Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

An Open Architecture for Real-Time: Engineering Aspects (Nov 93 TC Presentation Slides

List of author names goes here

Abstract

Need some instructions here.

APM.1090.00.02 **Draft** 10 November 1993

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

An Open Architecture for Real-Time: Engineering Aspects (Nov 93 TC Presentation Slides

Request for Comments (confidential to ANSA consortium for 2 years)

An Open Architecture for Real-Time: Engineering Aspects (Nov 93 TC Presentation Slides

List of author names goes here

APM.1090.00.02

10 November 1993

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK	(0223) 323010
INTERNATIONAL	+44 223 323010
FAX	+44 223 359779
E-MAIL	apm@ansa.co.uk

An Open Architecture for Real-Time: Engineering Aspects

(Work in Progress Report)

Dave Otway

<djo@ansa.co.uk>

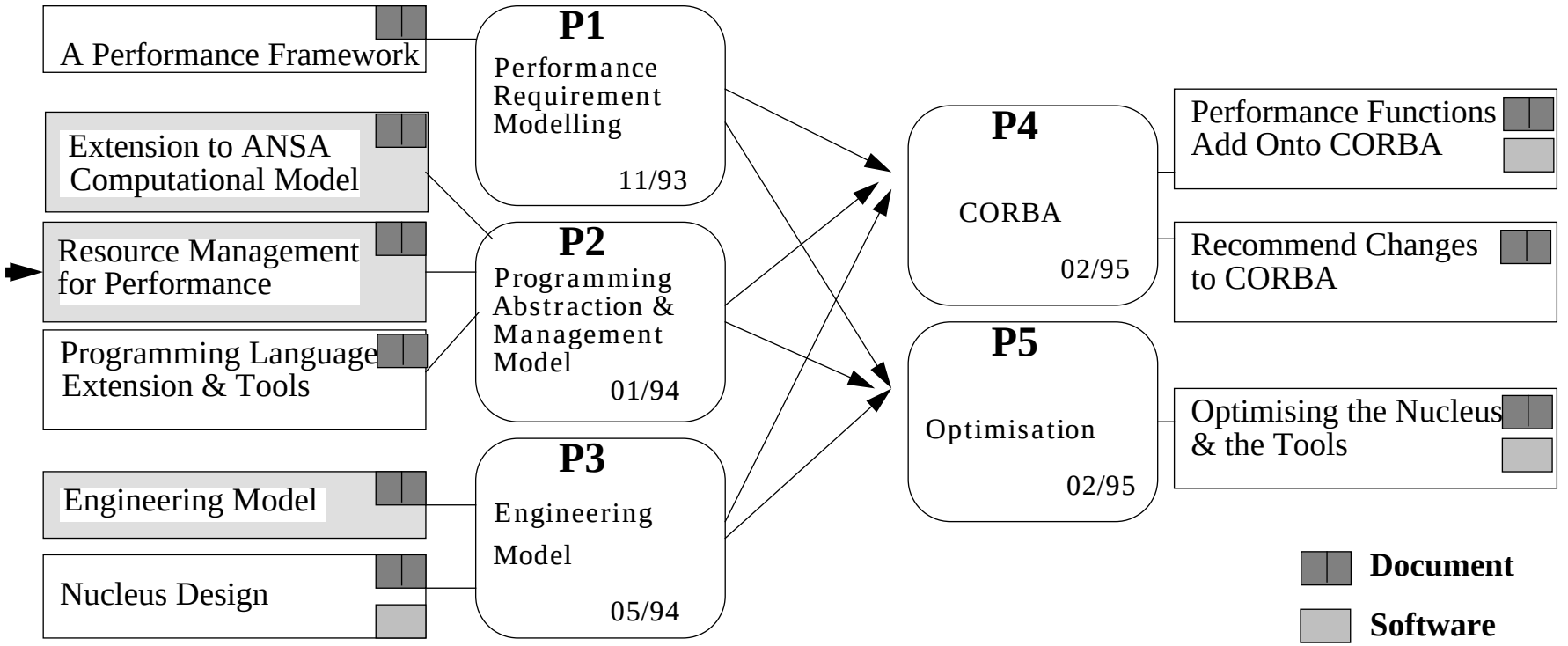
Guangxing Li

<gxl@ansa.co.uk>

Performance Task Group

<perftg@ansa.co.uk>

Group Activities



Objectives, Benefits and Outline

- **identifies real-time architectural issues in engineering aspects**
  - **background, rationale and context**
- **benefits**
