



# Real-Time Systems and CORBA

Vic Giddings

Senior Product Engineer

Objective Interface Systems

[victor.giddings@ois.com](mailto:victor.giddings@ois.com)

<http://www.ois.com>



# Introduction to Real-Time Principles



## What is Real-Time All About?

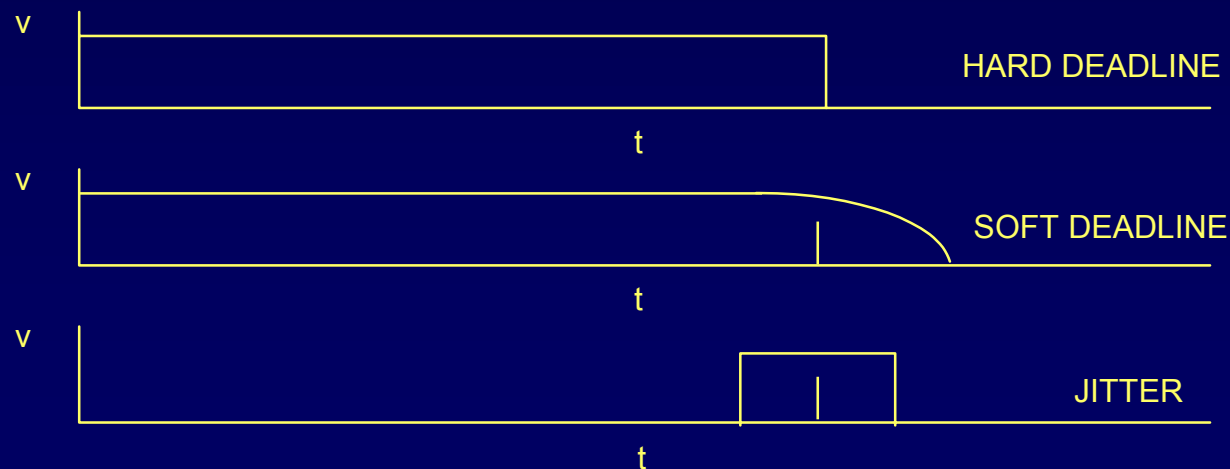
- **Real-Time is Not (Necessarily) About**
  - Speed
  - Efficiency
- **Real-Time is About Time Constraints**
  - Objects contend for system resources (e.g., CPU, LAN, I/O)
  - Rules for contention resolution follow different strategies
    - These strategies are NOT equivalent in terms of resource utilization or engineering viability
    - There is a wide range of approaches (e.g., frames, round-robin, priority)



## Definitions of a Real-Time System

- A real-time system is one in which correctness depends on meeting time constraints.
  - Must satisfy response time requirements as well as functional requirements
- A real-time system produces a value to the user which is a function of time

### Sample real-time value functions





# Hard Real-Time, Soft Real-Time

- **Hard Real-Time**
  - Resources must be managed to guarantee all hard real-time constraints are met, all the time
  - No unbounded priority inversions are permitted
  - Missing a time constraint is considered a failure to meet requirements



# Hard Real-Time, Soft Real-Time

- **Soft Real-Time**
  - At least three kinds of requirements, possibly in combination:
    - Time constraints may be missed by only small amounts (usually expressed as a percentage of the time constraint)
    - Time constraints may be missed infrequently (usually expressed as a percentage of instances)
    - Occasional events or periodic instances may be skipped (usually expressed as a probability)
  - Resources must be managed to meet the stated requirements



# Timing Requirement Sources

- **Time Constraint Requirements come from:**
  - **Explicit Top Level Requirements, e.g.,**
    - Display a video frame within 33 milliseconds.
  - **Derived Requirements, e.g.,**
    - **Accuracy** – “Maintain aircraft position to within 1 meter => Periodicity”
    - **Fault Tolerance** – “Recover from message loss within 500 ms.”
    - **Human Computer Interface Requirements** –
      - Process switch depressions within 250 ms.
      - Refresh display within 68 ms.



# Real-Time Scheduling

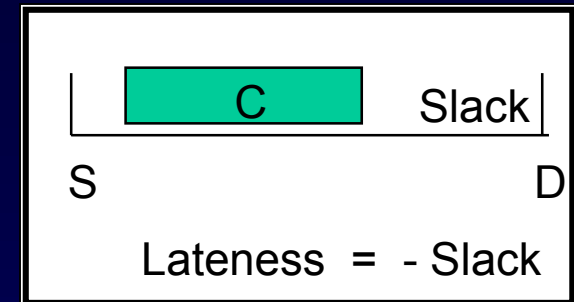
- **The Principal Difference Between a Real-Time and Non-Real-Time System.**
- **Definition:**
  - The process of sequencing shared resource allocations to meet user's time constraints.
- **The Principal Three Goals:**
  1. Meet all application time constraints, if feasible.
  2. Meet all important time constraints if meeting all time constraints is not feasible.
  3. Be able to accurately predict how well goals 1 and 2 are met for any given process load.

*Real-Time ≠ Real-Fast*



# Known Real-Time Scheduling Algorithms

- **Shortest Processing Time First (SPT)**
  - Minimizes mean lateness
  - Optimal for Goals 1 and 3, stochastically
- **Earliest Deadline First (EDD, or Deadline)**
  - Minimizes maximum lateness
  - Optimal for Goals 1 and 3 (fails disastrously on overload)
- **Smallest Slack Time First (Minimum Slack or Minimum Laxity)**
  - Maximizes minimum lateness
  - Optimal for Goals 1 and 3 (fails disastrously on overload)





# Known Real-Time Scheduling Algorithms

- **Rate Monotonic Scheduling (RMS)**
  - Optimal for fixed priority scheduling – fulfills all real-time goals for mostly periodic processing domains
  - Approximates EDD with reduced utilization bound
  - Reserves computation capacity to ensure predictability



# Scheduling in Real-Time CORBA 1.0

- **Assumes Static Fixed Priority Scheduling**
  - Threads have priority based on time constraints - underlying theoretical basis
    - Rate Monotonic Scheduling
    - Deadline Monotonic Scheduling
  - Fixed Priority means priorities change only
    - To prevent priority inversion
    - When required by application
- **Upcoming Dynamic Scheduling spec**
  - Accommodates dynamic loads
  - Allows schedulers to be replaced



# Scheduling Example

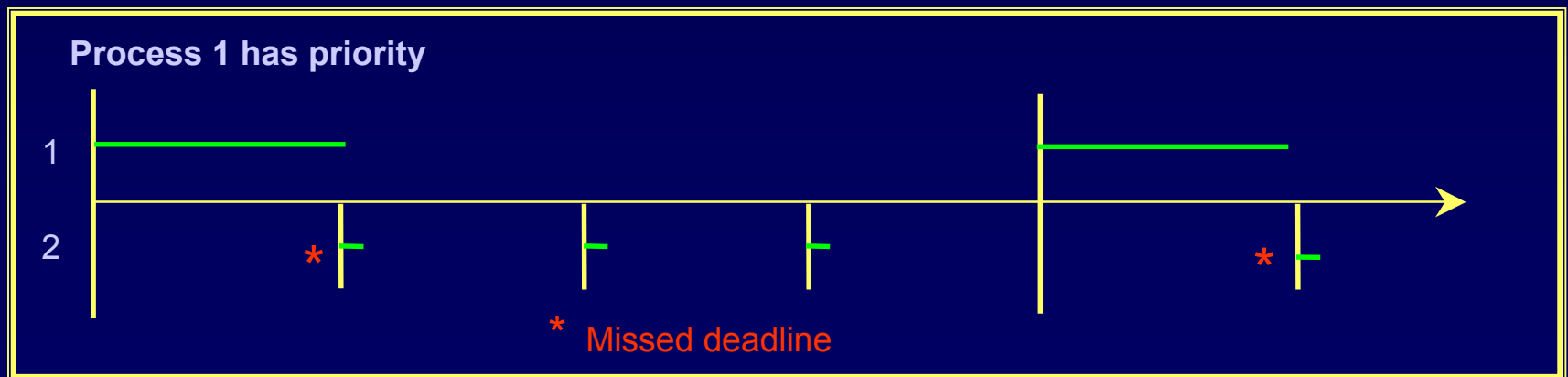
## World's Lowest Budget Nuclear Power Plant

