Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

ANSAwise - CORBA and Real-Time Systems

Chris Mayers

Abstract

Organizations building real-time systems may wish to exploit distributed object technology and integrate real-time and non-real-time systems.

This module of the ANSAwise training programme describes the work done in ANSAware/RT and further planned in ANSA Phase III.

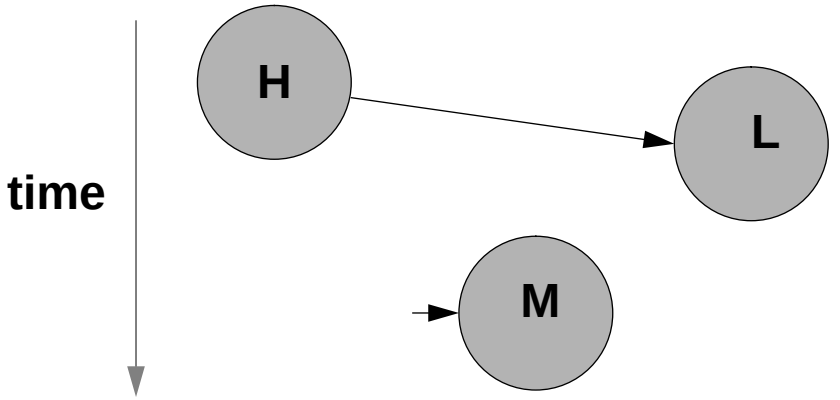
[This presentation draws on material from APM.1452. It also reuses some material from APM.1353 The graph of RPC performance has been dropped from later versions of this document here, because the graphic does not have a preview image.]

APM.1544.02 **Approved** 22nd May 1996

External Paper

Distribution:
Supersedes:
Superseded by:

CORBA and Real-Time Systems



In this session

- **Identify the characteristic properties of real-time systems**
- **Show how these properties affect the design of distributed systems**
  - **distributed systems with real-time properties**
- **Show how CORBA real-time systems will evolve**

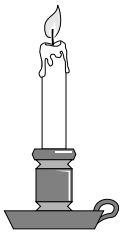
Timeliness

- A real-time system is one in which timing is explicitly considered
  - in requirements
  - in specification
  - in design
  - in implementation
- Real-time systems require timely responses to events
  - even under failure conditions
  - even under extreme load conditions

The nature of real-time systems

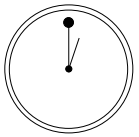
- **Distinctions are usually made between**
  - **non-real-time systems**
  - **soft real-time systems**
  - **hard real-time systems**
- **The distinctions are not rigid, but are realistic**
  - **the different trade-offs currently result in very different system designs**

Non-real-time systems



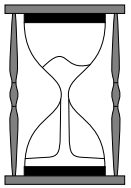
- **Have no timing requirement**
  - **best-effort is all that is offered**
- **Timings are not monitored**

Soft real-time systems



- Typically have an “average case” timing requirement
  - for example, 99% of requests to take less than 1 second
- Failure to meet the timing requirement is not critical
  - a late response to a request is bad, but may still be useful
- Timing may be monitored
  - consistent failure to meet a timing requirement may require human or automatic intervention
- Often based on priority schemes

Hard real-time systems



- Typically have a “worst case” requirement
  - for example, all requests must be completed in less than 1 second
- Failure to meet timing requirement is catastrophic
  - a late response is useless or even dangerous
- Timing must be monitored
- Failure to meet a deadline requires automated handling
- Based on deadline schemes, usually using priority as well

Misconceptions about real-time systems

- **Real-time Means Fast**
  - ***NO*: real-time means predictable timing/*'fast enough'*...**
  - **... acceptable response times in seconds are not unusual**
- **Real-time Means Assembler**
  - ***NO*: selective programming in assembler may be necessary for performance**
  - **... but it doesn't guarantee predictable timing**
- **Faster Hardware Solves All Real-time Problems**
  - ***NO*: faster hardware may make new problems solvable**
  - **... but it doesn't guarantee predictable timing, either**