  - **permits the application of open system architecture to real-time systems**
  - **identifies how and where real-time applications may constrain open systems**
  - **explains how real-time technologies are integrated with open systems**
- **presents the areas of work**
  - **programming model, real-time communication, temporal synchronization**
- **explains initial design**
  - **stress on a design that would stand on both *current* and future technologies**
  - **resource management, real-time scheduling, tasking, communications**
- **documented in *APM.1072***

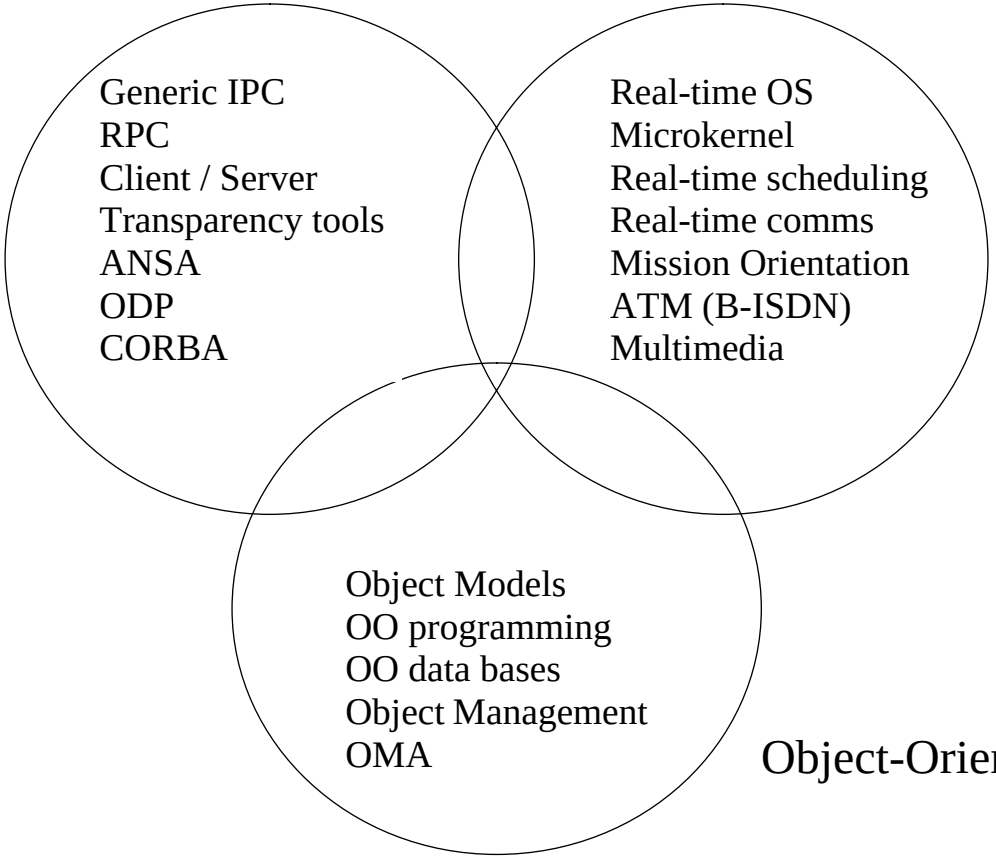
Engineering Initial Design: Context

- practical approach --- check feasibility --- able to demonstrate
- stress on extensions to AW based on available real-time technology
- gain experience of an execution environment for the RT computational model
  - which will be developed in parallel
- input to detailed engineering design
  - alongside an exploration of transparency issues and mechanisms
- explore the relationships between
  - programming environment
  - computational model
  - engineering model
  - execution environment
- guide for prototyping

