Design and Performance of a Real-Time Notification Service

Pradeep Gore and Irfan Pyarali
OOMWorks
{pradeep, irfan}@oomworks.com

Chris Gill
Washington University
cdgill@cse.wustl.edu

Douglas C. Schmidt
Vanderbilt University
d.schmidt@vanderbilt.edu

Real-time and Embedded Systems Workshop
Reston, VA USA - July 12-15, 2004
Contents

• Introduction to Real-Time Notification
• IDL extensions and programming paradigm
• Performance Analysis
Notification Service Component Structure
Real-Time Notification Service

• RT-CORBA 1.0 provides end-to-end QoS guarantees for direct client-server communication
• Standard Notification Service does not make use of the priority and scheduling capabilities defined in RT-CORBA 1.0
• RT-Notification provides end-to-end QoS guarantees for anonymous event communication
• Improved timeliness and predictability in the transmission and delivery of events to event consumers via Event Channels
• Integration with Real-Time CORBA, particularly in the areas of configuration of priorities and scheduling
Requirements of RT-Notification RFP

- Limit Complexity of Filters.
- Provide subset of filtering process.
- Subset functionality for Real-time behavior.
- Priority aware end-to-end event propagation.
- Provide means to set RT QoS parameters.
- Interface for resource management.
- Support Interaction with a scheduling service and describe schedulable entities.
RT-Notification Architecture

- Concurrency options
  - Reactive
  - ThreadPool
  - ThreadLane
- QoS support
  - Priority
  - Ordering
  - Discarding
  - Timeouts
Integration with RT-CORBA 1.0

- Support RT-CORBA Thread Pools and Lanes
- Proxy’s are activated in RT-POA
- Support Priority Models
- Extends TAO Notification Service.
- IDL extensions and new QoS properties.
IDL Extensions

• `set_qos` method sets QoS properties via name-value pairs
  • Specify thread-pools at Channel and Admin levels
  • Notify IDL Extension defines `ThreadPoolParams`, `LanesParams`

• `obtain_proxyconsumer` and `obtain_proxysupplier` modified to accept QoSProperties

```cpp
module NotifyExt {
    const string ThreadPool = "ThreadPool";

    // ThreadPoolParams : same as
    // RTCORBA::create_threadpool
    struct ThreadPoolParams {
        unsigned long stacksize;
        unsigned long static_threads;
        unsigned long dynamic_threads;
        Priority default_priority;
        boolean allow_request_buffering;
        unsigned long max_buffered_requests;
        unsigned long max_request_buffer_size;
    };
};
```
Thread Pool Support in RT-Notification

- Thread Pool can be set at 3 levels – Channel, Admin and Proxy
- Proxy Objects are activated in RT-POA at one of the 3 levels
- Levels allow sharing of thread pools; e.g., supplier admin thread pool allows multiple proxy consumers to share threads
Possible Thread Pool Configurations

Real-Time Notification Service
End-to-end priority preservation
Performance Results

- Experimental Setup
- Load vs Throughput
- Paths vs Throughput
- Max Throughput
Experimental Setup

- Each Supplier runs in a separate thread.
- Each Supplier can be configured with a priority and period.
- Each ProxyConsumer is activated in a Lane that matches the supplier priority.
- “Path” refers to event path from 1 supplier to 1 consumer.
- Results from TAO 1.3.3 on Redhat Linux 7.1, 800Mhz CPU (Emulab)
Load vs Throughput
Latency (Cos Notification)

In previous setup at Load = 25
Latency (RT-Notification)

In previous setup at Load = 25
Paths vs Throughput
**Paths vs Throughput**

- 1 High Priority Path
- Several same Low priority paths
- Each at 100Hz
- 2 Lanes at EventChannel level.
- Increasing number of low priority paths: 1, 3, 5, 10, 20
- No Threads in CosNotification.
- `$TAO_ROOT/orbsvcs/tests/Notify/performance-tests/scripts/Paths_vs_Throughput`
Real-Time Notification Service

Paths vs Throughput

Load = 30

• Throughput of High Priority Path is maintained.
Conclusion

• TAO’s RT Notification extends the Notification service.

• Integrates with RT CORBA features
  – ThreadPools and Lanes
  – Priority Models

• Performance tests demonstrate
  – Priority Preservation
  – Low Jitter
  – Low Overhead