URI and NRaD and has been transitioned to Tri-Pacific and Iona. The efforts were coordinated and several team members worked on both projects.

3.1 Dynamic RT CORBA

Our dynamic RT CORBA system is designed to meet the requirements of the RT SIG whitepaper. It is implemented on Sparc workstations running Solaris 2.5 (with POSIX threads) using Iona Technology's Orbix_2.0.1MT(multi-threaded) as the CORBA baseline. All of our prototype software assumes an operating environment that is compliant with the POSIX real-time operating system standard. This environment satisfies most of the RT CORBA white paper operating environment requirements. Figure 1 shows a CORBA-style diagram of our RT CORBA system.

Our Dynamic RT CORBA system consists of a new CORBA service for Global Priority, and modifications of existing CORBA services for Real-Time Events, and Real-Time Concurrency Control. Our RT CORBA system also includes several new CORBA Interface Definition Language (IDL) types for expressing real-time parameters, along with library code that is added to clients and to servers. Together these components allow *timed distributed method invocations* (TDMIs) [7]. A TDMI is a client's invocation on a CORBA server that expresses real-time constraints for the CORBA run-time system to enforce.

Timed Distributed Method Invocations. A TDMI includes a structure with attributes that currently include *importance*, *deadline*, and *period*, but other real-time and quality-of-service parameters can also be added. Our RT CORBA run-time system attaches this structure to the client's TDMI request so that the CORBA run-time system can use it to enforce the TDMI requirements at all places in its end-to-end path. For instance, the CORBA Name Service would queue requests based on priority derived from this structure, and the CORBA server would execute the invoked method based on priority derived from the structure. All parts of our RT CORBA run-time system examine this structure to acquire information necessary to enforce the expressed real-time requirements by doing things such as establishing priority and setting alarms.

Once the real-time parameters are established, our RT CORBA run-time system uses library code to enforce the implied constraints by setting a dynamic *transient priority* for the TDMI and by setting operating system alarms to detect when crucial times, such as deadlines, have arrived. A transient priority is a single integer that is derived by our new RT CORBA Global Priority Service based on the information in the TDMI structure. The Global Priority Service ensures that the transient priority is meaningful relative to all other transient priorities in the RT CORBA system as described next.

Global Priority Service and Distributed Real-Time Scheduling. Real-Time scheduling is performed by the our CORBA run-time system in cooperation with other components, such as the local real-time operating systems' schedulers. This scheduling is based on a transient priority. The transient priority is only valid while the TDMI is active. That is, the client and all execution on its behalf assume the transient priority during the execution of the TDMI, but the client resumes a previous priority when the TDMI completes. Schedulers and queues throughout the distributed RT CORBA system, such as RT POSIX priority-based operating system schedulers, use these transient priorities to order all execution that is associated with a TDMI.

The transient priority for each TDMI is established by our RT CORBA Global Priority Service. This service uses a uniform function (uniform for all clients and servers in the system) to compute transient priority as a function of the attributes in the structure associated with the TDMI. The function implementation in our prototype orders priorities based on the *importance* attribute first, and then based on the *deadline* attribute - essentially establishing a global *earliest-deadline-first within importance level* priority assignment throughout our RT CORBA system. Changing the calculation of transient priorities based on other scheduling policies, such as global rate-monotonic pri-