- **Process 1** => **Reactor Control Rods** - needs 1 second every 4 seconds
- **Process 2** => **Dishwasher Water Valve** - needs 0.01 second every second

Two Preemptible Processes:

Process 1 – utilization = 25%  
Process 2 – utilization = 1%

} Total Utilization  
26 %





# Scheduling Example

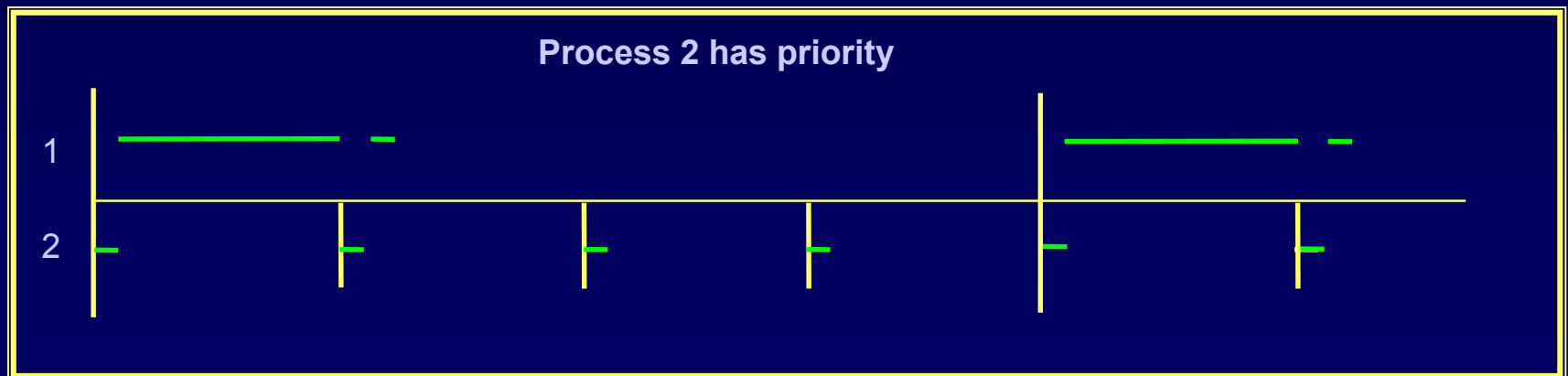
## World's Lowest Budget Nuclear Power Plant

- **Process 1** => **Reactor Control Rods** - needs 1 second every 4 seconds
- **Process 2** => **Dishwasher Water Valve** - needs 0.01 second every second

Two Preemptible Processes:

Process 1 – utilization = 25%  
 Process 2 – utilization = 1%

} Total Utilization  
 26 %





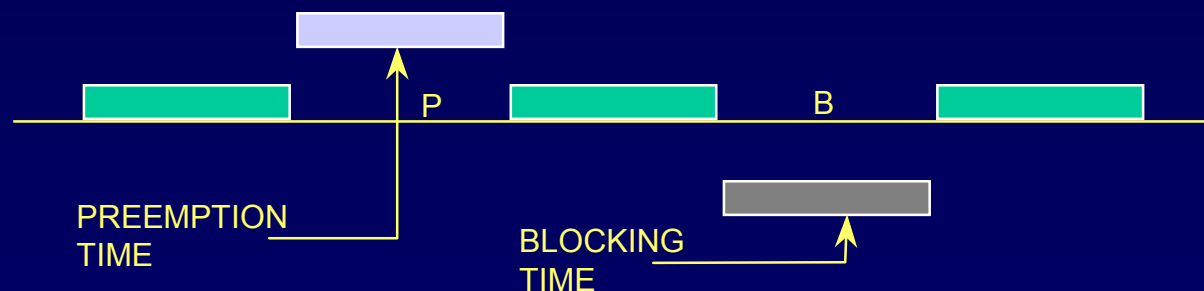
# Shared Resource Concepts

- Preemption.
  - Execution delayed by higher priority tasks
- Blocking (Priority inversion)
  - Execution delayed by lower priority tasks
- Mutual Exclusion (Mutex)
  - Sequenced access to a shared resource, typically implemented by locking and waiting for locks.
- Critical Section
  - Execution while holding a lock.

HIGHER PROCESS

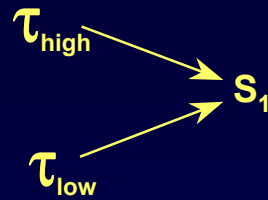
MY PROCESS

LOWER PROCESS





# Shared Resources and Priority Inversion



High and low priority tasks share a critical section- Can lead to **UNBOUNDED** Blocking



■ Normal execution     ■ Execution in Critical Section  
■ Unbounded Priority Inversion



## Concurrency & CORBA

- **Concurrency is fundamental to real-time systems**
  - CORBA extends the concurrency model across systems
  - Requires resource controls, priority propagation
- **Concepts supported by RT CORBA**
  - Threadpools
    - Multiple, Static or Dynamic, Default priorities
  - Scheduling policies
    - Client to Server Propagation or Server Defined



# Communications Transports

- **Communication from client to server**
  - Represents yet another resource to be scheduled
  - Source of potentially unbounded delays
    - Long, low priority messages on shared media may block high priority messages
    - IIOP does not allow interleaving of request fragments
- **RT CORBA**
  - Supports multiple “banded” connections
  - Permits choosing among available protocols



# Real-Time Principles Summary

- Real-Time  $\neq$  Real-Fast
- Scheduling concepts are frequently counter-intuitive
- Resource management is the fundamental requirement to deal with response time issues, whether hard real-time or soft real-time



# Real-Time CORBA Specification

- History and Goals
- Requirements
- Architecture
- Example
- Future Work



# Real-Time CORBA Specification

- History and Goals
- Requirements
- Architecture
- Example
- Future Work



## History

- Real-Time CORBA RFP out 9/97
- Real-Time CORBA 1.0 accepted 7/99
- **OMG Document ptc/99-05-03**
  - <ftp://www.omg.org/pub/docs/ptc/99-05-03.pdf>