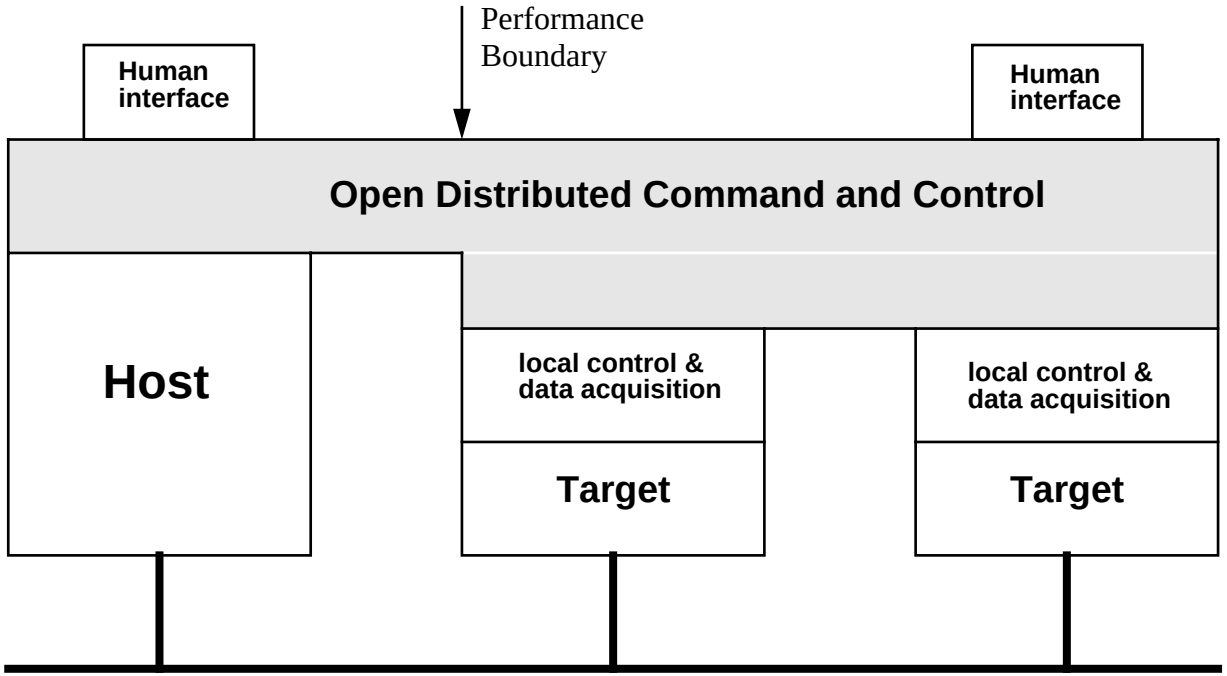
System Integration

- **real-time objects are *first class citizens***
- **a common platform for both real-time and non-real-time objects**
- **a *TOTAL* system design architecture**
- **importance of system integration**
  - **general purpose distributed computing environments are evolving towards real-time systems**
  - **real-time applications are evolving towards large distributed systems**
- **real-time applications increasingly need to cross organisational boundaries**
  - **all distributed systems will become federated**

Contributory Technologies



Target: Supervisory Control



Areas of Work

- **real-time programming model --- predictability, programmer control and mission orientation.**
  - **what differentiates real-time objects from non real-time objects**
  - **mapping real-time programming concepts into ANSA programming concepts**
- **resource management engineering**
  - **tasking, communications, synchronisation**
- **new object execution model**
  - **mapping tasks to interfaces and activities**
- **RPC based real-time communication system**
  - **predictability, timeliness and performance**
- **temporal synchronization --- predictability and timeliness**
  - **adding synchronous concepts**