More misconceptions

- **Real-time is Art, not Science**
  - **It may have been in the past...**
  - **... this is no longer so**
- **Real-time Problems Are Solved By Conventional Techniques**
  - **The nature of timing constraints poses special challenges for systems designers...**
  - **... challenges we can handle**

Real-Time Architecture

- **Interoperability**
  - **between applications with different real-time requirements**
  - **... executing on different real-time platforms**
  - **... interworking with non-real-time applications**
- **Portability**
  - **of applications between real-time platforms**
- **Support for continuous information flows (multimedia streams)**

Real-Time Design Principles

- **Separate Resources**
- **Reserve Resources**
- **Control Sharing**
- **Let the Application be in Control**

Resource reservation is a key facility

- **Reservation of resources (pre-allocation) is vital to provide guarantees...**
  - **just as in real life generally**
- **...but computers often try to share resources**
  - **by multiplexing**
  - **by pooling**
  - **by time-sharing**
- **It must be possible to reserve resources selectively**

Resources are needed immediately

- **Real-time means that resources must be delivered immediately they are required**
  - **Waiting Is Evil!**
- **Resources include**
  - **CPU**
  - **Memory**
  - **I/O bandwidth**

Resource reservation in distributed systems

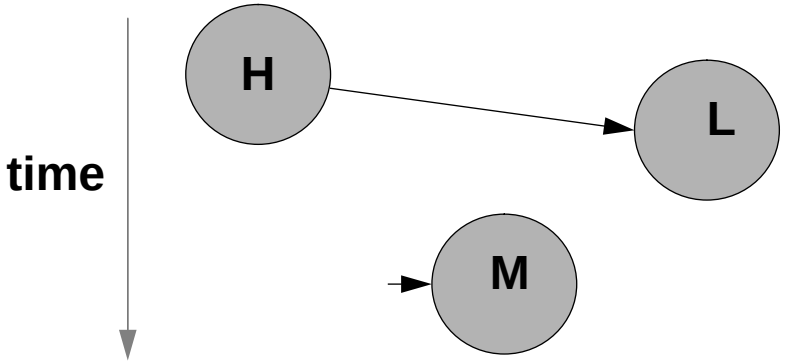
- In distributed systems, the resources must be reserved end-to-end
  - in the end systems
  - throughout the network
- This means end-to-end quality-of-service (QoS) guarantees

Programming with priorities

- Every multi-tasking operating system offers ‘task priorities’...
  - ... but exactly what do they mean?
- Many such systems evolved from time-sharing environments
  - offering round-robin scheduling (task quanta) for tasks of equal priority
  - ... often with automatic adjustment of task priority
- Real-time systems require strict priority
  - highest priority first
  - no round-robin scheduling
- Some systems offering ‘real-time’ priorities give you only this
  - but strict priority is not enough

The priority inversion phenomenon

- Suppose we have a system with three priority levels
  - high, medium, and low



- High-priority process synchronizes with low-priority process
  - ... meanwhile, it can be pre-empted by a medium-priority process
  - ... this is not what is wanted

Solutions to priority inversion

- The scheduler's protocol must prevent priority inversions, typically using
  - PIP (Priority Inheritance Protocol)
  - PCP (Priority Ceiling Protocol)
- The POSIX 1003.1b/d standard requires PIP
  - with PCP as an option
- You'll need a scheduler that prevents priority inversions
  - you can buy such systems now

Priority is not a panacea

- **Priority is only a measure of importance**
  - **it is not a measure of urgency**
  - **... it does not give you deadline guarantees**
  - **... which are necessary for hard real-time systems**
- **Separate mechanisms are needed to support importance and urgency**
- **Priority does not directly help meet the average-case guarantee for soft real-time systems either**
  - **it's not clear how to allocate priorities at design time**
  - **... it's often done by tuning the priorities by hand in the final system**
  - **... this does not assure confidence in the system**