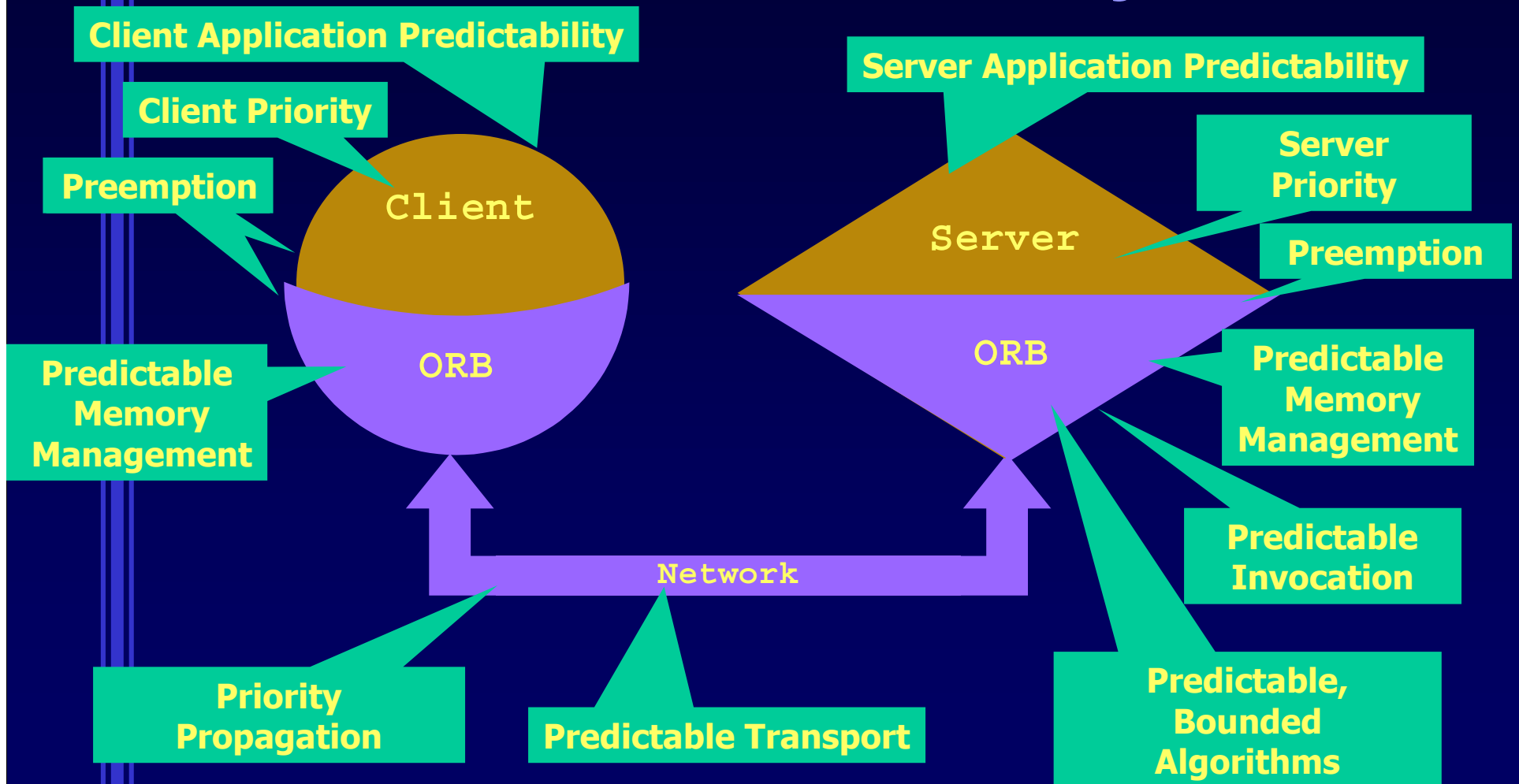


## Real-Time CORBA Goals

- Support developers in meeting real-time requirements by:
  - *Facilitating* end-to-end predictability
  - Providing application-level management of ORB resources
- Provide interoperability with traditional ORB's

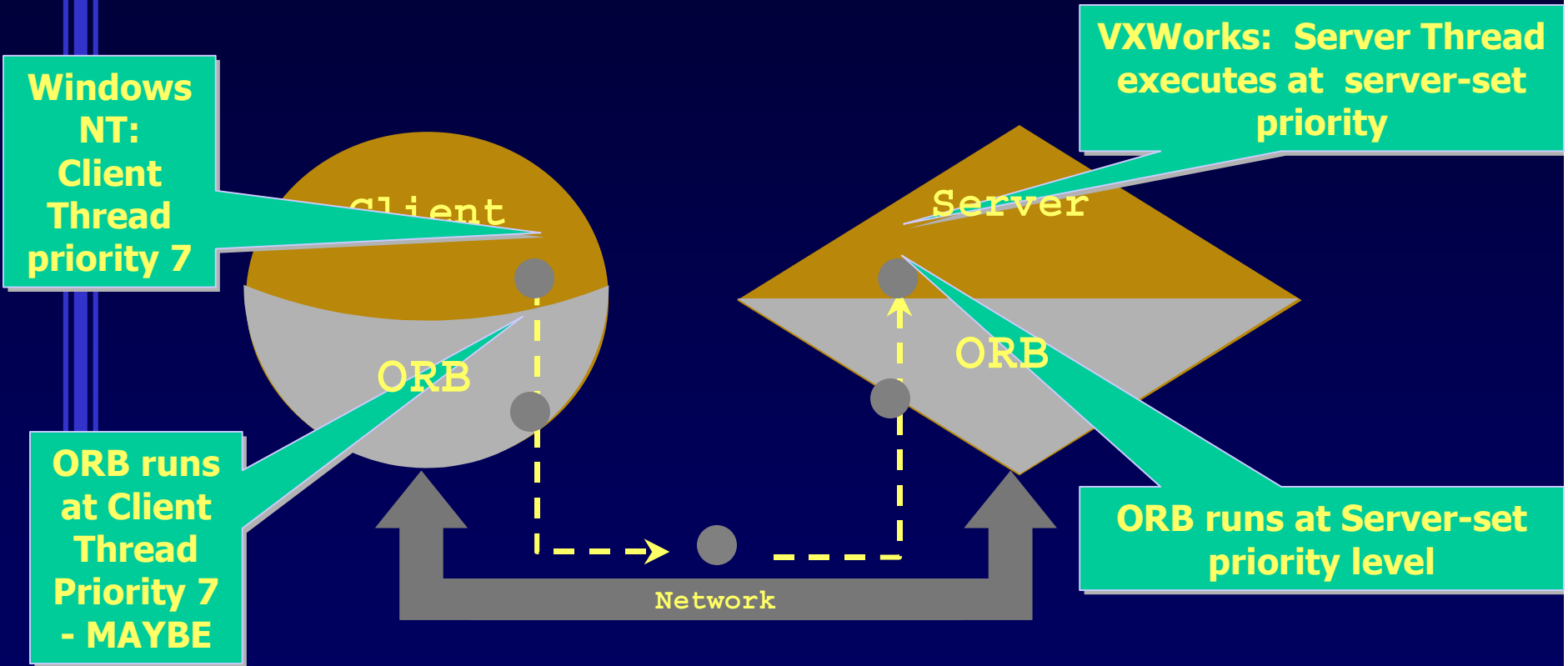


# Real-Time System Predictability





# Traditional CORBA



**Results are a Possible Priority  
Inversion**



## End-to-End Predictability

- **Respect thread priorities between client and server**
- **Bound duration of thread priority inversions**
- **Bound latencies of operation invocations**
- **Provide explicit control over ORB-managed resources**

