MicroQoSCORBA:
A Reflective, QoS-Enabled, Configurable MicroCORBA with CASE Support*

A. David McKinnon, David E. Bakken, John C. Shovic

SANDS Laboratory
School of Electrical Engineering & Computer Science
Washington State University

**Updated slides at http://www.eecs.wsu.edu/~bakken/omg-mqc.pdf*

Existing CORBA Specifications

- MinimumCORBA
  - Removes support for dynamic interfaces, etc
  - Reduces the memory footprint of an ORB
- Real-time CORBA
  - Provides tools to better predict time delays
  - Enables hard real-time CORBA applications

MicroQoSCORBA Objectives

- Go beyond Minimum & Real-Time CORBA
- Be Small
  - Distributed sensor networks, home appliances
- Have QoS
  - Run time performance, security, fault tolerance
- Maintain interoperability

Architecture Overview

- Lifecycle choices
  - Exploit limiting choices from the initial idea until the end of execution
- Quality of Service choices
  - Decide what to support
- Development tools

Lifecycle Choices

- Initial design
- IDL compilation
- Code compilation/linkage
- System initialization/startup
- Run time execution

Initial Design Overview

- Embedded systems hardware
- Role definitions
- Interaction styles
- Communications support
- IDL subsetting

Hardware Choices

- Hardware characteristics
  - Homogeneity, Capabilities, Communications
- Motorola 6805
  - 4 Kbtyes ROM, 64 Bytes RAM
- Dallas Semiconductor Java iButton (rel. 1.1)
  - 64 Kbytes ROM, 6 Kbytes RAM
- Palm m100 Handheld
  - 2 Mbytes RAM

Role Choices

- Client
  - Accesses server resources
  - Thin vs. fat client
- Server
  - Fulfills client requests
- Peer
  - Sends and receives requests

Interaction Style Choices

- Send/receive
  - Synchronous messaging
- Push/Pull
  - Need lists of nodes to push/pull data from
- One-way messages
  - Asynchronous messaging support
- Exceptions
  - Occur at unpredictable times
- Event & Notification Services

Communications Support

- Hardware
  - Serial/UART
  - Ethernet
  - Wireless
- Protocols
  - Internet Standards (e.g., TCP/IP, UDP)
  - Propriety

IDL Subsetting

- Data types constraints
  - 8, 16, 32, 64-bit integers
  - Floating point values
  - Structures
- Functionality constraints
  - Fixed format messages
  - Maximum message length
- Exceptions

Interface Compilation Overview

- IDL Subsets
- Implementation specific code generation
- Representation Optimizations

Interface Compilation–IDL Subsets

- Verify conformance with the IDL subset
  - Supported data types
  - Supported message formats/styles
- Generate meta-data for the IDL compiler

Interface Compl.–Code Generation

- Application specific
  - Leverage IDL subset meta-data
- Client stub/Server skeleton Generation
  - Space/Time optimizations of the send/receive call stack
  - Fixed format messages
  - Insert QoS support, as needed

Interface Compl.–Representations

- Optimize Static String Representations
  - Method names
  - Exception names
  - String constants
- Object References
- Exceptions

Application Compilation/Linkage

- Space/Time optimizations
  - Use existing compiler techniques
- Library usage
  - Static
  - Dynamic

System Initialization/Startup

- Device initialization
  - Use reflection to load appropriate modules
  - Initialize QoS subsystems
- Network initialization
  - Locate other key nodes on the network

Run Time Execution

- Fixed functionality
- Dynamic functionality
  - Increased Resource Usage

Quality of Service

- Security
  - Multiple strengths/Algorithms
- Fault Tolerance
  - Quantity and types of faults tolerated
- Real-time Behavior
  - Scheduling Algorithms, Network performance
- Resource Issues
  - Power awareness

Configurability

- Each embedded system is slightly different
- Not everyone will need the kitchen sink
- Let the developer pick and choose
- Configure QoS properties, too
  - Multiple implementations/strengths for one property to choose from
  - Choose at startup with reflection/introspection

CASE Tools

- Not all developers are created equal
- Make it easy for the casual programmer
  - Domain expert, but QoS novice
  - Lifecycle support personnel
  - Temporary/contract employees
- Let the tools choose compatible components based upon
  - QoS requirements
  - Resource configuration

MicroQoSCORBA Challenges

- What ORB/service constraints/subsets make sense (and what are their resource costs)?
- What QoS implementations/strengths make sense for each QoS property?
- What combinations of QoS property implementations make sense? (e.g., ‘X’ real-time & ‘Y’ security)
- How can reflection be used to improve the QoS performance of an ORB at runtime?

Conclusions

- MicroQoSCORBA will
  - Enhances software reliability & security
  - Allows standards-based connectivity to the deeply embedded systems market
  - Provide QoS in a variety of subsystems
- Further work is needed
  - QoS components
  - CASE configuration tools

Note: Updated slides at http://www.eecs.wsu.edu/~bakken/omg-mqc.pdf