Priority is useful

- **Well supported**
- **Well accepted**
- **Well understood**

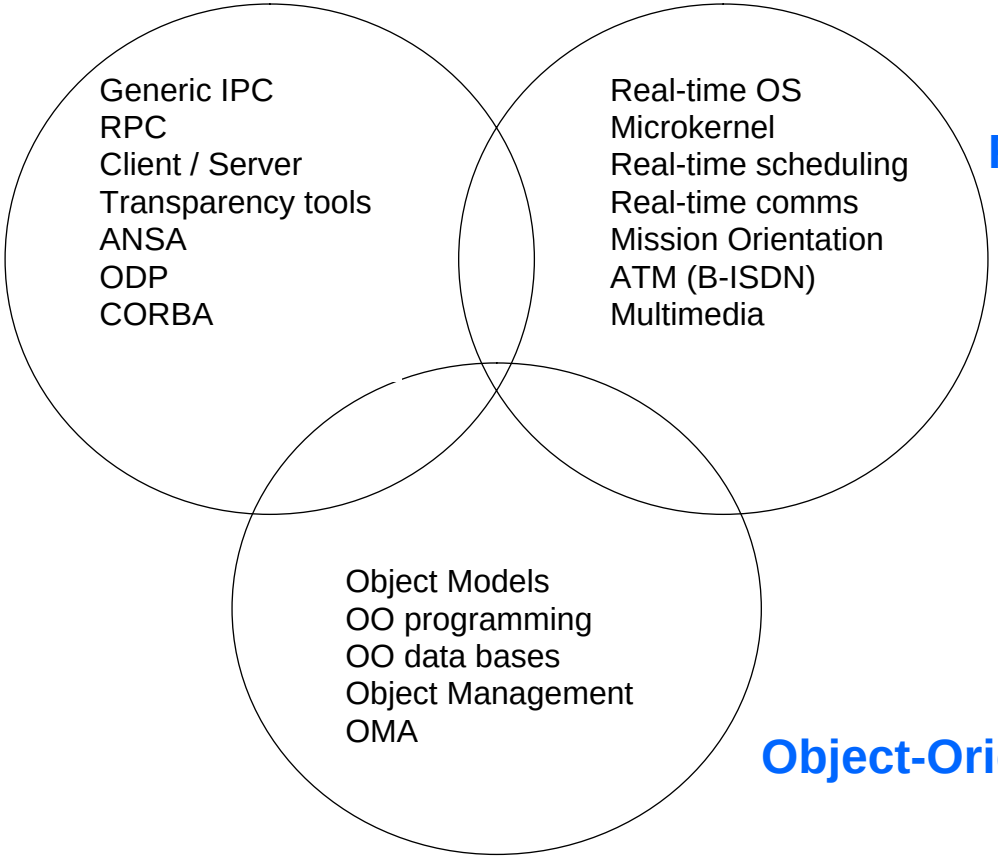
Priority in real-time distributed systems

- The same issues arise in a remote procedure call
  - strict priority
  - prevention of priority inversion
- This means that
  - both end systems must support this
  - the RPC must convey the priority information between the systems
- This is done with a modification to the RPC mechanism
  - making it a Timed RPC

Real-time Unix solves many problems

- The POSIX 1003.1* Real-Time extensions meet many of the needs
  - and convergence is beginning to deliver usable products
- There are still some gaps
  - no support for deadlines for hard real-time systems
  - no generalized support for resource reservation