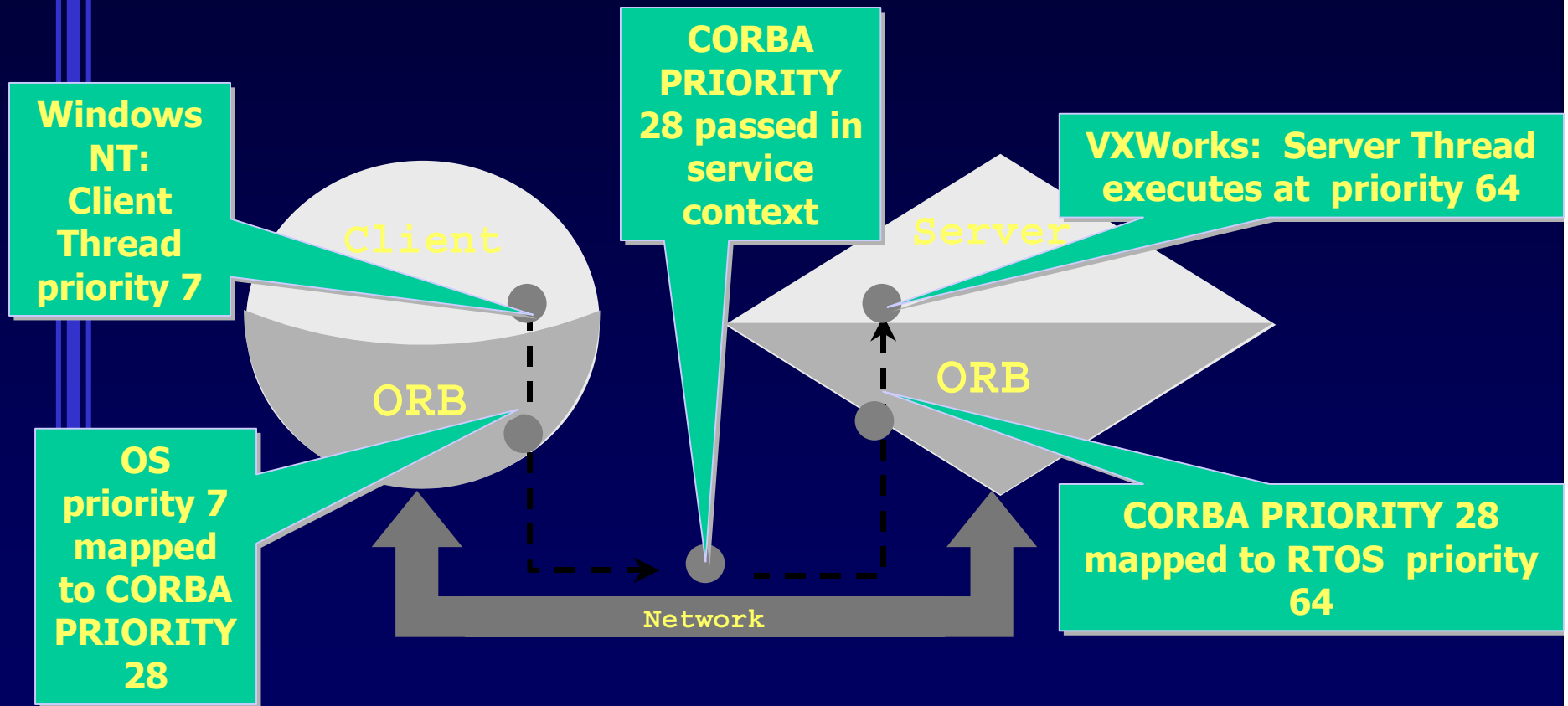


## Activities

- **“End-to-End” ... the ends of what ?**
- **“Activity” is a design concept**
  - defined according to the problem being solved
  - may span client and server
  - may span chains of client- server interactions
  - may be limited to chains within one node
- **“Activity” will be manifested at Run- time**
  - thread, on-the-wire message, buffered request



# CORBA Priority Propagation



**End-to-End Predictability for the Activity**



# Operating System Assumptions

- **Not a portability layer for heterogeneous Operating Systems**
- **POSIX is sufficient ... but not necessary**
  - **POSIX 1003. 1b- 1996 Real-time Extensions**
- **Extent to which the ORB is deterministic may be limited by RTOS characteristics**

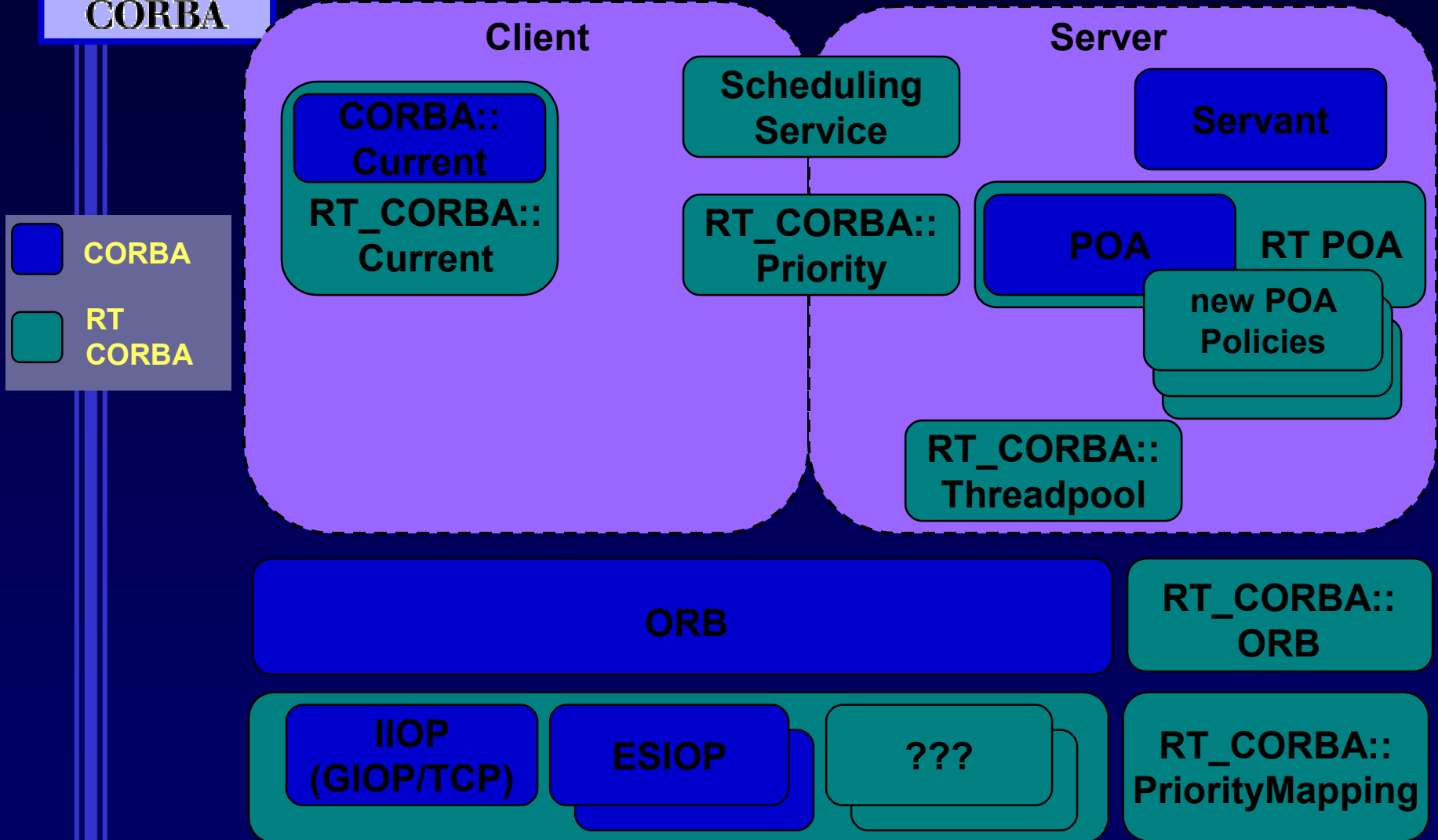


# Real-Time CORBA Specification

- History and Goals
- **Architecture**
- Example
- Future Work



# Real-Time CORBA Extensions





## RTCORBA::RTORB

- Conceptually an extension of the CORBA::ORB interface
- Specified just to provide operations to create and destroy other Real-Time CORBA entities
  - Mutex, Threadpool, Real-Time Policies
- One RTORB per ORB instance
- Obtain using:  
`ORB::resolve_initial_references("RTORB");`



