Enabling Flexible QoS Support in the Object Request Broker COOL

Tom Kristensen and Thomas Plagemann
University of Oslo, Center for Technology at Kjeller (UniK)
Norway
{tomkri, plageman}@unik.no

International Workshop on Distributed Real-Time Systems (IWDRS 2000)
at 20th International Conference on Distributed Computing Systems (ICDCS'00)

Outline

• Role of middleware in general
• How CORBA fails to support QoS
• MULTE-ORB
  – approach
  – QoS specification and negotiation
  – flexible protocol framework
  – status and further work
• Conclusion
The Problems to Solve by the Middleware

Heterogeneity of user requirements:
→ user preferences, QoS, cost
→ user needs, blind students don’t need video but good audio

Heterogeneity of equipment and network:
→ network access, LAN or modem?
→ Intel Pentium III or 486 based PC?
→ which video formats are supported?

Lack of Existing CORBA Implementations

- Known problems
  - streams
  - QoS & realtime: CPU scheduling, FIFO queues, TCP/IP
  - performance optimizations: de-multiplexing, de-marshalling
CORBA Extensions

• Control and Management of A/V Streams
  – QoS specification
  – ORB for management
  – data on separate connections

• Real-Time CORBA
  – end to end priority propagation (fixed priority)
  – thread pools (different models)
  – explicit binding
  – protocol policies and configuration
  – scheduling service

MULTE-ORB: General Principles

• General goal: flexible protocol support for multimedia applications
  – decomposition of complex protocols into fine-granular micro-protocols
  – Quality-of-Service (QoS)
  – functional behaviour

• Principles:
  – run-time configuration & re-configuration
  – selection of optimal protocol configuration
  – implementing open and explicit bindings

• Prototype integrates Da CaPo and COOL
MULTE-ORB Architecture

- Based on commercial CORBA implementation COOL
  - Enables objects to specify QoS and performs QoS negotiation
  - Flexible protocol framework

Lack in CORBA Revisited

- MULTE-ORB v1
  - flexible protocol
  - QoS specification and negotiation

- MULTE-ORB v2
  - efficient adapter and stubs
  - buffer management
QoS Specification at Object and Message Layer - I

- QoS per object:
  - too coarse solution

- QoS per binding:
  - changes in QoS requirements require renegotiation of binding

- QoS per method invocation:
  - each method invocation results in a request and reply message
  - messages are transported over one transport connection

- QoS per parameter:
  - most flexible and most complex approach
  - multiplexing different parameters with different QoS over one transport connection? or using multiple connections?

- QoS per binding and per method invocation

QoS Specification at Object and Message Layer - II

- QoS per binding and per method invocation
  - \texttt{setQoSParameter(struct QoSparameter **qp)} to inform stub about QoS requirements

```
setQoSParameter(Q_1)
object.method()
setQoSParameter(Q_2)
object.method()
```

binding established
QoS Negotiation at the Message Layer

Extensions in GIOP

- Differentiating GIOP and QIOP:
  - use of version field in GIOP header:
    - standard GIOP: major 1, minor 0
    - QIOP: major 9, minor 1
- Integrating QoS parameters in QIOP messages
  - extended header of request message QoS parameter specification
  - IDL compiler has been extended to marshal method invocation and QoS parameter into the extended request message
The operation will be aborted if the requested QoS cannot be supported and an exception will be returned to the client.

### Related Research

- **Lancaster University**
  - Ensemble + COOL
    - focus on group communication
    - coarse grained modules
    - no dynamic QoS specification and negotiation

- **Electra ORB**
  - Ensemble based
    - focus on security and reliability
    - operation invocation
Status

- Implemented
  - QIOP
  - Da CaPo integration v1
- Ongoing
  - performance measurements and evaluation
  - Da CaPo integration v2

Further Work

- Multimedia stream adapter
- Open signalling protocol
  - build signalling protocol from modules
  - connection management
  - binding protocol
  - QoS and configuration negotiation
  - resource reservation
Summary and Conclusion

• Presented
  – today’s middleware fails
  – solutions to the problem
  – our approach

• Basis in CORBA implementation
  – conforms to standard

• Able to support QoS
  – specify dynamic QoS
  – flexible protocol in ORB core
  – minor changes to extend the ORB
  – backwards compatible