CORBA/*e*, NEW CORBA PROFILES FOR SDR

Victor Giddings (Objective Interface Systems, Herndon, VA, USA;
victor.giddings@ois.com)

ABSTRACT

This paper will present the rationale behind CORBA/*e* and how it replaces Minimum CORBA for SDR development. It will also review the technical standards of each CORBA/*e* profile and discuss how each profile applies to developing SCA-based radios. The JTRS JPEO has indicated its desire to use CORBA/*e* to replace Minimum CORBA for the next version of the SCA.

The CORBA/*e* (CORBA for embedded) profiles were driven by user demand for smaller size and greater speed and performance, especially in the defense and telecommunications industries.

CORBA/*e* starts with assumptions derived from experience in the pathfinding SDR developments:

- Resources are not infinite. In fact, they may be very constrained.
- Features are needed to propagate priorities across the network and deal with networked resources consistently.
- Network priorities must be managed in real-time, respecting scheduling, deadlines, priorities, etc.
- Changes to one system should not require re-writing code for all systems on the network.

CORBA/*e* offers an architectural solution to keep up with the rapid pace of technological change – in processors, models, and particularly communications bus types. CORBA/*e* lets developers protect their investment in development work despite rapidly accelerating change.

The OMG has merged the interoperability of standard CORBA with the reliability and deterministic execution of Real-time CORBA into a pair of specification profiles – Compact CORBA/*e* and Micro CORBA/*e* – that meet the middleware needs of the SDR community:.

- Compact CORBA/*e* fits easily on a typical 32-bit microprocessor, running a standard Real-Time Operating System (RTOS); these systems may run such applications as signal or image processing with real-time dependability.
- Micro CORBA/*e*, even smaller, fits on the kind of low-powered microprocessor or high-end DSP found on mobile or hand-held equipment.

CORBA/*e* systems are compact, fast, and reliable: Freed of the dynamic aspects of standard CORBA, including unpredictable response times and unlimited potential memory usage, these systems provide real-time execution in a small footprint that fits easily onto either board-based or chip-based systems.

CORBA/*e* systems are fully interoperable and support the OMG's mature interoperability standards: GIOP (General Inter-ORB Protocol) and IIOP (Internet Inter-ORB Protocol).

1. INTRODUCTION

One of the most sensitive and demanding applications of CORBA is in military communications. In high-risk combat environments, radio often provides the only means of communication. With the vast number of incompatible and aging legacy devices in the field, ensuring interoperability among different types of field-based radios operating at different frequencies is mission-critical.

The Joint Tactical Radio System (JTRS) was initiated by the Department of Defense to provide a flexible new approach to meet diverse warfighter communications needs – through high-assurance software programmable radio technology, or software-defined radio (SDR). SDR uses embedded CORBA as the communications architecture for the entire radio. CORBA/*e* is a perfect architecture for SDR because it ensures intra- and inter-network compatibility that is transparent to the radio operator.

Real-time CORBA-based SDR systems ensure the most effective, reliable radio system possible for military and aerospace use today, and for flexible and secure wireless communications for commercial markets in the future. Commercial middleware solutions supports the CORBA/*e* standard and offer a commercially available SDR platform that enables interoperability through software modifications, not hardware changes. As a result, future software-defined radios will be interoperable much like the international phone system where equipment communicates to each other with standard protocols regardless of manufacturer.

2. WHAT IS CORBA/*e*?

For systems that need small memory footprint and deterministic execution, embedded developers can use the