Real-Time Programming Model

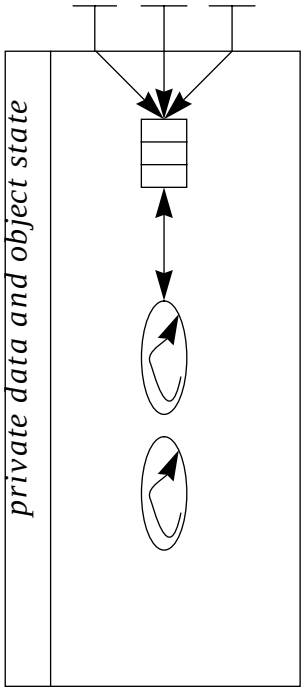
- **object oriented programming model can be divided into two parts**
  - **static part --- typing, interfacing, etc.**
  - **dynamic part --- object execution model**
- **real-time object oriented programming model (or what differs real-time objects from non-real-time objects)**
  - **needs a predictable (taking resource sharing into account) object execution model**
  - **programming resource requirement and management**
- **specifying resource requirement**
  - **QoS --- simple as priority or complex as multimedia QoS (jitter, volume etc.)**
  - **associated with activities, invocations, interfaces and bindings**
- **explicit resource management**
  - **explicit binding**
  - **resource allocation and scheduling (e.g., tasks and communication channels)**

Object Execution Model: Tasking

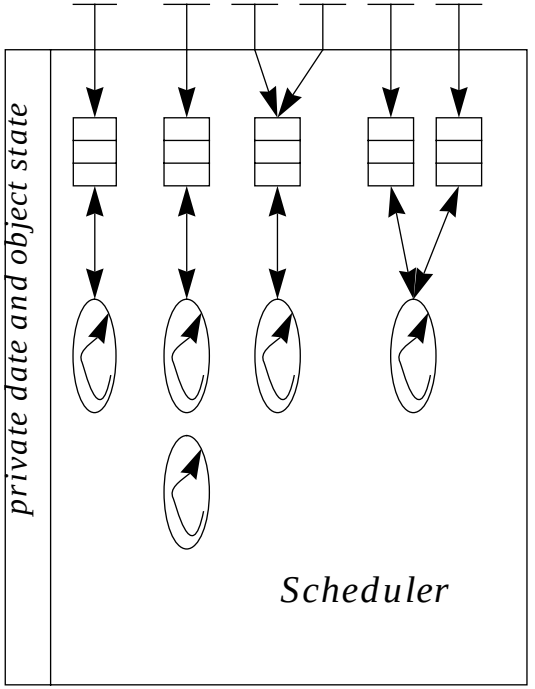
- goals
  - separation of concerns from scheduling point of view
    - the separation of real-time services from non-real-time services
    - the separation of different real-time services
  - permitting the pre-allocation of tasks with possibly pre-allocated special (application dependent) resources to services, e.g. a task with a specific QoS
  - supporting multiple (service based) scheduling schemes
  - enabling application direct control over scheduling
  - engineering --- policy/mechanism separation
- a design
  - a new abstraction called *entry,* which is a thread queue
  - an explicit task/thread *rendezvous* facility