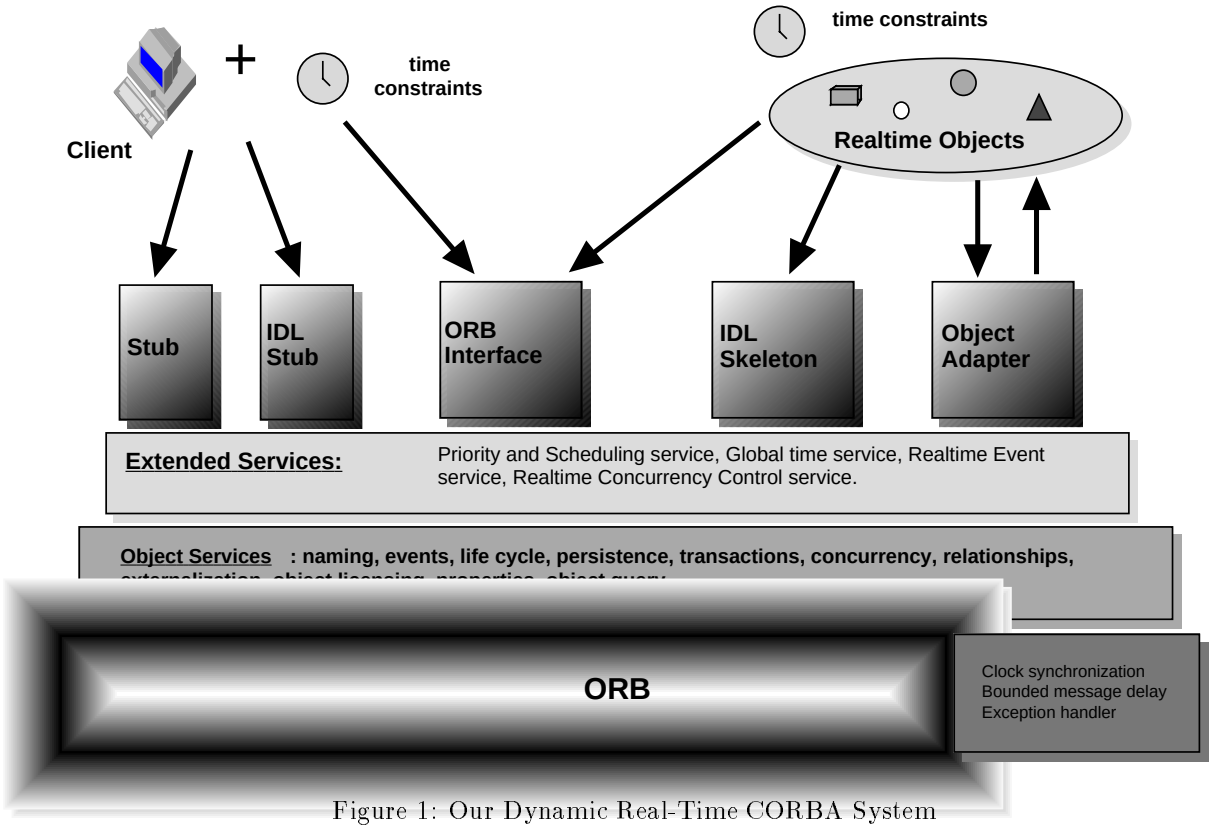

Figure 1: Our Dynamic Real-Time CORBA System

ority assignment, is facilitated by the function's central implementation in the RT CORBA Global Priority Service.

The implementation of the Global Priority Service in our prototype is accomplished through a combination of library code and a *RT Daemon* process running on each node. The library code calculates the transient priority. The RT Daemon on each node maps the transient priority to the priorities available on the local real-time operating system. In our prototype, which uses RT Solaris operating systems with 60 local priorities, the RT Daemon must map the (wide) range of transient priorities into the 60 local priorities. The mapping is done by using a statistical model of the likely deadlines and calculating transient priorities such that TDMIs are probabalistically evenly distributed among the local priorities. For example, if there were 60 TDMIs on a Solaris node, the mapping would ensure the highest probability of each TDMI being at a unique priority.

Additionally, the RT Daemon enforces *aging* of transient priorities. Aging is the process of increasing priority as time goes on, which is necessary in dynamic earliest-deadline-first scheduling. The RT Daemon keeps track of the transient priorities on its node. The RT Daemon increases a TDMI's transient priority if, due to the passage of time, the TDMI's priority is too low (in an EDF ordering of priorities) compared to a newly-arrived TDMI. The aging facility can be "turned off" for real-time scheduling policies that do not require aging, such as a static rate-monotonic-based policy.

RT Exceptions. Our RT CORBA run-time system also augments the CORBA exception mechanism to handle real-time exceptions. These exceptions are derived from the attributes set in the TDMI structure such as a deadline and/or a period. Our prototype implements the mapping of deadline exceptions to CORBA exceptions by catching the signal sent by the POSIX alarm that was set for deadline and then raising the CORBA exception in library code.