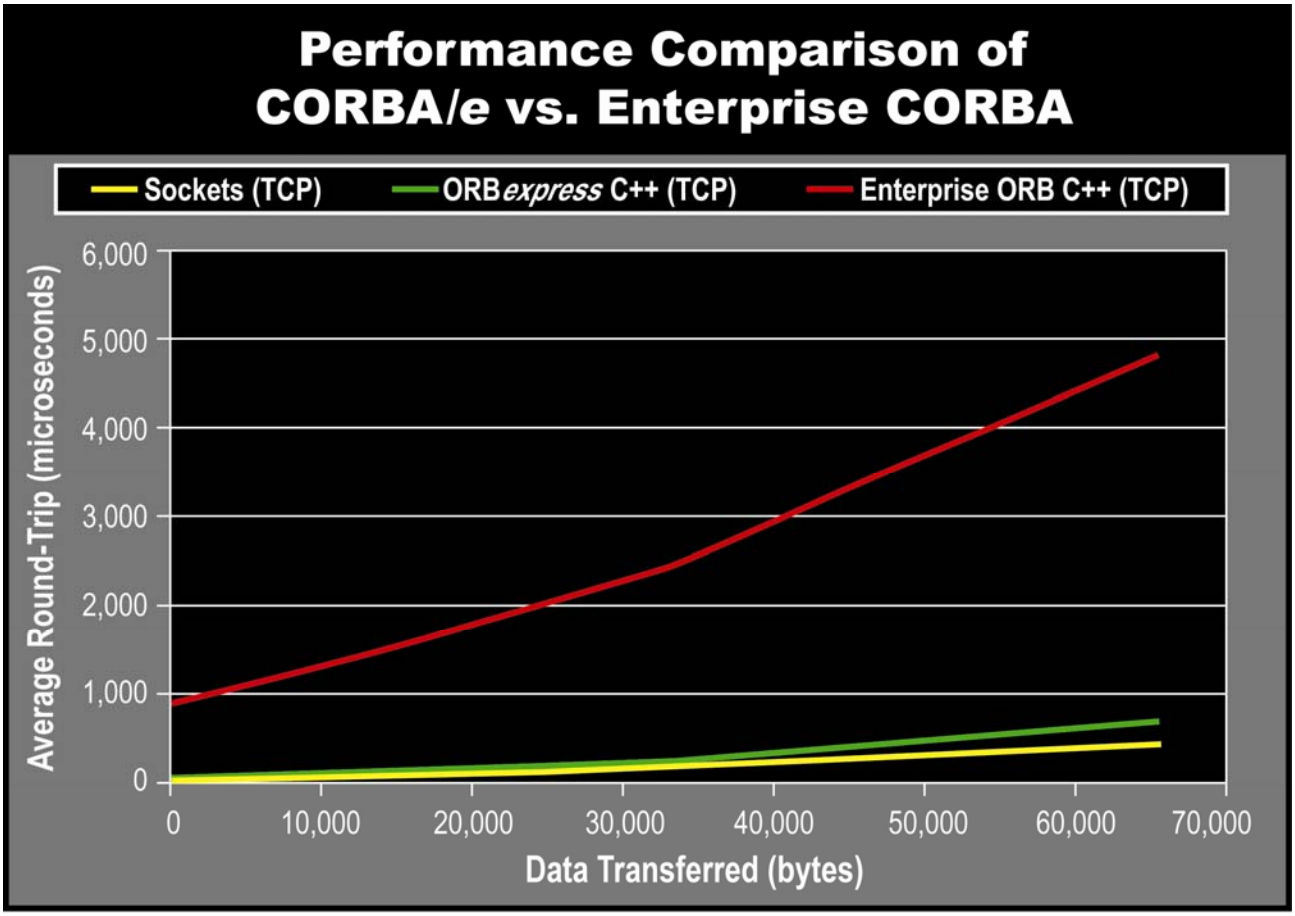
latest generation of CORBA: CORBA/*e* (CORBA for embedded). An Object Management Group (OMG) standard, CORBA/*e* provides an architecture for distributed processing that fits systems from the largest server farms to the smallest networked Digital Signal Processors (DSPs).

The OMG has merged the static aspects of industry-standard CORBA with the essential Real-time CORBA features into two new profiles grouped under the banner of CORBA/*e*. The CORBA/*e* Compact Profile fits easily on a typical 32-bit microprocessor, running a standard Real-Time Operating System (RTOS); these systems may run such applications as communications, signal or image processing with real-time dependability. The CORBA/*e* Micro Profile is even smaller and fits on the kind of low-powered microprocessor or high-end DSP found on mobile or hand-held equipment. As with standard CORBA, CORBA/*e* provides the benefit of code reuse. Developers do not need to re-write code for all systems on a network when they want to make changes, thus preserving their investment in existing applications.

3. WHY CORBA/*e*?

Embedded system software development is an expensive and time-consuming task. But with a sound middleware architecture, this investment can pay dividends across many generations of technology. For developers of real-time and embedded systems, CORBA/*e* is ideally suited to the challenges of today's mission-critical environments:

