Real-Time CORBA

Chenyang Lu
CSE 520S
CORBA
Common Object Request Broker Architecture

- CORBA specifications
  - OMG is the standards body
  - Over 800 companies
  - CORBA defines interfaces, not implementations

- Object Request Brokers (ORB) allow clients to invoke operations on distributed objects transparently of
  - object location
  - programming language
  - operating system
  - communication protocols and interconnect
  - hardware
CORBA Architecture

- CORBA: Common Object Request Broker Architecture
- Remote Procedure Call (RPC): Client invokes operations on objects.
- An Object =
  - An interface specified by an Interface Definition Language (IDL)
  - Servant(s) that implements the IDL interface
The ACE ORB (TAO)

- Open-source Real-Time CORBA
- >> 1M SLOC
- 100+ person years of effort
- Pioneered R&D on DRE middleware design & optimization

Source: D.C. Schmidt, [http://www.dre.vanderbilt.edu/~schmidt/TAO.html](http://www.dre.vanderbilt.edu/~schmidt/TAO.html)
Application Example: Avionics

- Apply COTS & open systems to mission-critical real-time avionics
  - Deterministic & statistical deadlines
    - ~20 Hz
  - Low latency & jitter
    - ~250 us
  - Periodic & aperiodic processing
  - Complex dependencies
  - Continuous platform upgrades

- Test flown at China Lake NAWS by Boeing OSAT II '98
  - www.cs.wustl.edu/~schmidt/TAO-boeing.html
- Also used on SOFIA project by Raytheon
  - sofia.arc.nasa.gov
- First use of Real-time CORBA in avionics
- Drove Real-time CORBA standardization
Application Example: Hot Rolling Mills

- **Goals:** Control the processing of molten steel moving through a hot rolling mill in real-time

- **System Characteristics**
  - Hard real-time process automation requirements
    - *i.e.*, 250 ms real-time cycles
  - System acquires values representing plant’s current state, tracks material flow, calculates new settings for the rolls & devices, & submits new settings back to plant

- **Key Software Solution Characteristics**
  - Affordable, flexible, & COTS
  - Product-line architecture
  - Design guided by patterns & frameworks
  - Windows NT/2000
  - Real-time CORBA

- www.siroll.de
Application Example: Image Processing

**Goals:** Examine glass bottles for defects in real-time

**System Characteristics**
- Process 20 bottles/sec
  - ~50 ms per bottle
- Networked configuration
- ~10 cameras

**Key Software Solution Characteristics**
- Affordable, flexible, & COTS
- Embedded Linux
- Compact PCI bus + Celeron processors
- Remote booted by DHCP/TFTP
- Real-time CORBA

[Image of a machine processing bottles]

[www.krones.com]
CORBA: Common Object Request Broker Architecture

Remote Procedure Call (RPC): Client invokes operations on objects.

An Object =
- An interface specified by an Interface Definition Language (IDL)
- Servant(s) that implements the IDL interface
Limitations of CORBA

- Lacks QoS specification interfaces to applications
  - Applications cannot specify rate, deadline or importance

- Lacks QoS enforcement
  - Does not map task QoS specification to priorities of threads
  - Contains significant priority inversion

- Lacks performance optimization
  - Poor worst-case and average latency
Latencies and Priority Inversions

1) CLIENT MARSHALING
2) CLIENT PROTOCOL QUEUEING
3) NETWORK DELAY
4) SERVER PROTOCOL QUEUEING
5) THREAD DISPATCHING
6) REQUEST DISPATCHING
7) SERVER DEMARSHALING
8) METHOD EXECUTION
The ACE ORB (TAO)

• Open-source Real-Time CORBA

• >> 1M SLOC

• 100+ person years of effort

• Pioneered R&D on DRE middleware design & optimization
Objective: Simplify the development of distributed real-time & embedded (DRE) systems

Approach: Use standard technology, patterns, & frameworks

• Based on ACE wrapper facades & frameworks
• Available on Unix, Win32, MVS, QNX, VxWorks, LynxOS, VMS...
• Thousands of users around the world

Commercially supported
http://www.dre.vanderbilt.edu/~schmidt/commercial-support.html
I/O Subsystem: Priority Inversions

- Messages are processed in FIFO order regardless of priorities

- Kernel has lower priority than real-time priorities
  - Processing of a high priority message in the kernel can be blocked by a lower priority task at the application level
I/O Subsystem: Solutions

- Early demultiplexing
- Prioritized kernel processing
- Map task priority to kernel thread
- Note: This needs kernel support
ORB Core: Priority Inversions