Real-Time Event Service. The current CORBA event service allows for the exchange of named events in the CORBA system. For instance, a client might synchronize with another client by waiting for it to generate a CORBA event. Our RT CORBA system has implemented a modified *RT Event Service* that prioritizes the delivery of events and delivers the time

that the event occurred. Prioritized events are based on the transient priorities of the producers and consumers and are important to maintain global real-time priority scheduling. Delivery of the (global) time of the event occurrence is important to allow events to be used to establish timing constraints relative to them.

RT Concurrency Control Service The existing CORBA standard specifies a Concurrency Control Service that is used to maintain consistent access to servers. Our prototype RT CORBA system includes a RT Concurrency Control Service that implements *priority inheritance* [3]. When a TDMI requests a lock on a resource from the RT Concurrency Control Service, the TDMI transient priority is compared to those of all TDMI's holding conflicting locks on that resource. Conflicting clients with lower priorities are raised to the requesting TDMI's priority, and the requesting TDMI is suspended. Whenever a lock is released, the releasing TDMI resets its priority to that of the highest priority TDMI it still blocks (this is possible since clients can hold several locks of different types). If it no longer blocks any higher priority TDMIs, then the releasing TDMI is reset to its original priority. Finally, the highest priority blocked TDMI that can now run is allowed to obtain its lock and continue execution.

3.2 Fixed Priority RT CORBA

Team members from MITRE developed a RT CORBA system on networked Intel-based computers under the LynxOS 2.4 real-time operating system, with a baseline CORBA system derived from SYLU, a Python implementation of Xerox's ILU ORB developed by Scott Hassan at Stanford University [6, 1]. SYLU is the only CORBA ORB with complete source code available that will execute on a real-time operating system. Our effort first identified requirements for the use of real-time CORBA in command and control systems. Our real-time ILU ORB provides a distributed scheduling service supporting rate-monotonic and deadline-monotonic techniques. The resulting infrastructure combines a POSIX-compliant real-time operating system, a real-time ORB, and an ODMG-compliant real-time object-oriented database [1].

We are currently incorporating fixed priority techniques into the Orbix version of our RT CORBA system as well. This system assumes periodic CORBA clients, intermediate TDMI deadlines within the period, and known execution time of CORBA services. The Global Scheduling Service will assign fixed static priorities based on a Deadline Monotonic policy. All synchronization in the RT CORBA system will use the Distributed Priority Ceiling protocol (DPCP) [3]. These policies were chosen to facilitate real-time analysis of the RT CORBA system as described in Section 4.

4 Current Work

The versions of RT CORBA from Section 3 have been delivered to vendors including Lockheed/Martin, Iona, and Tri-Pacific for full development into RT CORBA products. This section briefly describes our current research and development of RT CORBA.

4.1 Open ORB

A part of our current approach, centered at MITRE, is to design an *open* RT CORBA system using a *meta-object and design pattern approach* where the CORBA run-time system itself is made of object components (meta-objects) that can be configured to tailor the system to various real-time requirements. A major criticism of existing commercial middleware is that it dictates particular implementation decisions that may not be appropriate for all applications. Some aspects of the middleware may not deliver the required performance or may lack the necessary functionality. For example, we found that the current commercial CORBA implementations fail to fulfill many military real-time requirements. Because the middleware is often implemented as a black box, it allows little or no flexibility in adapting it to the application.

A conceptual diagram of our approach is shown in Figure 2. Note the existence of meta-objects in the ORB that allow its configuration to meet real-time requirements.

Another key part to our approach is the development of a *reflective ORB*, which monitors its own behavior and performance and changes based on that information. We are developing an adaptive system that is also reflective, such that it would be capable of adapting dynamically with the changes in real time. For example, if a selection of resources of different quality were available (e.g. a set of radars), a reflective system would be able to choose different radars depending on the application as well as additional user constraints imposed on a specific operation (e.g., timing constraints on getting current information about a contact). In our system there would be a metalayer (set of meta-objects) whose job it is to monitor the rest of the objects in the system to determine whether or not they are meeting their QoS requirements. Depending on the result, the metalayer might choose a different set of objects (that represent groups of resources) to handle certain tasks, sacrificing something (time vs. quality) in order to get back a result within its specified parameters.