- Standalone systems are a thing of the past. Embedded processor environments are networked and highly interconnected. Software must cope with communications and interoperability issues, while delivering the same reliability and performance as the isolated embedded systems of the past. Even systems that appear to be standalone need a communications infrastructure to merely report their status. CORBA/*e* provides easy access to a variety of sophisticated transports including Shared Memory, Rapid IO and Firewire, as well as Ethernet.
- Platform flexibility is essential. Embedded software development is no longer confined to a specific processor model on a particular board. A developer may need to support many different processors at the same time, or migrate to new environments as the initially targeted compilers, operating systems and processors change. There is increased pressure to preserve investment through development of "reusable product lines." Developers writing their own infrastructure are more susceptible to increased cost due to code changes accommodating inevitable environment changes. CORBA/*e* insulates embedded developers from the headaches of rewriting code with every processor and system change.
- Multicore Processors. Embedded systems will increasingly use multicore processors. Developers will need to migrate their applications from single core processors to processors containing two or more cores. CORBA/*e* enables a seamless migration to multiple cores through the benefits of location transparency.
- Embedded systems interact with the real world in real-time. Devices require interactions that are predictable in time as well as in function. CORBA/*e* provides distributed predictability by recognizing and propagating priority in its own processing and across the system.
- Power, weight, size and speed are constrained. CORBA/*e* is specifically designed to support board-based and networked systems with the smallest footprint and the highest performance requirements. CORBA/*e* focuses on providing just the most useful features of previous versions of CORBA, paring away those that bloat footprint and processing requirements.
- Reliability must be built-in. Real-time and embedded developers are rightly skeptical about adopting code they don't write themselves. Robust, implementations of CORBA/*e* like ORBexpress RT, field-tested in the harshest environments, have proven their reliability time and time again. Portability among implementations assures the longevity of the solution.
- Application flexibility through refactoring. Developers can build a CORBA/*e* application as if it were a standalone application. They can then spread and re-spread the application logic across multiple resources in a way that doesn't create any extra work. Deferring deployment decisions enables optimized resource allocation and the flexibility to adjust to changing conditions.
- Time to market is critical. Economic pressures are requiring greater productivity from systems development. By supplying a high-performance communications framework, CORBA/*e* enables greater productivity because it is no longer necessary for developers to write their own protocols. CORBA/*e* solves the tedious and time-consuming part of distributed applications by establishing a reliable, flexible architecture.



Source: Lockheed Martin's Advanced Technology Laboratories

Figure 1 shows the minimal latency added by a robust CORBA/*e* implementation compared to a traditional enterprise ORB.

4. CORBA/*e* PROFILES

The CORBA/*e* profiles – Compact and Micro – package the static features of CORBA middleware and real-time capabilities into a small footprint. Figure 2 shows the relationship between the CORBA/*e* profiles and standard CORBA. Designed by the most experienced providers of Distributed, Real-time and Embedded (DRE) software, the two profiles fulfill the range of industry requirements.

CORBA/*e* Compact Profile

CORBA/*e* Compact Profile merges the key features of standard CORBA 2.6 and Real-time CORBA into a powerful yet compact middleware package that interoperates with other CORBA clients and servers of every scale, executes with the deterministic characteristics required of a true real-time platform, and leverages the knowledge and skills of software development teams through industry-standard architecture.

Shedding the dynamic aspects of CORBA and support for the CORBA Component Model (with their unpredictable response times and unlimited potential memory usage), CORBA/*e* Compact Profile retains interoperability by retaining the full IIOP protocol. It also preserves many server-side implementation options through the Portable Object Adapter (POA) policies, and a rich type model that includes a lightweight version of "Valuetypes" and type Any.

CORBA/*e* Micro Profile

CORBA/*e* Micro Profile shrinks the footprint even more, small enough to fit low-powered microprocessors or DSPs. This profile further trims features and constrains options that increase footprint, such as the Valuetype, the Any type, most of the POA options, and all of the real-time functions except the real-time Mutex interface. In exchange for these limitations, the profile defines a CORBA executable that vendors have fit into only tens of kilobytes of memory – small enough to fit onto a high-end DSP or microprocessor on a hand-held device. Even at this small size, the CORBA/*e* Micro Profile retains full IIOP interoperability.

5. CORBA/*e* PERFORMANCE

One of the most important determinants of performance is the ORB used. ORB performance may differ by multiple orders of magnitude in performing the same operation. High-performance ORBs avoid unnecessary context switches and minimize data copying to optimize both latency and throughput. ORBexpress has been shown to be the performance leader in ORBs available today.

CORBA has been evaluated by a variety of independent benchmark tests, including those conducted by Lockheed Martin's Advanced Technology Laboratories: "During our extensive performance testing of CORBA ORBs and during our evaluations of other forms of communications middleware, we concluded that CORBA can be very small, extremely fast and fully optimized for the industry's highest-performance

TECHNICAL FEATURES OF CORBA/*e*

CORBA/e Compact Profile:

- Compact yet powerful: Fits resource-constrained systems (32-bit processor running a RTOS), but supports sophisticated applications such as signal or image processing in real-time.
- Interoperable:
  - Compiles all OMG IDL (although dynamic aspects of CORBA – IFR, DII, DSI, recursive Valuetypes, dynamic Any – do not execute).
  - Integrates with applications running full CORBA, CORBA/i, CORBA/e Compact Profile, and CORBA/e Micro Profile.
  - Supports native IIOP (all versions through the current GIOP 1.4 and IIOP 1.4).
- Deterministic:
  - Supports Real-time CORBA with static scheduling.
  - Propagates Real-time CORBA priorities over the wire.
  - Disallows dynamic aspects of CORBA – IFR, DII, DSI, dynamic Any, recursive Valuetypes.
- Server-side: POA supporting transient or persistent objects; retained servants (disallows implicit activation); prioritized multithreading under ORB control.
- Does not support CORBA components.
- Complete: includes Naming, Events, and Lightweight Logging Services.