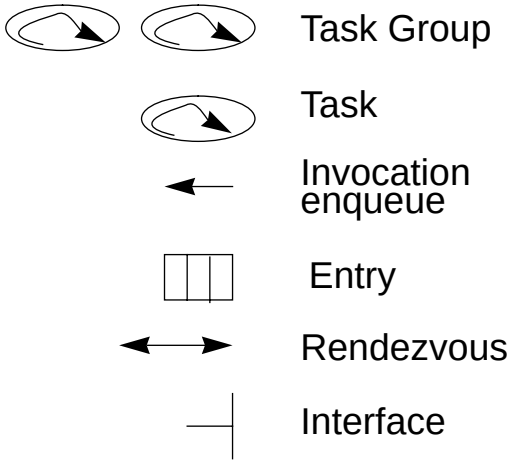
Tasking: Current Engineering Model

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

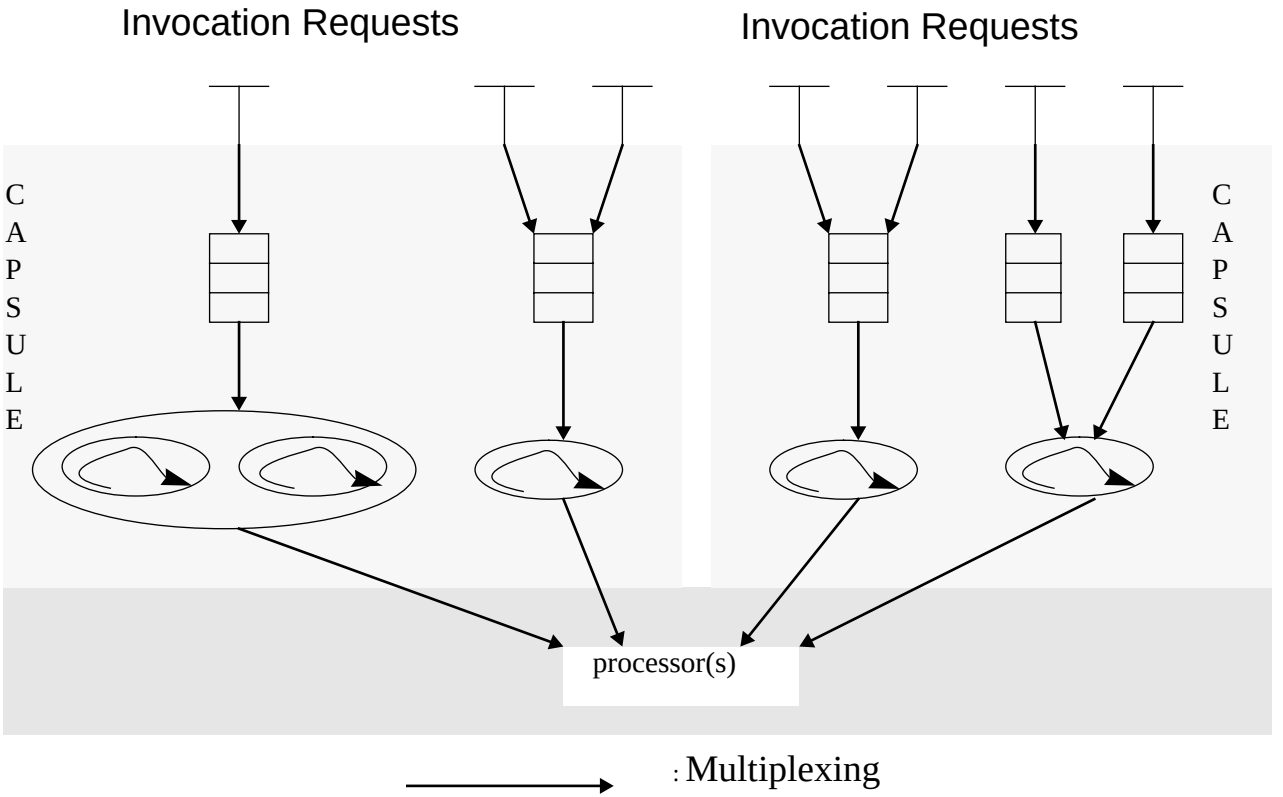


Tasking: New Design





Scheduling: Two Levels



Programming with Priority

- **why programming with priority?**
  - **well understood --- researched for about 20 years**
  - **well supported --- most commercial real-time platforms, e.g. OSF/1, HP/RT, Chorus.**
  - **well accepted --- POSIX 1003.4a, Ada 9X**
  - **widely used**
- **issues**
  - **static and dynamic priorities --- the design of the (task/thread) rendezvous model**
  - **task allocation and task preemption --- vital in a mixed environment to give real-time activities the required response time.**
  - **curb priority inversions --- to bound the computation time of high-priority activities**