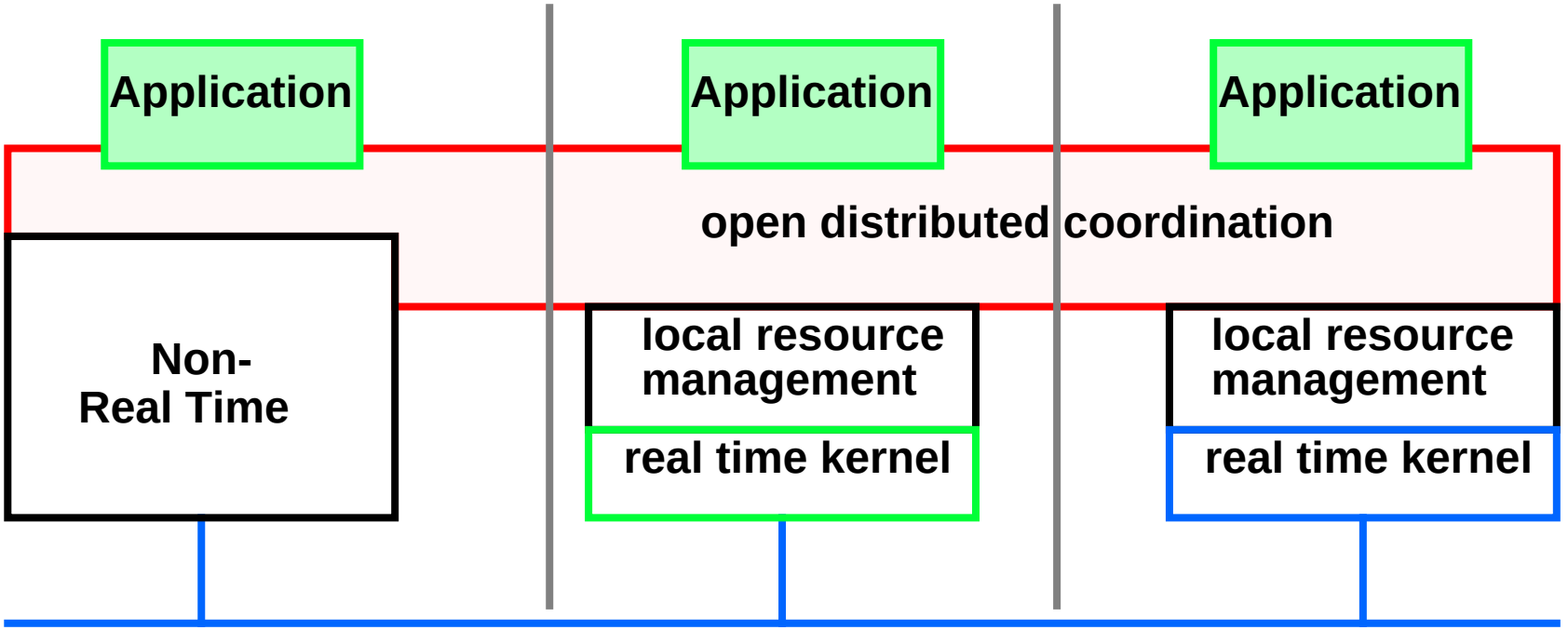
The wider picture - contributory technologies



ANSA Phase 3 Activities in Distributed Real-Time

- **ANSAware/RT**
  - **a version of ANSAware supporting some of the features described earlier**
  - **interworking with ANSAware 4.1**
- **Distributed Interactive Multimedia Architecture**
  - **re-engineered ORB infrastructure**

