QoS for Distributed Object Computing Middleware – Fact or Fiction?

Chair: Douglas C. Schmidt, Washington University, St. Louis
Panelists:
Maximilian Ott, NEC
Guru Parulkar, Washington University, St. Louis
Rolf Stadler, Columbia University
Andreas Vogel, Visigenic, Inc.

1 INTRODUCTION

A growing class of distributed applications require end-to-end quality of service (QoS) guarantees (such as bandwidth, latency, jitter, and level of reliability). These applications include telecommunication systems (*e.g.,* call processing and switching), avionics control systems (*e.g.,* operational flight programs for fighter aircraft), multimedia (*e.g.,* video-on-demand and teleconferencing), and simulations (*e.g.,* battle readiness planning). In addition to requiring QoS guarantees, these applications can benefit by being built with flexible and reusable components, which can reduce development effort and increase software quality.

Requirements for flexible and reusable components motivate the use of Distributed Object Computing (DOC) middleware and Object Request Brokers (ORBs). Example DOC ORBs include OMG's Common Object Request Broker Architecture (CORBA), Microsoft's Distributed COM (DCOM), and JavaSoft's Remote Method Invocation (RMI). Following in the tradition of Remote Procedure Call (RPC) toolkits like Sun RPC and OSF DCE, DOC ORBs are well-suited for conventional request/response-style applications running on low-speed networks.

However, the QoS specification and enforcement features of current DOC middleware and ORBs, as well as their performance levels, are not yet suitable for applications with hard real-time requirements (e.g., avionics mission computers) and stringent statistical real-time requirements (e.g., teleconferencing). Conventional DOC ORB specifications and implementations are characterized by the following deficiencies:

- **Lack of QoS specification and enforcement** – Conventional DOC

ORBs do not define APIs that allow applications to specify their end-to-end QoS requirements. Likewise, existing DOC ORB implementations do not provide support for end-to-end QoS enforcement between applications across a network. For instance, CORBA provides no standard way for clients to indicate the relative priorities of their requests to an ORB. Likewise, there are no means for DCOM or RMI clients to inform an ORB how frequently to execute operations that must run periodically.

- **Lack of real-time features** – Conventional DOC ORBs do not provide key features that are necessary to support real-time programming. For instance, although the CORBA inter-operability protocol (GIOP) supports asynchronous messaging, there is no standard programming language mapping for exchanging ORB requests asynchronously. Likewise, the DCOM and RMI specifications do not require an ORB to notify clients when transport layer flow control occurs. Therefore, it is hard to write portable and efficient real-time applications that are guaranteed not to block indefinitely when ORB endsystem and network resources are temporarily unavailable.
- **Lack of performance optimizations** – Conventional ORBs incur significant throughput and latency overhead. These overheads stem from excessive data copying, non-optimized presentation layer conversions, internal message buffering strategies that produce non-uniform behavior for different message sizes, inefficient demultiplexing algorithms, long chains of intra-ORB virtual method calls, and lack of integration with underlying real-time OS and network QoS mechanisms.

Although some operating systems, networks, and protocols now support real-time scheduling, they do not provide integrated end-to-end solutions. In particular, QoS research at the IPC and OS layers has not necessarily addressed key requirements and usage characteristics of DOC middlware such as CORBA, DCOM, or RMI. For instance, research on QoS for communication systems has focused largely on policies for allocating network bandwidth on a per-connection basis. Likewise, research on real-time operating systems has focused largely on avoiding priority inversions and non-determinism in synchronization and scheduling mechanisms for multi-threaded applications. In contrast, the programming model for developers of DOC applications focuses largely on invoking remote operations on distributed objects. Determining how to map the results from QoS work at the IPC and OS layers to DOC middleware is an important open research topic.

Meeting the QoS needs of next-generation distributed applications requires much more than defining IDL interfaces or building real-time scheduling into ORBs. It requires a vertically integrated architecture that can deliver end-to-end QoS guarantees at multiple levels of an entire distributed system. This panel will describe the architectural features and optimizations that are necessary to develop Distributed Object Computing middleware that can deliver end-to-end QoS guarantees to applications. The topics presented by the panelists will include:

- The enchancements required to existing DOC specifications (such as OMG CORBA) that will enable applications to define their Quality of Service (QoS) requirements to ORB endsystems.
- Key architectural patterns required to build real-time ORB endsystems that can enforce deterministic and statistical end-to-end QoS guarantees to applications.
- Strategies for integrating I/O subsystem architectures and optimizations with DOC middleware to provide high bandwidth and low latency guarantees to distributed applications.
- Overview of forthcoming CORBA specifications (such as the Notification service, Messaging service, Streaming service, and passing objects by value) that address QoS requirements.
- Techniques for associating an IDL interface with multiple implementations that can provide different QoS characteristics.

We look forward to your participation in the panel.