Programming with Deadline

- **why deadline?**
  - **intuitive (easy to use) and global**
  - **associated with activities**
- **issues**
  - **deadline as priority in scheduling (doesn't mean guarantee) --- e.g., earliest deadline scheduling**
  - **task preemption is not possible if OS has no support**
  - **deadline and RPC protocol (the meaning of deadline) --- requires the cooperation of processor scheduler and the communication scheduler**
  - **deadline guarantee --- requires many of almost impossible assumptions (activity computing time known, message transportation time bounded, a scheduler (one application, or one address space) controls the whole system cpu resource etc.).**
- **other scheduling schemes --- priority and deadline combination**

Real-Time Communication

- **real-time applications require complicated communication support --- both in functionality and performance**

- **three RPC based designs for:**

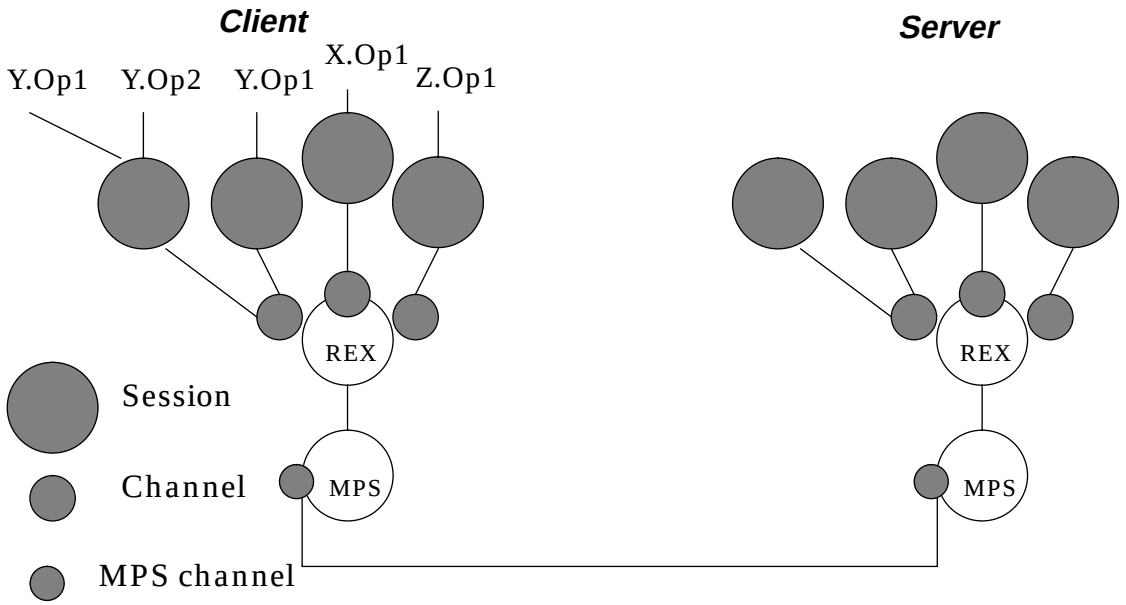
  - **minimize layered multiplexing --- parallel protocol stack to allow the association of communication QoS in the future**

  - **defining the meaning of deadline --- a dependable timed RPC protocol --- for better predictability (important for event synchronization and control)**