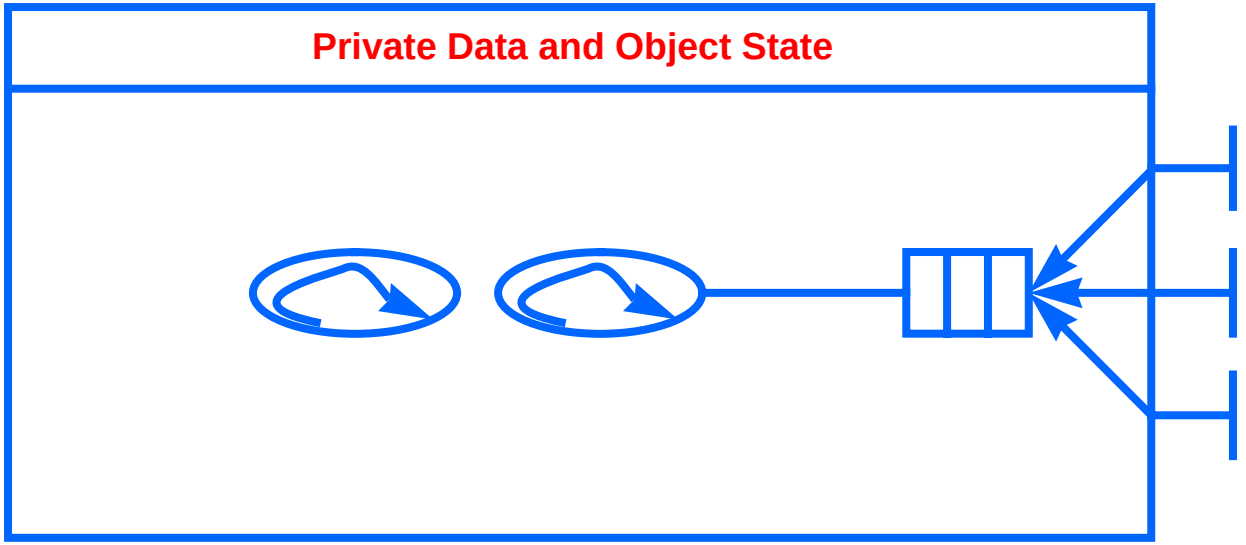
ANSAware/RT 1.0



ANSAware/RT

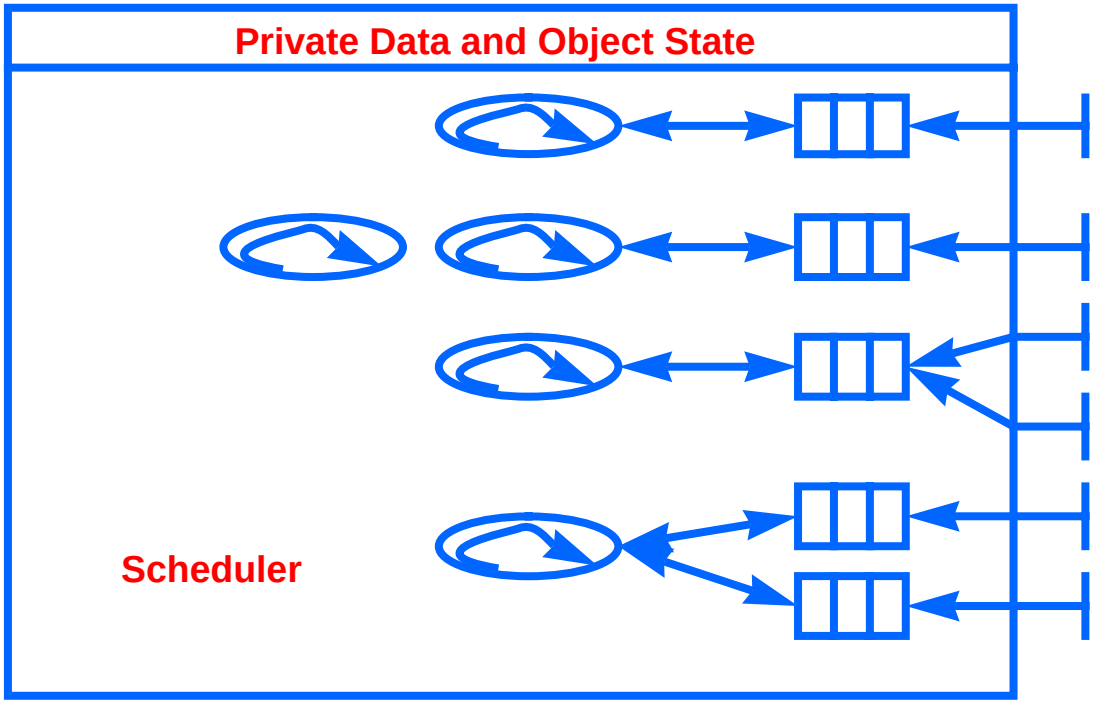
- **CPU resource control**
  - **by flexible tasking model and scheduling policies**
- **Communications resource control**
  - **by parallel protocol stacks and Timed RPC**
  - **with associated quality-of-service**

