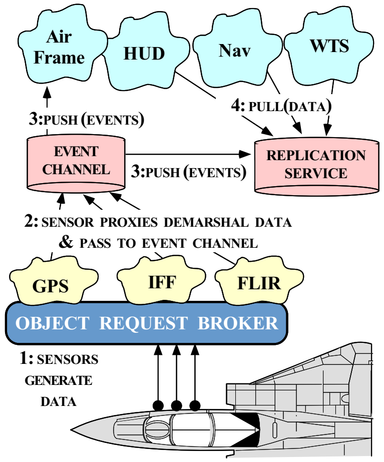
Real-time CORBA

Irfan Pyarali

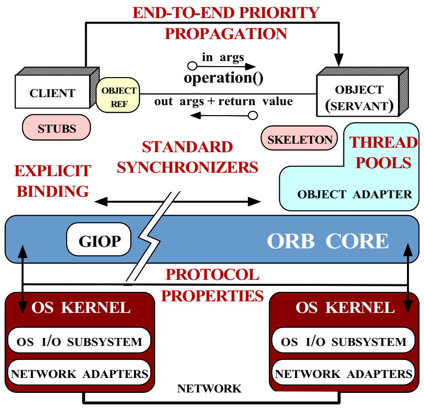
April 21, 2000

Motivation for Real-time CORBA



- Growing class of systems require QoS support:
  - bandwidth
  - jitter
  - latency
- COTS middleware infrastructure
- Standard-based open systems

Real-time CORBA



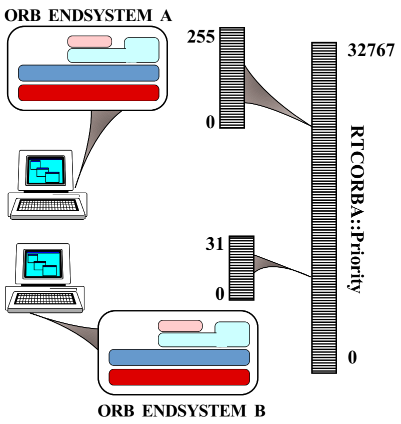
Vertical and horizontal integration and management of resources:

- **Communication infrastructure**
  - Network reservation
  - Cell pacing
- **OS scheduling**
  - Thread priorities
- **ORB behavior**
  - Message buffering and prioritization
  - Queue ordering
  - Request routing
- **Scheduling and predictability**
  - Global scheduling service
  - Predictable execution
  - Admission control

Managing Processor Resources

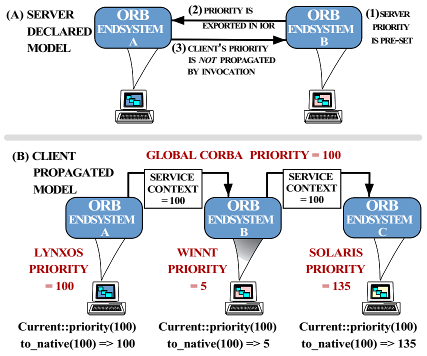
- Provides strict control over scheduling and processor execution
  - Essential for fixed-priority real-time systems
- Users specify priority at which CORBA requests will execute
- Servers can have thread pools
- Consistent thread synchronizers to avoid priority inversion

Priority Type System



- **CORBA priorities** → 0 to 32767
- **Native priorities** → platform dependent
- ORB provides mapping between CORBA and native priorities
  - Users can provide custom mapping

Priority Models



- **Server declared**
  - Server dictates priority at which invocation will execute
  - Priority assigned when object was registered at server
- **Client propagated**
  - Server executes invocation at priority requested by client
  - Priority encoded as part of client request

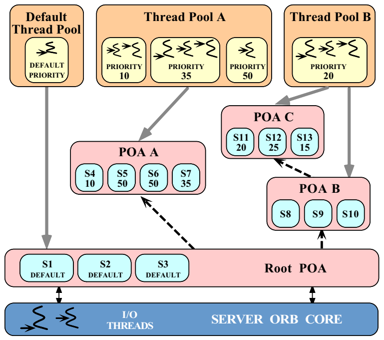
Priority Transforms

- User can provide priority transforms for invocations:
  - Inbound transforms for servers
  - Outbound transforms for clients
- Custom priority transforms are typically based on:
  - Server load
  - Operation criticality
  - Global scheduling