## RTPortableServer::POA

- Adds operations to POA to allow setting of priority on a per-Object basis
- Implements policies for the RT POA
- Inherits from PortableServer::POA (so not PIDL)



## New POA Policies for RT

- **Priority Model Policy**
- **Threadpool Policy**
- **Priority Banded Connection Policy**
- **Private Connection Policy**
- **Server Protocol Policy**
- **Client Protocol Policy**



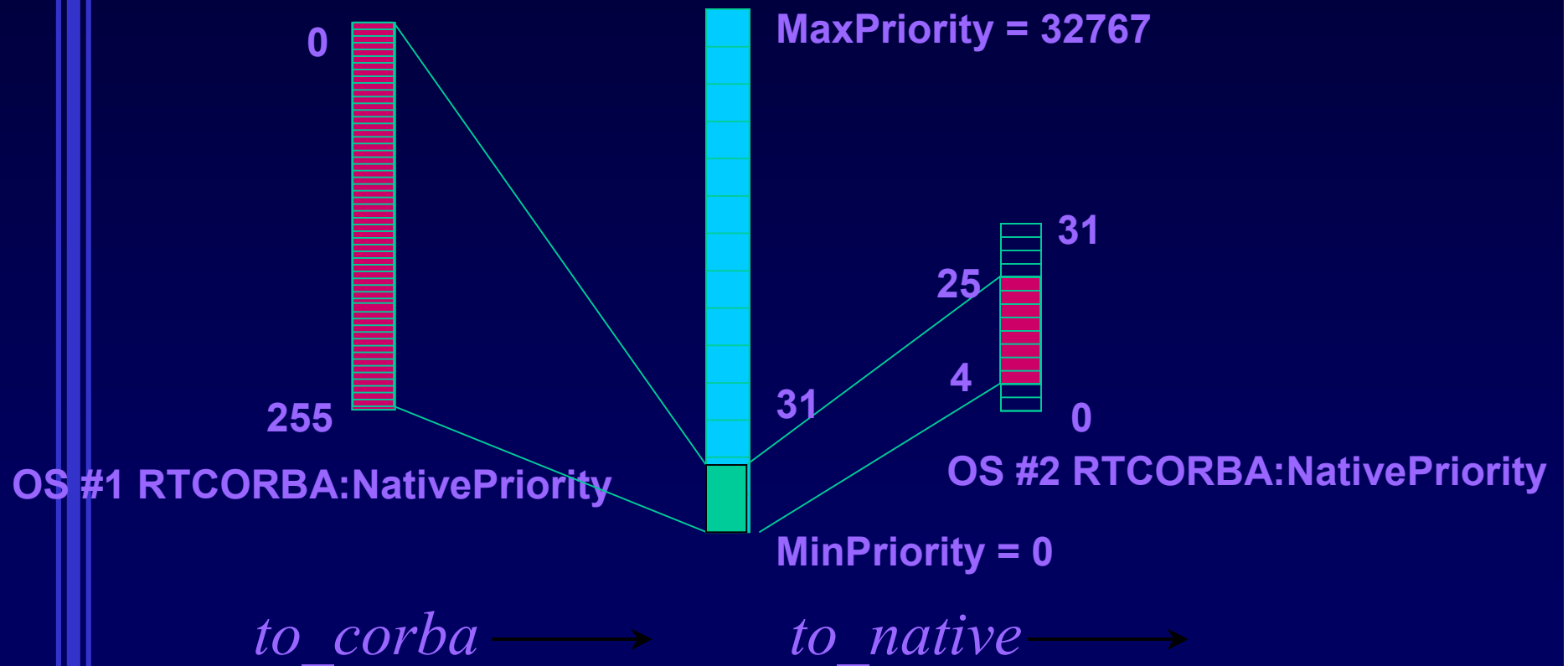
## RTCORBA::Priority

- **Universal, platform independent priority scheme**
- **Allows prioritized CORBA invocations to be made in a consistent fashion between nodes with different native priority schemes**



# Real-Time CORBA Priority Mapping

RTCORBA::Priority





# Priority Mapping

- **Default Priority Mapping**
  - Not standardized
  - Real-time ORB must supply a default mapping for each platform (RTOS) that the ORB supports
- **User-defined Priority Mapping**
  - Real-time ORB must provide method for user to override default mapping
- **One Priority Mapping installed at any one time per ORB instance**
  - installation mechanisms are not standardized



## Priority Model Policy

- **A Server-Side (POA) Policy**
  - configure by adding a **PriorityModelPolicy** to policy list in **POA\_create**
  - all objects from a given POA support the same model
- **Two Models Specified**
  - Client Propagated
  - Server Declared



# Client Propagated Priority Model

Client running  
at RTCORBA::Priority 7

Invocation handled  
at RTCORBA:Priority 7



Scheduling based on end-to-end activities across node boundaries.



# Server Declared Priority Model

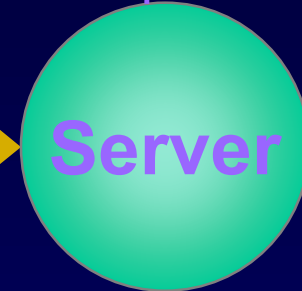
Client running  
at `RTCORBA::Priority 7`



Client's priority **is not**  
propagated with invocation



Server Priority  
is pre-set



Invocation handled  
at pre-set Server  
priority

Scheduling based on relative priorities of different objects on the same node. Can set Server Priority on per object basis.



