Patterns and Performance of Real-time Object Request Brokers

Douglas C. Schmidt
Associate Professor
University of California, Irvine
www.ece.uci.edu/~schmidt/
(949) 824-1901

Sponsors
NSF, DARPA, ATD, BBN, Boeing, Cisco, Comverse, GDIS, Experian, Global MT,
Hughes, Kodak, Krones, Lockheed, Lucent, Microsoft, Mitre, Motorola, Nokia, Nortel,
OCI, Oresis, OTI, QNX, Raytheon, SAIC, Siemens SCR, Siemens MED, Siemens ZT,
Sprint, Telcordia, USENIX

High-performance, Real-time ORBs
Motivation: the QoS-enabled Software Crisis

• Symptoms
  – Communication hardware gets smaller, faster, cheaper
  – Communication software gets larger, slower, more expensive

• Culprits
  – Inherent and accidental complexity

• Solution Approach
  – Standards-based COTS Hardware & Software

www.arl.wustl.edu/arl/

Problem: the COTS Hardware & Software Crisis

• Context
  – Adopting COTS hardware & software is increasingly essential for real-time mission-critical systems

• Problems
  – Inherent and accidental complexity
  – Integration woes

• Solution Approach
  – Standards-based adaptive COTS middleware
High-performance, Real-time ORBs

Context: Levels of Abstraction in Internetworking and Middleware

- RTP
- TELNET
- DNS
- FTP
- HTTP
- UDP
- TCP
- ETHERNET
- ATM
- FDDI
- FIBRE CHANNEL

INTERNETWORKING ARCH

MIDDLEWARE ARCH

UC Irvine

3

Problem: Lack of QoS-enabled Middleware

- Many applications require QoS guarantees
  - e.g., avionics, telecom, WWW, medical, high-energy physics
- Building these applications manually is hard and inefficient
- Existing middleware doesn’t support QoS effectively
  - e.g., CORBA, DCOM, DCE, Java
- Solutions must be integrated horizontally & vertically

UC Irvine

4

Candidate Solution: CORBA

Goals of CORBA

- Simplify distribution by automating
  - Object location & activation
  - Parameter marshaling
  - Demultiplexing
  - Error handling
- Provide foundation for higher-level services

UC Irvine

5
**Overview of the Real-time CORBA Specification**

**Features**
1. Portable priorities
2. End-to-end priority propagation
3. Protocol properties
4. Thread pools
5. Explicit binding
6. Standard synchronizers

**Portable Priorities**

- Designed to support heterogeneous real-time platforms
- CORBA priorities range from 0 → 32767
- Users can map CORBA priorities to native OS priorities
- No silver bullet, but rather an "enabling technique"
**High-performance, Real-time ORBs**

**Thread Pools**

- Pre-allocate threads and thread attributes
  - Stacksize
  - Static threads
  - Maximum threads
  - Default priority
- Applicable at both the ORB and POA level

**Features**

- Configurable Protocol Properties

  ```
  interface ProtocolProperties {
    typedef struct {
      IOP::ProfileId protocol_type;
      ProtocolProperties orb_protocol_properties;
      ProtocolProperties transport_protocol_properties;
    } Protocol;
  }
  ```

- Select and configure communication protocols
  - TCP socket options
  - ORB protocol and transport protocol configuration
- Ordering in ProtocolList indicates preferences

**End-to-End Priority Propagation**

- Client priorities can propagate end-to-end
- Servers can also declare priority

**UC Irvine**
High-performance, Real-time ORBs

Our Approach: The ACE ORB (TAO)

Standard Synchronizers

- Mutex
  - lock()
  - unlock()
  - try_lock()

Features

- A portable Mutex API
  - e.g., lock, unlock, try_lock
- Necessary to ensure consistency between ORB and application synchronizers
- Locality constrained

Explicit Binding

_validate_connection (out CORBA::PolicyList inconsistent_policies);

Features

- Enables pre-establishment of connections
  - Priority-banded connections
  - Private connections
  - Protocol policies
**ACE Overview**

- A concurrent OO networking framework
- Available in C++ and Java
- Ported to POSIX, Win32, and RTOSS

**Related work**

- x-Kernel
- SysV STREAMS

---

**ACE and TAO Statistics**

- Over 50 person-years of effort
  - ACE > 200,000 LOC
  - TAO > 200,000 LOC
  - TAO IDL compiler > 130,000 LOC
  - TAO CORBA Object Services > 150,000 LOC
- Ported to UNIX, Win32, MVS, and RTOSS platforms
- Large user community
  - ~schmidt/ACE-users.html
- Currently used by dozens of companies
  - Bellcore, BBN, Boeing, Ericsson, Hughes, Kodak, Lockheed, Lucent, Motorola, Nokia, Nortel, Raytheon, SAIC, Siemens, etc.
- Supported commercially
  - ACE → www.riverace.com
  - TAO → www.theaceorb.com