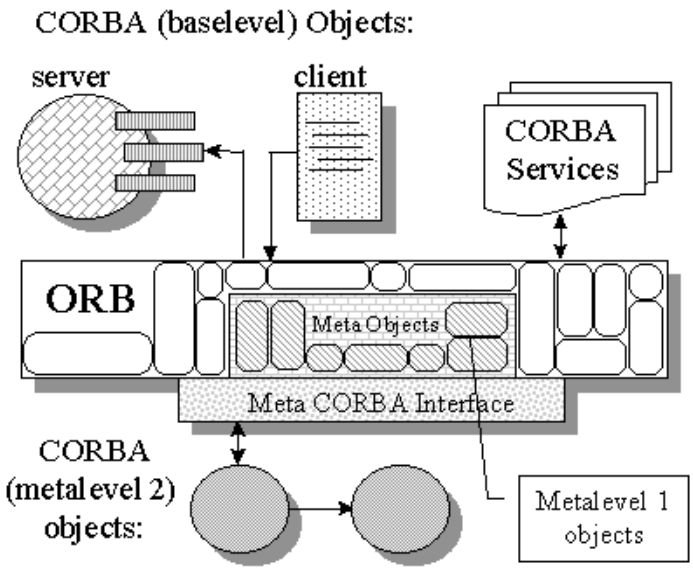

Figure 2: An Open RT CORBA System With a Meta-Object Architecture

The use of meta-object protocols and reflective techniques will make our open ORB tailorable to demanding real-time and fault-tolerance requirements. The components of the system are the ORB, services such as scheduling and data management and the application frameworks. Each component consists of various subcomponents and protocols. The idea is to switch the subcomponents and protocols without disrupting the operation of the system. For example, one could switch the scheduling protocols or the transport protocols or even the application components.

Our model consists of objects in the ORB, distributed objects (CORBA objects) and CORBA Services. We classify these objects into three levels, called *metalevels*. All the objects in the implementation of the ORB, services and facilities are meta-objects of metalevel 1. Objects in the ORB and distributed objects that control the execution of baselevel objects and metalevel 1 objects (through a meta interface) are meta-objects of metalevel 2. Consider, an example with Protocol substitution in a CORBA ORB. If Protocol is an object inside the ORB, it is a metalevel 1 object according to the definition above. A metalevel 2 object for the Protocol could be an object that controls which instance of Protocol is currently active - e.g. Sun RPC or HTTP. Another example could be the following: an object inside the Concurrency Control service is a metalevel 1 object. An object that controls which CC algorithm is used in a particular situation (e.g. priority ceiling, two-phase locking, etc.) would be a metalevel 2 object. The enumeration of metalevels is useful for establishing the control relationship between sets of objects. Figure 3 shows a conceptualization of how a meta-object switch can be used to switch from a typical first-come CORBA Naming Service to a Real-Time Load Balancing Naming Service.

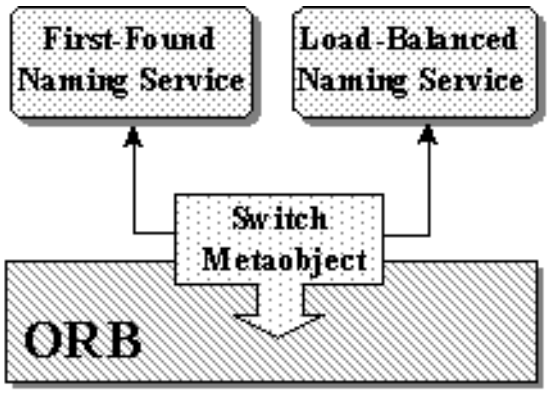

Figure 3: A Meta-Object Switchable CORBA Naming Service

We will integrate our earlier work on both static and dynamic real-time CORBA systems into this open ORB framework to allow flexible middleware that can be configured to the many shades of real-time.

4.2 RT CORBA PERTS Analysis Tools

We are modifying the PERTS Real-time analysis tool from Tri-Pacific Software (see [5]) to perform analysis for RT properties, such as whether a static RT CORBA system is schedulable. PERTS was originally developed at the University of Illinois and commercialized by our team members at Tri-Pacific to perform a rich set of schedulability analyses on RT systems. Our initial version of PERTS for RT CORBA will analyze fixed priority global RT CORBA scheduling, following the capabilities specified in the OMG RFP for fixed priority RT CORBA.