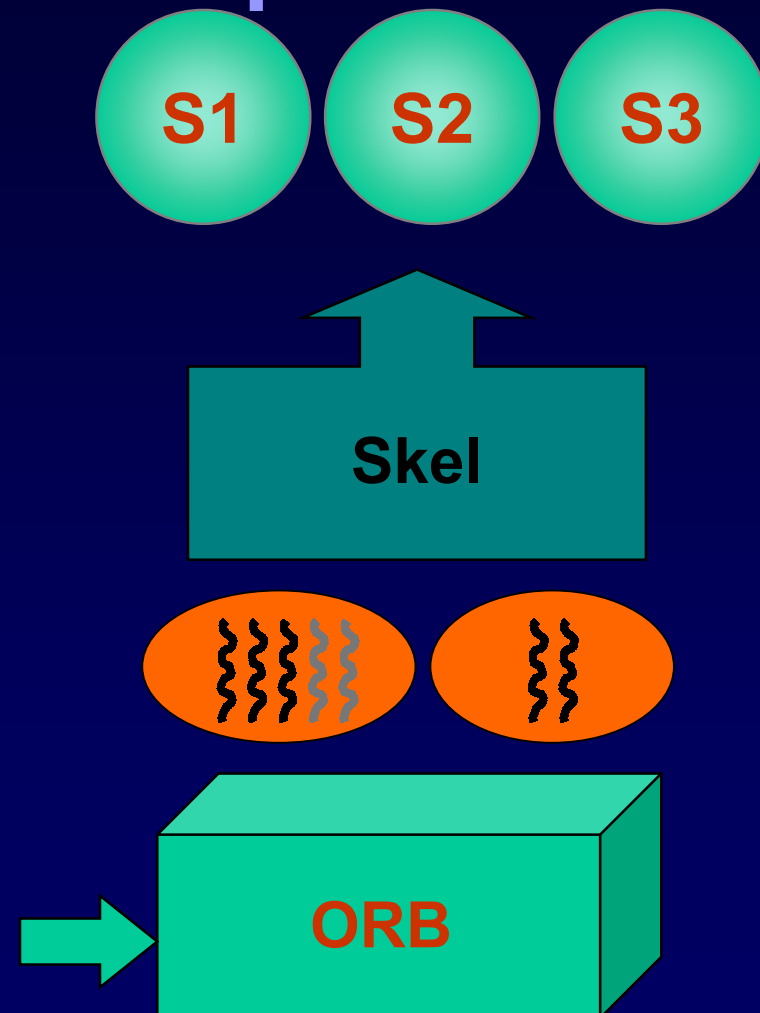
## Real-Time CORBA Mutex

- API that gives the application access to the same Mutex implementation that the ORB is using
  - important for consistency in using a Priority Inheritance Protocol  
e.g. Priority Ceiling Protocol
- The implementation must offer (at least one) Priority Inheritance Protocol
  - No particular protocol is mandated though
  - application domain and RTOS specific



# Threadpools

- **Threadpool Benefits**
  - Control invocation concurrency
  - Thread pre-creation and reuse
  - Partition threads according to Priority
- **Multiple Threadpools**
  - System partitioning
  - Protect resources
  - Integrate different systems
  - more predictably





## Threadpool Policy

- A POA may only be associated with one Threadpool
- Threadpools may be shared between POA's
- Threadpool parameters
  - number of static threads
  - dynamic thread limit
    - 0 = no limit. same value as static = no dynamic threads
  - thread stacksize
  - default thread priority



## Laned Threadpools

- **Alternate way of configuring a Threadpool**
  - for applications with detailed knowledge of priority utilization
  - preconfigure ‘lanes’ of threads with different priorities
  - ‘borrowing’ from lower priority lanes can be permitted

without  
lanes

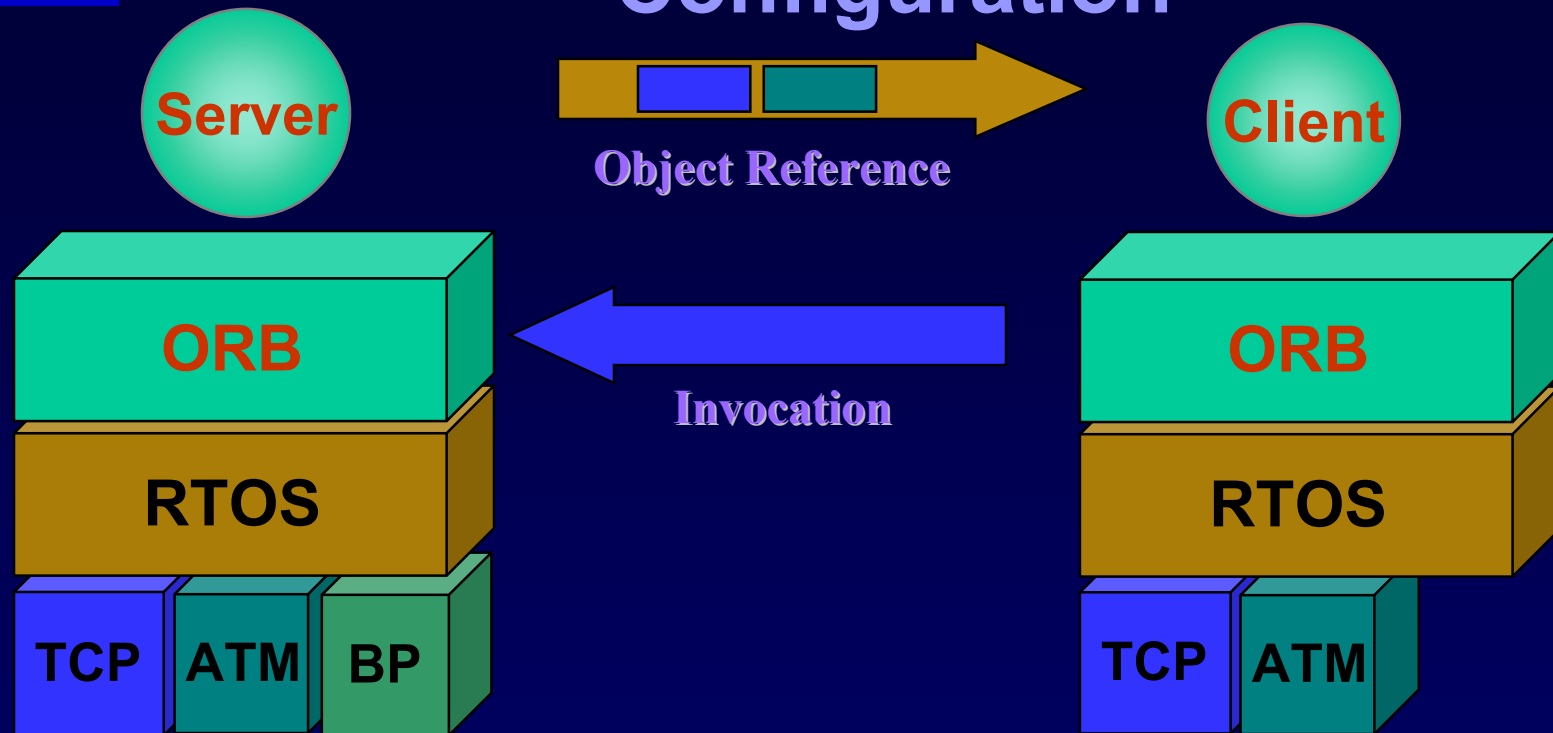
priority = 5  
static = 15  
dynamic = 15

with  
lanes

p = 5	p = 6	p = 8
s = 5	s = 5	s = 5
d = 5	d = 5	d = 5



# Protocol Selection and Configuration



On Server, this policy indicates which protocols to publish in Object Reference

On Client, this policy Indicates which protocol to use to connect



## Server Protocol Policy

- Enables selection and configuration of communication protocols on a per-POA basis
- Protocols are represented
  - By a unique id
  - And by Protocol Properties defined as ORB/Transport level protocol pairs
- `RTCORBA::ProtocolList` allows multiple protocols to be supported by one POA
  - Order of protocols in list indicates order of preference



## Protocol Properties

- **A Protocol Properties interface to be provided for each configurable protocol supported**
  - allows support for proprietary and future standardized protocols
- **Real-Time CORBA only specifies Protocol Properties for TCP**



# TCP Protocol Properties

```
module {  
    interface TCPProtocolProperties  
        : ProtocolProperties {  
        attribute long    send_buffer_size;  
        attribute long    rcv_buffer_size;  
        attribute boolean keep_alive;  
        attribute boolean dont_route;  
        attribute boolean no_delay;  
    };  
};
```



## Client Protocol Policy

- Same syntax as server-side
- On client, `RTCORBA::ProtocolList` specifies protocols that may be used to make a connection
  - order indicates order of preference
- If Protocol Policy not set, order of protocols in target object's IOR used as order of preference



# Connection Management

## Connection Multiplexing

Offered by many ORB's for resource conservation.