---

**Applying TAO to Avionics Mission Computing**

**Domain Challenges**

- Deterministic & statistical real-time deadlines
- Periodic & aperiodic processing
- COTS and open systems
- Reusable components
- Support platform upgrades

- www.cs.wustl.edu/~schmidt/TAO-boeing.html

---

**www.cs.wustl.edu/~schmidt/ACE.html**
**Applying TAO to Distributed Interactive Simulations**

### Domain Challenges
- High scalability and group communication
- High throughput and low latency
- "Interactive" real-time
- Multi-platform

---

**Optimization Challenges for QoS-enabled ORBs**

### Key Challenges
- Alleviate priority inversion and non-determinism
- Reduce demultiplexing latency/jitter
- Ensure protocol flexibility
- Specify QoS requirements
- Schedule operations
- Eliminate (de)marshaling overhead
- Minimize footprint
**Solution 2: ORB Optimization Principle Patterns**

**Definition**

- **Optimization principle patterns** document rules for avoiding common design and implementation problems that can degrade the efficiency, scalability, and predictability of complex systems.

**Optimization Principle Patterns Used in TAO**

<table>
<thead>
<tr>
<th>#</th>
<th>Optimization Principle Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Optimize for the common case</td>
</tr>
<tr>
<td>2</td>
<td>Remove gratuitous waste</td>
</tr>
<tr>
<td>3</td>
<td>Replace inefficient general-purpose functions with efficient special-purpose ones</td>
</tr>
<tr>
<td>4</td>
<td>Shift computation in time, e.g., precompute</td>
</tr>
<tr>
<td>5</td>
<td>Store redundant state to speed-up expensive operations</td>
</tr>
<tr>
<td>6</td>
<td>Pass hints between layers and components</td>
</tr>
<tr>
<td>7</td>
<td>Don’t be tied to reference implementations/models</td>
</tr>
<tr>
<td>8</td>
<td>Use efficient/predictable data structures</td>
</tr>
</tbody>
</table>
Lessons Learned Developing QoS-enabled ORBs

- Avoid dynamic connection management
- Minimize dynamic memory management and data copying
- Avoid multiplexing connections for different priority threads
- Avoid complex concurrency models
- Integrate ORB with OS and I/O subsystem and avoid reimplementing OS mechanisms
- Guide ORB design by empirical benchmarks and patterns

Concluding Remarks

- Researchers and developers of distributed, real-time applications confront many common challenges
  - e.g., service initialization and distribution, error handling, flow control, scheduling, event demultiplexing, concurrency control, persistence, fault tolerance
- Successful researchers and developers apply patterns, frameworks, and components to resolve these challenges
- Careful application of patterns can yield efficient, predictable, scalable, and flexible middleware
  - i.e., middleware performance is largely an “implementation detail”
- Next-generation ORBs will be highly QoS-enabled, though many research challenges remain

Web URLs for Additional Information

- These slides: ~/schmidt/TAO4.ps.gz
- More information on CORBA: ~/schmidt/corba.html
- More info on ACE: ~/schmidt/ACE.html
- More info on TAO: ~/schmidt/TAO.html
- TAO Event Channel: ~/schmidt/JSAC-98.ps.gz
- TAO static scheduling: ~/schmidt/TAO.ps.gz
- TAO dynamic scheduling: ~/schmidt/dynamic.ps.gz
- ORB Endsystem Architecture: ~/schmidt/RIO.ps.gz
- Pluggable protocols: ~/schmidt/pluggable_protocols.ps.gz
Web URLs for Additional Information (cont’d)

- Network monitoring, visualization, & control: ~/schmidt/NMVC.html
- Performance Measurements:
  - Demuxing latency: ~/schmidt/COOTS-99.ps.gz
  - SII throughput: ~/schmidt/SIGCOMM-96.ps.gz
  - DII throughput: ~/schmidt/ICOCOM-96.ps.gz
  - ORB latency & scalability: ~/schmidt/IEEE-TC-97.ps.gz
  - IIOP optimizations: ~/schmidt/JSAC-99.ps.gz
  - Concurrency and connection models: ~/schmidt/RT-perf.ps.gz
  - RTOS/ORB benchmarks:
    - ~/schmidt/RT-OS.ps.gz
    - ~/schmidt/words-99.ps.gz