Tasking: ANSAware 4.1 Engineering Model



time-sharing based design --- efficient in resource sharing

Tasking: ANSAware/RT 1.0



more flexible design --- resource separation and reservation

Legend

Task Group

Task

Invocation
enqueue

Entry

Rendezvous

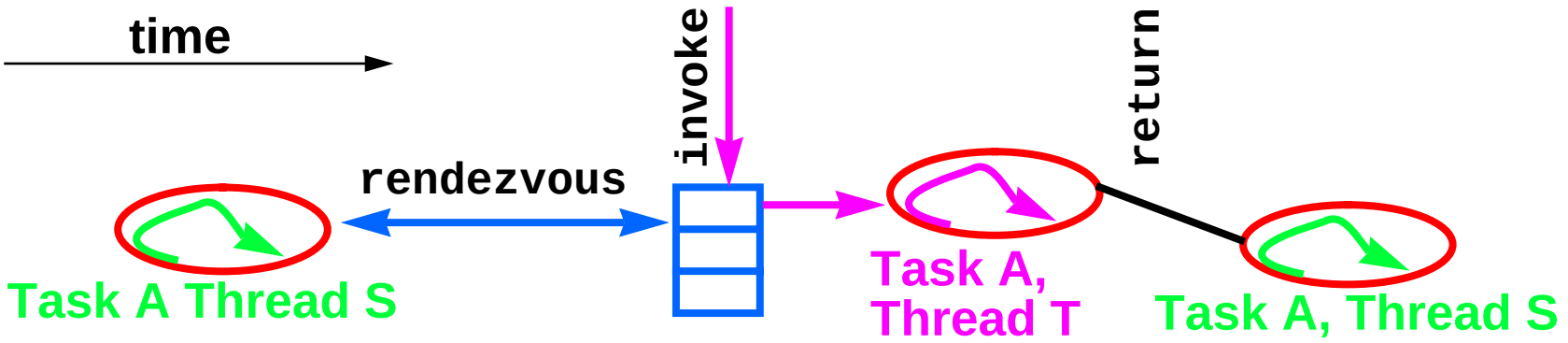
Interface

Entries