Can contribute to priority inversion

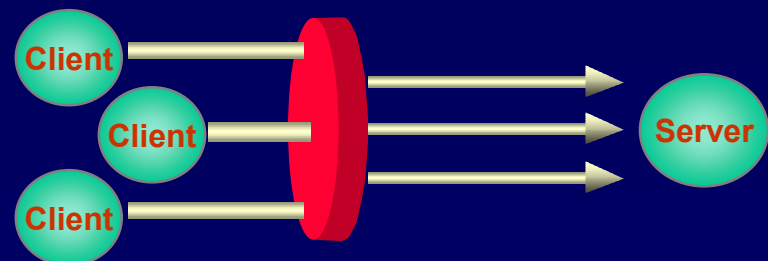
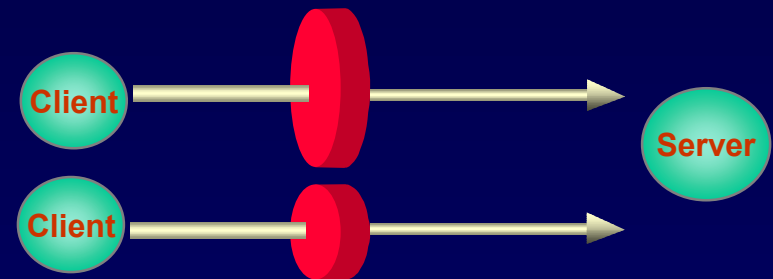
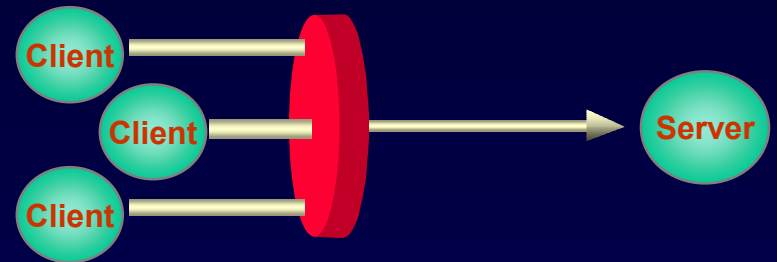
## Private Connection Policy

Guarantees dedicated connection

## Priority Banding Policy

Several connections between nodes

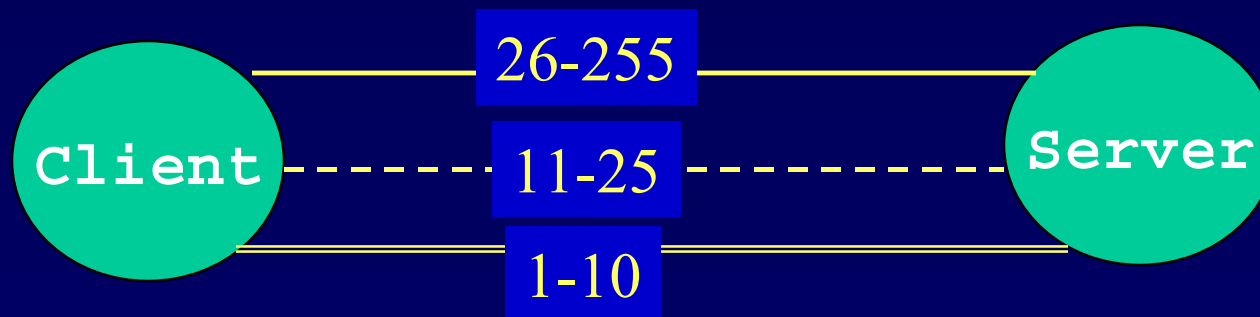
Connection to be used based on priority





# Priority Banding Connection Policy

- Multiple connections, to reduce priority inversion
  - each connection may represent a range of priorities
  - may have different ranges in each band, including range of 1





## Real-Time CORBA Scheduling Service

- **Effective Real-Time scheduling is complicated**
- **To ensure a uniform scheduling policy, such as global Rate Monotonic Scheduling, requires:**
  - **the Real-Time CORBA primitives must be used properly, and**
  - **their parameters be set properly in all parts of the CORBA system**



## Scheduling Service (continued)

- The Scheduling Service API abstracts away from the low-level realtime constructs
- Allows use of scheduling analysis tools, that support this API



# Real-Time CORBA Specification

- History and Goals
- Architecture
- Example
- Future Work



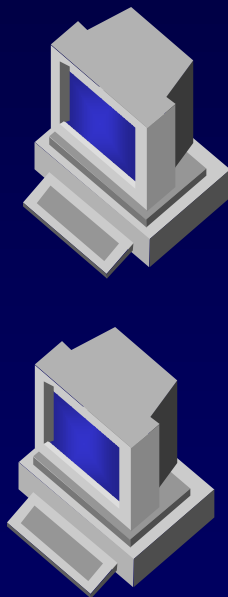
## Do I Need Real-Time CORBA?

- You need Real-Time CORBA if:
  - You use priorities in your distributed system
  - You want application-level control of the resources used by the ORB, including connections and connection-related resources