The PERTS RT CORBA tool will provide a graphic user interface to input characteristics of the RT CORBA system including periods of RT CORBA clients, intermediate client TDMI deadlines, network delay, context switch time, characterization of RT CORBA servers, execution time of CORBA methods, and execution time of client code. Originally the RT CORBA PERTS tool will assume the use of Distributed Priority Ceiling Protocol (DPCP - called "MPCP" in PERTS) for synchronization in the CORBA system and global Deadline Monotonic scheduling in the CORBA Scheduling Service, both of which we are implementing into our static RT CORBA

system in parallel. The tool will indicate whether the RT CORBA system is schedulable and, if so, produce priority assignments that can be used by the RT CORBA Scheduling Service.

4.3 RT CORBA Applications

Another important part of our current work is to evaluate the new RT CORBA technology on actual military applications. At MITRE, we have used the static RT CORBA system in a target tracking application from the US Airforce AWACS system. At NRaD, we are currently using the dynamic RT CORBA system in the JFLEX and C2MUV military planning applications [2] to coordinate distributed real-time planning among various parties from various military branches.

5 Conclusion

This paper has presented a summary of past and current work being done by a collaboration of researchers at MITRE, NRaD, Tri-Pacific Software, and the University of Rhode Island. We described static and dynamic prototype RT CORBA systems that we have developed using techniques such as a new CORBA Global RT Scheduling Service. Many of the results of our efforts are reflected in the current Whitepaper and Request For Proposals that have come from the OMG RT Special Interest Group.

Our new efforts in open ORB design and RT CORBA PERTS analysis tools reflect the next step for us, for the RT CORBA development by vendors, and for standardization at the OMG. Lessons from our two initial prototypes have led to the meta-object approach in our open ORB design so that we have a framework to incorporate both static and dynamic real-time components in a systematic way. The open approach also allows for other configurations, like the insertion of new RT network protocols, that better allow system designers to tailor their systems to the wide range of requirements found in complex real-time applications. Our initial RT CORBA efforts also allowed us to identify the features of RT CORBA that were feasible to implement while still allowing analysis by our modified PERTS tool. By designing RT CORBA and its analysis tool in parallel, we expect to achieve a real-time middleware solution that is supported by commercial vendors and, in many useful cases, has analyzable real-time behavior.

Web Sites. Additional Information on these projects can be found at:

- `http://www.cs.uri.edu/rtsorac`
- `http://atticus.nosc.mil/dhda`
- `http://www.tripac.com`

References

[1] E. Bensley, P. Krupp, et. al. Object-oriented approach for designing evolvable real-time command and control systems. In *Proceedings of the Workshop on Object-Oriented Real-Time Dependable Systems*, Feb. 1996.

[2] `http://da5id.nosc.mil`

[3] Ragunathan Rajkumar. *Synchronization in Real-Time Systems: A Priority Inheritance Approach.* Kluwer Academic Publishers, Boston, MA, 1991.

[4] The Realtime Platform Special Interest Group of the OMG. CORBA/RT white paper. ftp site: ftp://ftp.osaf.org/whitepaper/Tempa4.doc, Dec 1996.

[5] `http://www.tripac.com`

[6] B. Thuraisingham, P. Krupp, A. Schafer, and V. Wolfe. On Real-Time Extensions To The Common Object Request Broker Architecture. In *Proceedings of the Object Oriented Programming, Systems, Languages, and Applications (OOPSLA) Workshop on Experiences with CORBA*. Oct. 1994.

[7] V. Wolfe, J. Black, B.Thuraisingham, and P. Krupp. Towards Timed Distributed Method Invocations. In *The Proceedings of the Fourth International High Performance Computing Conference.* December 1995.

[8] V Fay Wolfe, L. Cingiser DiPippo, R. Johnston R. Ginis, M. Squadrito, S. Wohlever and I. Zykh Real-Time CORBA In *Proceedings of the Third IEEE Real-Time Applications Symposium*; June 1997.