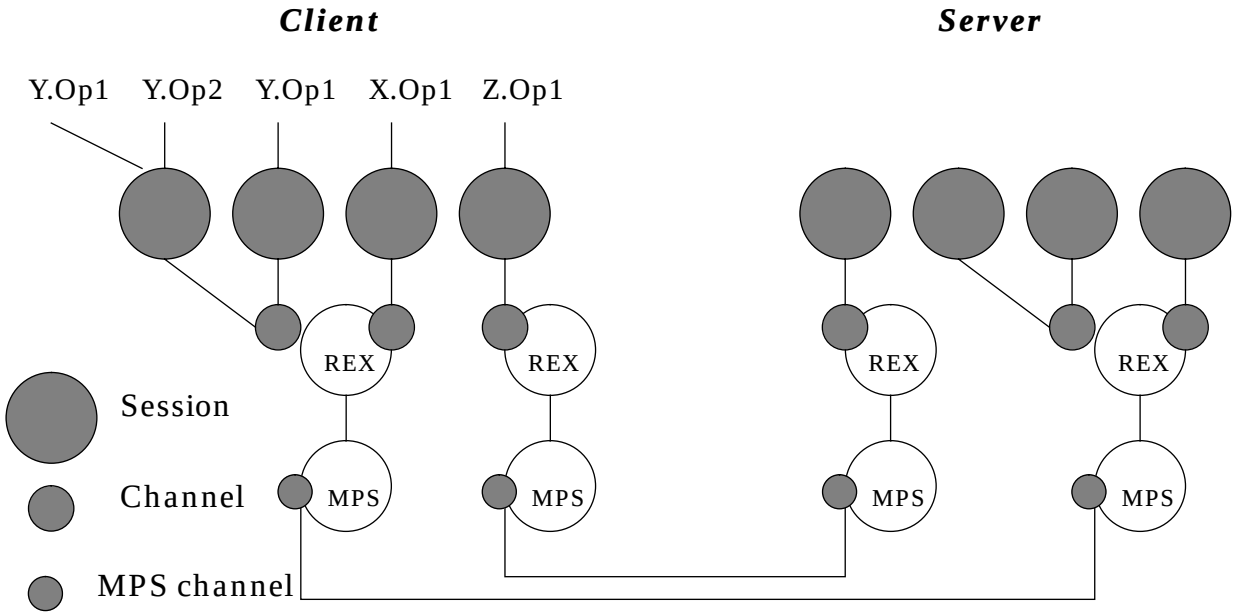
  - **providing an integrated and decomposable RPC protocol stack --- for supporting transportation QoS --- may lead to parametrised transportation protocols for multimedia streams in the future**

Current AW Communication System

- **multiplexing whenever possible --- efficient in resource usage**



New Design: Parallel Protocol Stack



A Timed RPC Protocol

- **associate a deadline with an invocation means:**
  - **the provision of a common sense of time**
  - **the definition of the meaning of the deadline**
  - **the interpretation of the deadline by a communication protocol**
- **the difficulty for a definition --- a deadline has two conflict goals**
  - **goal1: to establish a bound on the time at which the delay in awaiting a reply of an invocation expires**
  - **goal2: to establish a bound on the time at which an invocation is either scheduled to start and/or finish or is *unschedulable* and cancelled, i.e. a *consistent* decision in both a client and its server sites**
- **a design**
  - **using two values, a timeout and a deadline, for separation of concerns**
  - **using different relaxations to allow the programmer to choose different *realistic* combination of goals, i.e. a *typed* deadline**

Remaining Issues

- description of temporal relations
  - the synchronous model
  - the signal extensions
  - a state machine interpreter
- a QoS architecture
- a binding architecture
- a performance transparency architecture
- a stream (or non-operational) interface

Summary

- **architectural**
  - **real-time issues can be addressed in an *OPEN* architecture**
  - **boundaries pose the biggest problem**
- **computational**
  - **the computational model can be extended to cater for real-time programming (no need to redesign from scratch)**
- **engineering**
  - **a real-time open system engineering model is viable**
  - **real-time technologies can be integrated with open architecture**