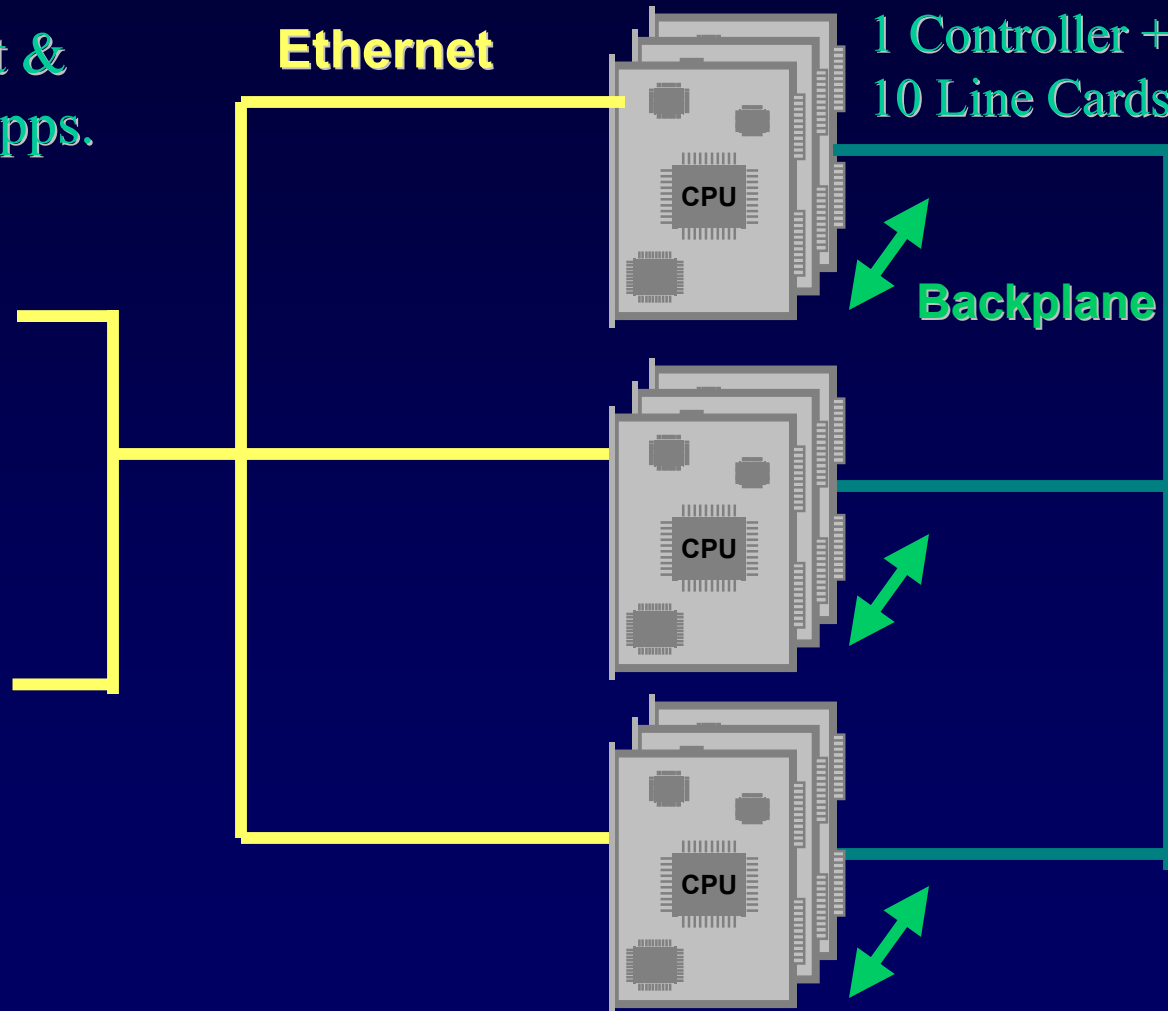


# Sample Real-Time CORBA System

Management &  
Enterprise Apps.



Ethernet





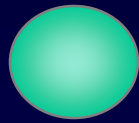
# Scheduling Analysis

- **Controller Card**
  - Alarm Receiver can receive alarms asynchronously at a rate of 10 per second from the 10 line cards
  - Controller can receive data from the Management Node at a rate of up to 10 messages per second
  - Other Apps run in background at maximum rate of 5Hz
- **Line Card**
  - Alarm can send no more than 10 Alarms per second
  - Video rate is 30Hz
  - Audio rate is 100Hz



# Real-Time Design Considerations

## Management Node



Manager

Alarm Receiver

High Priority,  
Own Threadpool

Controller

Medium Priority,  
Own Threadpool

App. Objects

Lower Priority,  
Share Threadpool

Alarm

Low Priority,  
Own Threadpool

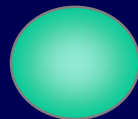
Video

Medium Priority,  
Own Threadpool

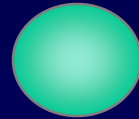
Audio

High Priority,  
Own Threadpool

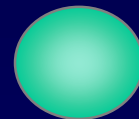
## Controller Card



Controller

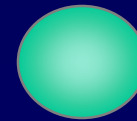


Alarm  
Receiver

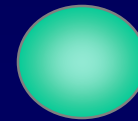


App.  
Object1

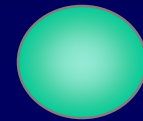
## Line Card



Video



Audio



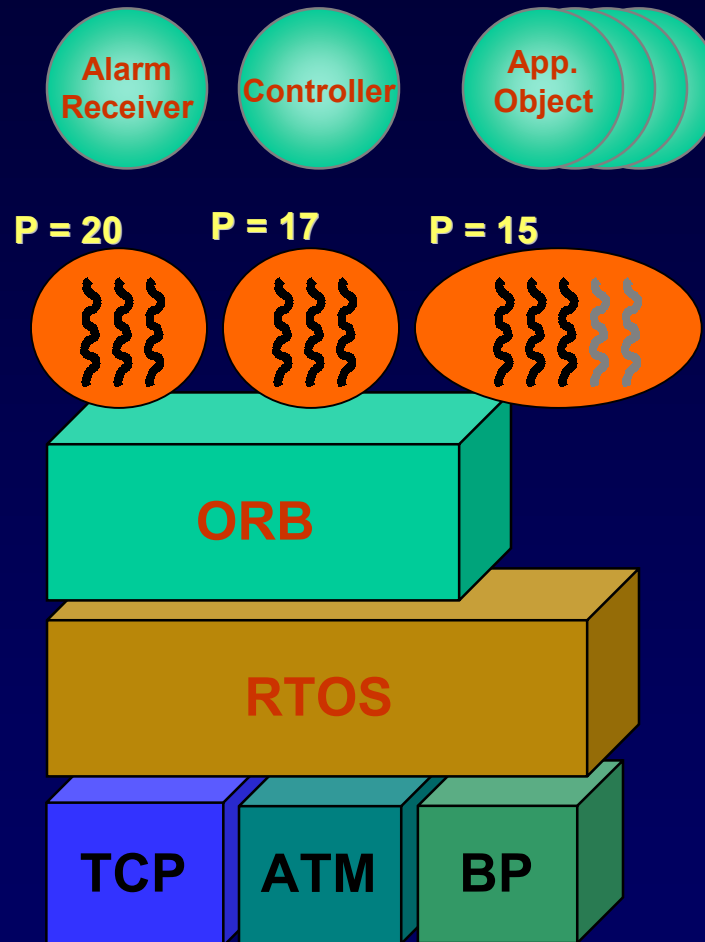
Alarm

- 1) ATM
- 2) Ethernet

- 1) Backplane
- 2) ATM

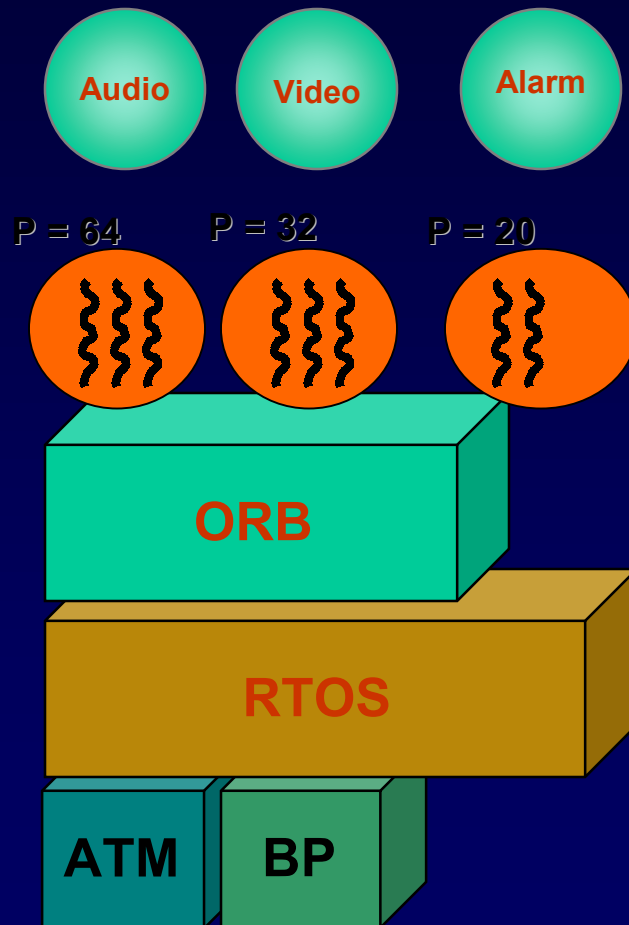


# CORBA on Controller Card





# CORBA on Line Card





# Real-Time CORBA Specification

- History and Goals
- Architecture
- Example
- Future Work



# Future Work on Real-Time CORBA

- **Dynamic Scheduling**
  - Will allow for a replaceable scheduler
  - Will allow optimization of real-time applications with dynamic loads
  - Represents completion of Real-Time CORBA capabilities