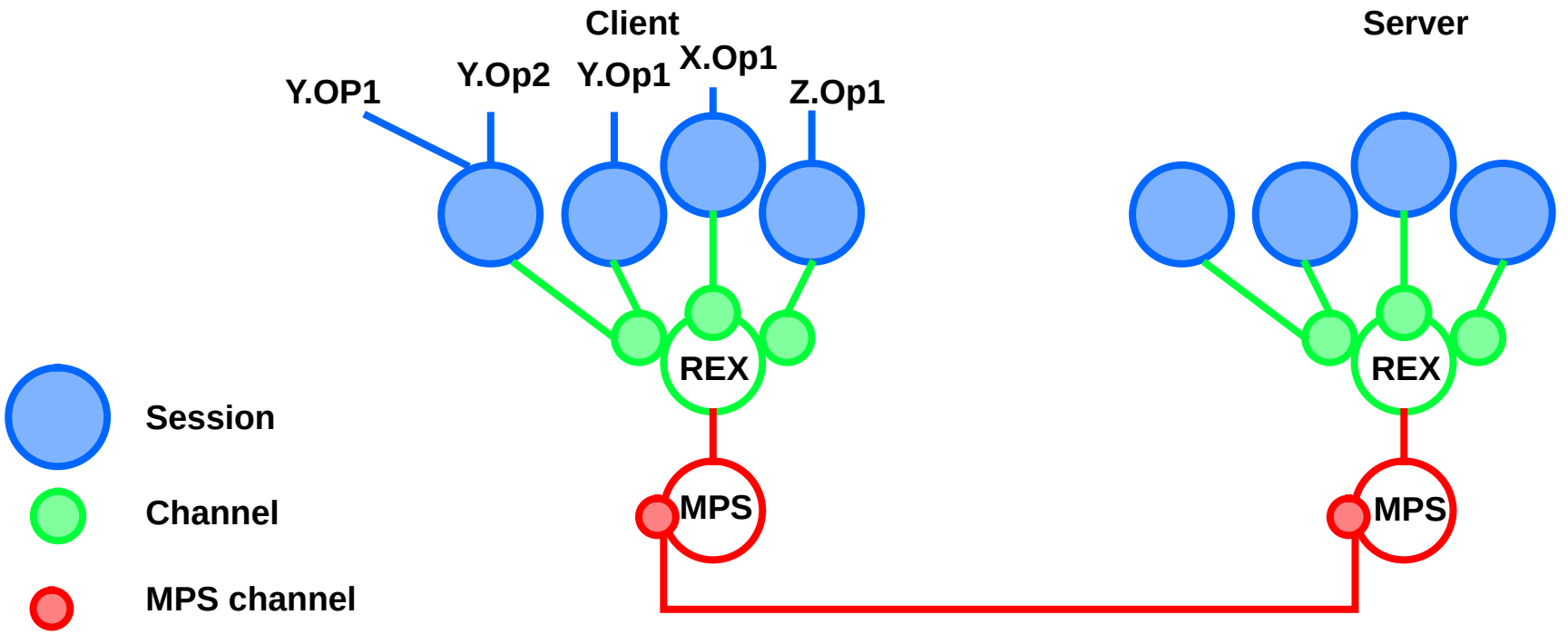
- An entry is a thread queue
  - a scheduling point with its own scheduling and processing policies
- There is a default entry to which all new interfaces are bound
- Applications can
  - create entries
  - bind interfaces to entries
  - unbind interfaces from entries (reverting to the default)
  - destroy entries