Multi-threading

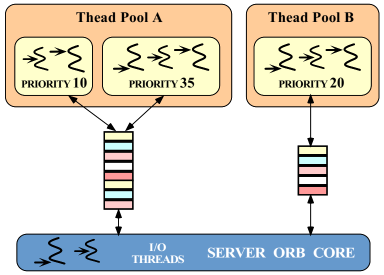
- **Goals**
  - Exploiting multiple CPUs
  - Overlap computation and I/O
  - Provide different levels of service
    - High-priority vs low-priority tasks
  - Use thread preemption to avoid unbounded priority inversion
- **Background**
  - CORBA originally did not support standard API for MT programming
    - Application used OS or ORB specific APIs
  - Reactive programming does not scale
    - Also unsuitable for long duration tasks
- New thread pool APIs address these issues

Thread Pools in Real-time CORBA



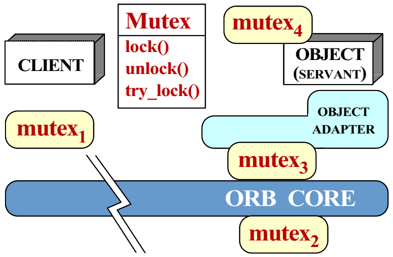
- Servers have multiple thread pools that execute incoming requests
- Each thread pool has:
  - Initial number of preallocated static threads
  - Maximum number of on-demand dynamic threads
  - Default priority of threads
    * Priority changes dynamically according to priority model or priority transform
- Pool with lanes: threads are split into different priority ranges

Buffering Requests



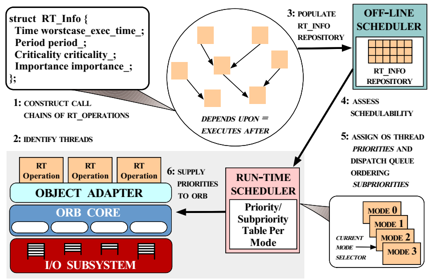
- Requests are buffered when all threads are busy
- Buffering can be specified in terms of:
  - Number of bytes
  - Number of requests
- When buffers are full:
  - A transient exception is thrown to client
  - Request is dropped by server
  - Request can be reissued later by client

Standard Synchronizers



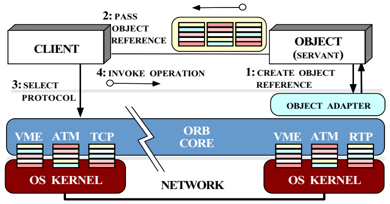
- Consistent synchronization semantics between application and ORB
  - Enforce priority inheritance
  - Priority ceiling protocol

Global Scheduling Service



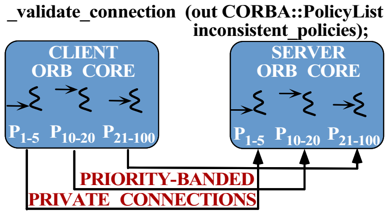
- Integrate RT dispatcher into ORB
- Application specify processing requirements
  - Worse-case execution time
  - Periodicity
  - Execution dependencies
  - Relative importance
- Support multiple request scheduling strategies
  - *e.g.,* MUF, MLF, and EDF

Protocol Selection and Configuration



- Protocol policies control protocol selection and configuration
  - Order of protocols indicates protocol preference
- Both server-side and client-side policies supported
  - Some policies control protocol selection, others control protocol configuration
  - Some policies are exported to client in object reference

Explicit Binding in Real-time CORBA



- Implicit binding creates connection on demand
  - Connection created on first request
  - Allows multiplexing
  - Leads to unpredictable jitter
- Explicit binding pre-establishes connections
  - Connections can be made private (non-multiplexed)
  - Priority banded connections allow end-to-end priority preservation

Concluding Remarks

- CORBA now supports propagation of client priority to server
- Mechanisms for avoiding or bounding priority inversion
- Mechanisms for limiting method invocation blocking
- Assumes support from underlying OS and network
- Fixed-priority scheduling addressed
  - Dynamic-priority scheduling to be addressed soon
- Backward compatible
  - Best-effort support for non-RT applications
- More information, source code, and documentation
  - `www.cs.wustl.edu/~schmidt/TAO.html`