- Communication of different tasks shares a same socket connection.
- Incoming requests are demultiplexed to threads in FIFO order.
ORB Core
Priority-based Concurrency Architecture

- Server sets model: Each priority is processed by a separate thread.
- A separate connection is maintained for each priority in the server ORB.
  - Use buffered connections to reduce run-time overhead.
- Suitable for fixed priority scheduling.
Object Adapter: Problems

- Layered demultiplexing is inefficient in terms of:
  - average latency
  - worst-case latency
Object Adapter: Solutions

- Perfect hashing
  - Generate a hash function offline
  - Computational complexity $O(1)$
- De-layered active demultiplexing
  - Clients obtain index to (servant, operation) ahead of time
Reduce Priority Inversion

Conventional ORB

TAO
Real-Time Event Service
Limitations of RPC

- Tight coupling between clients and server
- Lacks asynchronous message delivery
- Lacks support for group communication
Motivating Applications

Boeing Bold Stroke Middleware Infrastructure Platform
- Operations well defined
- Event-mediated middleware solution

### Domain
<table>
<thead>
<tr>
<th>Domain</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avionics mission computing</td>
<td>Boeing, Raytheon</td>
</tr>
<tr>
<td>Mass storage devices</td>
<td>SUTMYN, StorTek</td>
</tr>
<tr>
<td>Medical Information Systems</td>
<td>Siemens, GE</td>
</tr>
<tr>
<td>Satellite Control</td>
<td>LMCO COMSAT</td>
</tr>
<tr>
<td>Telecommunications</td>
<td>Motorola, Lucent, Nortel, Cisco, Siemens</td>
</tr>
<tr>
<td>Missile &amp; Radar Systems</td>
<td>LMCO Sanders, Raytheon</td>
</tr>
<tr>
<td>Steel Manufacturing</td>
<td>Siemens ATD</td>
</tr>
<tr>
<td>Beverage Bottling Automation</td>
<td>Krones AG</td>
</tr>
</tbody>
</table>
CORBA Event Service

- Asynchronous event delivery
- Decouples suppliers and consumers
- Support group communication
Event Delivery Models

Suppliers push to Event Channel, Event Channel pushes to Consumers

Event Channel pulls from Suppliers, Consumers pull from Event Channel

(A) The Canonical Push Model

(B) The Canonical Pull Model

(C) The Hybrid Push/Pull Model

(D) The Hybrid Pull/Push Model

Suppliers push to Event Channel, Consumers pull from Event Channel

Event Channel pulls from Suppliers, Event Channel pushes to Consumers
CORBA Event Service Limitations

- Broadcasts any supplied event to all consumers
- No filtering or correlation
- No real-time event dispatching/scheduling
- No periodic event
TAO Real-Time Event Service

- Suppliers specify **types** of events generated.
- Consumers subscribe to interested event types, to events from specific suppliers, or both.
Event Channel Architecture

- Push-push model for real-time delivery

- Modularity: features are implemented in pluggable modules
Suspend and Resume

- Light-weight operations to suspend/resume event delivery

- Useful for frequent changes in consumer sets
  - Mode change
Periodic Events

- Consumers can register for timeout events provided by Event Service

- Timeout events dispatched according to priority of consumer
Event Correlation

- Provides event correlation
  - Conjunctive ("AND")
  - Disjunctive ("OR")
Scheduling Service

- Separates dispatching mechanism from scheduling policy
- Dispatcher consults run-time scheduler for priorities
- Flexibility for different scheduling policies (e.g., RMS, EDF, MUF)
Real-Time Event Dispatching

- High Priority
  - Kokyu Dispatching Queue
    - Dispatching Thread

- Low Priority
  - Kokyu Dispatching Queue
    - Dispatching Thread
Critiques

- Full compatibility with CORBA
  - Too complex and heavy-weight for embedded systems?

- Centralized event channel
  - Scalability for distributed systems ➔ federated event channel
  - Single point of failure ➔ fault tolerant event channel

- Overhead of redirection

- Support for multicore and multiprocessor platforms.
Federated Event Channel

- Gateway forwards events to remote processors.

Diagram:
- Application Processor 1
  - Event Channel (EC)
  - T₁₁
- Application Processor 2
  - Event Channel (EC)
  - T₁₂
- Application Processor 3
  - Event Channel (EC)
  - T₁₃
Slides partially based on Douglas Schmidt’s TAO tutorial and Joe Hoffert’s class presentation.