


# Managing Cross-Cutting QoS Issues in MULTE Middleware

T. Plagemann, F. Eliassen, B. Hafskjold, T. Kristensen,
R.H. Macdonald and H.O. Rafaelsen

University of Oslo, Norwegian Defence Research Establishment, University of Tromsø

Norway

Workshop on Quality of Service in Distributed Object Systems
at 14th European Conference on on Object-Oriented Programming (ECOOP 2000)



## Outline

- MULTE project approach
- Goals for QoS management
- Abstract QoS architecture
  - layered model
- Current prototype - MULTE-ORB
  - QoS specification and negotiation
  - flexible protocol framework
  - flexible management and signalling
- Conclusion

## MULTE-ORB: General Principles

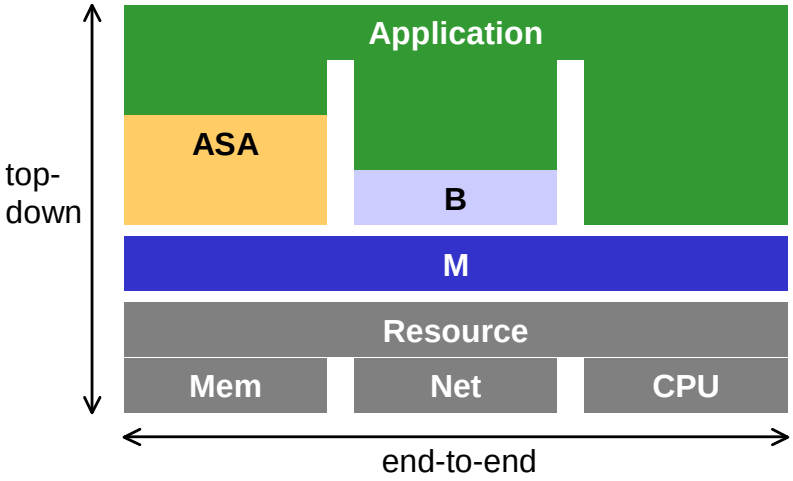
- General goal: flexible protocol support for multimedia applications
  - decomposition of complex protocols into fine-granular micro-protocols
  - Quality-of-Service (QoS)
  - functional behaviour
- Principles:
  - run-time configuration and re-configuration
  - selection of optimal protocol configuration
  - implementing open and explicit bindings
- Application domains
  - distance education (OMODIS LoD)
  - command and control systems
- Prototype integrates Da CaPo in COOL-ORB



## Goals for QoS Management

- Dynamic QoS support
  - adaptable at runtime
- Transparency vs. fine-grained control
- Policy control
  - QoS mapping, control, adaption, ...
- Automatic support for compatibility control
- Support for seamless system evolution

# Abstract QoS Architecture

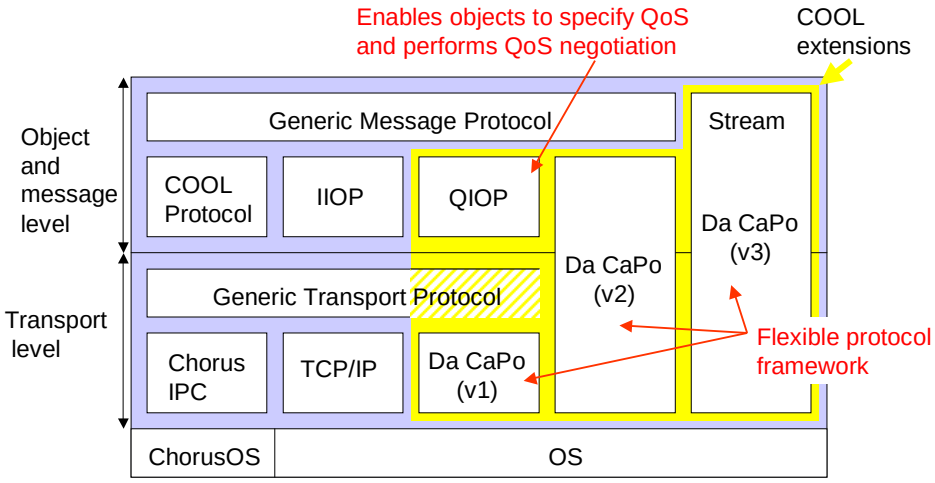



# ASA, B and M

- QoS mediator language (M)
  - intermediate syntax for QoS
- Model of Bindings and Stream (B)
  - language for describing streams/bindings
- Application Specific Adapter (ASA)
  - video conference ➔H.323 ASA

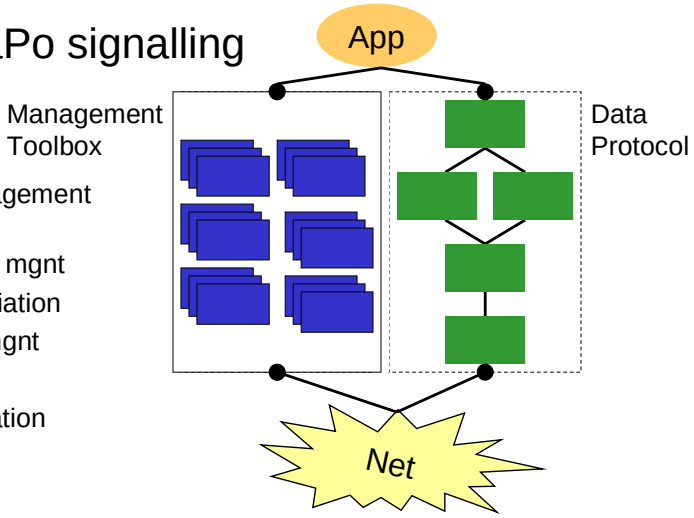
# MULTE-ORB Prototype - I

- Based on commercial CORBA implementation COOL




# MULTE-ORB Prototype - II

- Da CaPo signalling



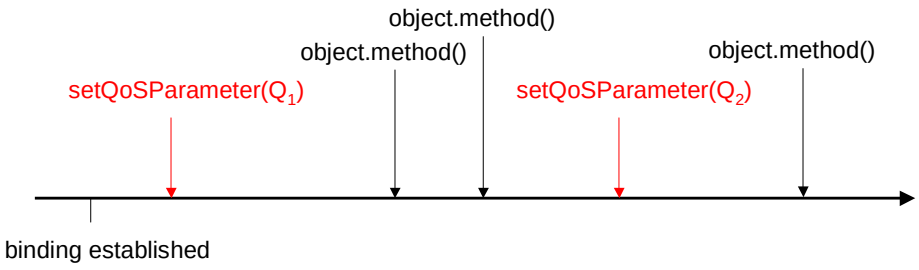

- Different management modules
  - connection mgnt
  - QoS negotiation
  - resource mgnt
  - monitors
  - reconfiguration
- Policies

# 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
  - `setQoSParameter(struct QoSparameter **qp)` to inform stub about QoS requirements



# Conclusion

- QoS architecture
  - 3 level access (M, B, ASA)
  - automatic conformance testing
  - dynamic QoS, runtime (re-)configuration
  - flexible data and signalling protocols
- Basis in CORBA implementation
  - conforms to standard, backwards compatible
- Integrated prototype with OMODIS LoD
  - map to B
  - ASA for given application (optimization)
  - generate protocols after database query

# Further Information

**http://www.unik.no/~multe**

**tomkri@unik.no**

➔Questions?