CORBA/e Micro Profile:

- Truly Micro: Fits on a mobile or similar device with a low-power microprocessor, or high-end DSP.
- Interoperable:
  - Compiles all OMG IDL (Dynamic aspects of CORBA – IFR, DII, DSI, Any, Valuetypes, transient servants – do not execute).
  - Integrates with applications running full CORBA, CORBA/i, CORBA/e Compact Profile, and CORBA/e Micro Profile.
  - Supports native IIOP (all versions through the current GIOP 1.4 and IIOP 1.4).
- Deterministic:
  - Supports only statically defined Interfaces, Interactions, and Scheduling.
  - Supports Real-time CORBA Mutex interfaces.
- Server-side: For compactness and deterministic behavior, supports exactly one POA; allows only transient, retained servants with unique, system-assigned IDs and multi-threading under ORB control.

applications." (Gautam Thaker, Distributed Processing Laboratory, Lockheed Martin's Advanced Technology Laboratories)

The performance of an application using CORBA for a request is determined by several criteria. The amount of data to be transferred during the request is determined by the parameters of the request, which is defined by the needs of the application. Fewer, shorter parameters can be transferred in less time than those that are more in number or greater in size.

The execution environment performance makes a large difference. Some operating systems are faster than others, e.g., during a context switch. Processors vary widely in processing power as generally measured by clock speed. Finally, as we will see, the transport is very much a factor in both latency and throughput.

There is almost no overhead to the use of CORBA-based components if the components are collocated in the same process. The overheads disappear because the ORB "steps out of the way" while the portability and re-use benefits are retained. Even when components are distributed in different processes or on different processors, the overhead is negligible. In fact, with a well-crafted ORB, the overhead can be less than using a communication mechanism other than CORBA.

6. REAL-TIME PREDICTABILITY

Real-time systems are concerned with both the timeliness and the correctness of responses. Features are needed to propagate priorities across the network and deal with networked resources consistently. Network priorities must be managed in real-time, respecting scheduling and deadlines.

Predictability is critical to ensure the system will meet the most stringent demands and meet deadlines. The primary goal of real-time CORBA is end-to-end predictability. Real-time systems achieve this through priority-base scheduling. Because the remote system doesn't know about the originating priority, the system is indeterminate. Real-time Object Request Brokers (ORBs) solve this problem by propagating the priority to the server. The result is distributed priority inheritance and the ability to correct schedule-distributed processes.

The heterogeneous nature of CORBA is important in priority propagation. Most real-time operating systems have different ranges of priorities. For example, in one Real-Time Operating System (RTOS), the priority range is 0 to 255, where 255 is the highest precedence. Another RTOS may have a range from 0 to 127. In some cases, 0 may be the highest. If different operating systems are on different network nodes in an application, your code has to know which operating system you're using. The Real-time CORBA standard defines a universal priority range – just one more way CORBA provides a portable and transparent approach to building distributed systems.

Priority propagation ensures that the RTOS can schedule correctly end-to-end. However, the network can still be a source of priority inversions. A high-priority message may have to await the completion of a large low-priority message. That represents a message-based priority inversion. To avoid this bottleneck, Real-time CORBA supports Priority-Banded Connections. The ORB sends messages over user-specified bands based on the priorities and QoS involved. Developers can trade off network and OS resource usage against priority inversions.

7. SUMMARY

Once thought of as "big, fat and slow," the latest implementations of real-time and embedded CORBA are very small, very fast and extremely high-performance.

CORBA, as a standard, has continued to evolve over the last decade, and it has been accepted by many industries as the preferred middleware solution as evidenced by the large industry involvement within the OMG. Some of the most exciting standards development has come from the Real-Time, Embedded and Specialized Systems (RTESS) Task Force in the form of CORBA enhancements designed for the real-time and embedded developer.

A team of diverse companies, including representatives from telecommunications, aerospace, and CORBA vendors, jointly authored the CORBA/*e* specification. Objective Interface was a key contributor in this initiative, taking a leading role in driving the specification to completion.

CORBA/*e* offers an architectural solution to keep up with the rapid pace of technological change – in processors, models, and particularly communications bus types. CORBA/*e* lets developers protect their investment in development work despite rapidly accelerating change.

8. REFERENCES

[1] Object Management Group, *CORBA for embedded Specification*, OMG document ptc/06-07-02 http://www.omg.org/cgi-bin/docs?ptc/06-07-02, July 2006.