Rendezvous

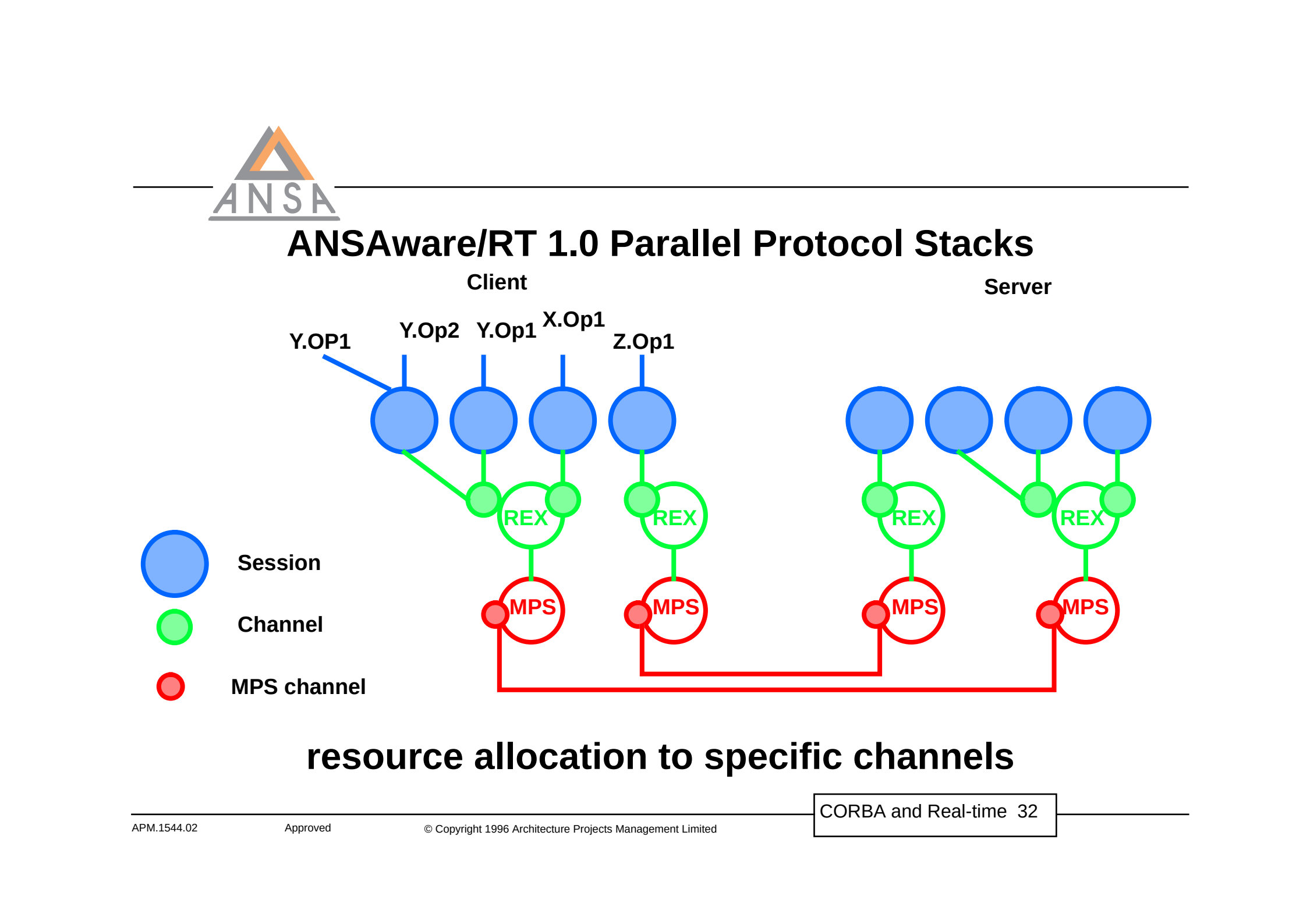
- A thread may wait at an entry to rendezvous and allow execution of another thread (e.g. an invocation)
  - the rendezvous can have an associated time-out



ANSAware 4.1 Communication System



multiplexing whenever possible --- efficient resource usage



ANSAware Timed RPC

- **TREX (Timed Remote EXecution) RPC protocol**
  - **efficient cut-down version of ANSAware REX RPC protocol**
  - **supports priority and deadline**

CORBA and Real-Time

- CORBA IDL does not allow specification of
  - quality-of-service
  - streams
- Resource reservation and control is an ORB engineering issue
  - vendor-specific
- A modular ORB infrastructure with standard interfaces is needed

CORBA Real-Time Products

- Teknekron Object Bus
- Chorus COOL/ORB
  - built on Chorus micro-kernel
- IONA Real Time Orbix
  - built on VxWorks

Related needs and issues

- **Typical related needs are**
  - **fault tolerance**
  - **automated recovery**
  - **load balancing**
- **No programming language provides adequate support for real-time**
  - **this will have to be provided by auxiliary tools**

General principles

- Design in real-time capability from the start
- Do not change the problem to fit the hardware
- Partition functions between objects with real-time aims in mind
- Specify testability interfaces
- Construct from general components into special configurations

Summary

- Distributed real-time is not yet part of CORBA standards
- Decide whether your needs can be met by specific systems
  - do not be fooled by features simply labelled as ‘real-time’
  - beware of systems that claim to offer simply ‘real-time’ priorities
- For more on this topic
  - see *Misconceptions about Real-Time Computing* by John A. Stankovic (IEEE Computer Oct 1988)