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

A Real-Time Distributed Processing Infrastructure

Dave Otway

Abstract

There is an increasing need to add real-time capability to distributed systems.

This requires real-time guarantees to be provided to applications in an asynchronous distributed system.

This can be acheived by adding streames, signals, synchronous programming, explicit binding, Quality of Service specifications, resource separation and pre-allocation to the flexible, lightweight distributed processing infrastructure framework described in APM.1580.

APM.1581.01 **Approved** 20th September 1995
Briefing Note

Distribution:
Supersedes:
Superseded by:

A Real-Time Distributed Processing Infrastructure

Dave Otway

Contents

- Requirements
- Approach
- Proof of concepts
- Streams
- Synchronous programming
- Explicit Binding
- Nucleus communications framework
- Multiplexing and concurrency
- Reading List

Requirements

- **Add real-time capabilities to the ANSA/ODP architecture**
  - **preserving its ability to cope with federation, heterogeneity, scaling**
- **Provide interoperability between real-time and non real-time objects and between real-time and non real-time distributed infrastructures**
- **Provide a flexible modular framework that can be used to build optimised infrastructures for specialised applications**
- **Provide real-time guarantees in an asynchronous distributed system**
  - **predictable islands in an unpredictable sea**

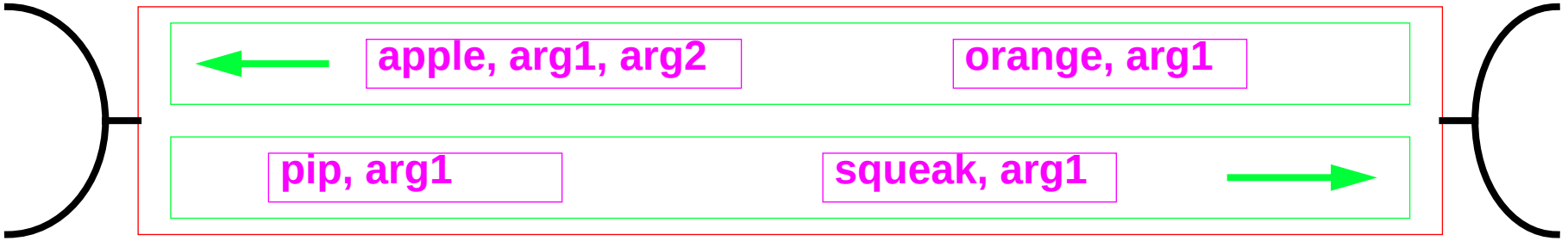
Approach

- interaction models: client/server + streams
- invocation models: call/reply (RPC) + signals
- control model: asynchronous + synchronous programming
- binding model: implicit + explicit
- QoS model: addresses non-functional requirements
- scheduling model: resource separation, pre-allocation, deadlines

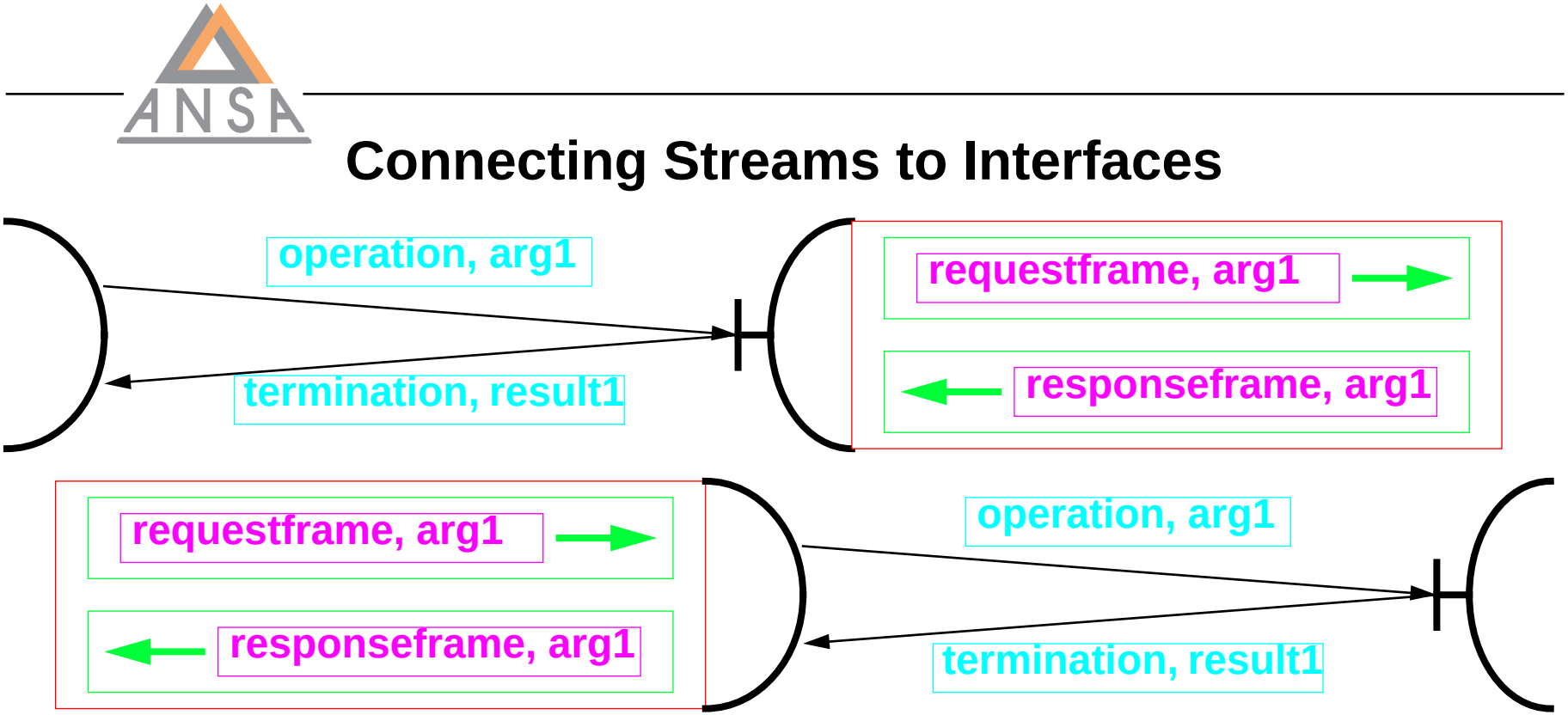
Proof of concepts

- **ANSAware/RT**
  - **add prototype real-time mechanisms to ANSAware 4.1 with Posix Threads**
- **added resource separation, pre-allocation, deadlines, priorities, real-time scheduling, QoS & explicit binding**
- **Performance gain (measured against ANSAware 4.1)**
  - **null RPC 30% faster, 1000 byte RPC 16% faster**
- **Available now over OSF/1, HP/RT, LynxOS**
- **Being ported to Windows/NT**

Streams



- A stream has a set of flows
- A flow has a set of frames and a direction
- A frame has a name and a set of typed arguments
- Streams are typed and can be conformance type checked
- Frames are transmitted by non-blocking writes and read by blocking reads



Connecting Streams to Interfaces

- A stream interface can be programmed to use or provide an operational interface
- But relies on the programmer correctly implementing the operation invocation semantics

Synchronous Programming

- a reactive system continuously interacts with its environment
  - its execution is divided into a sequence of discrete instants
  - each instant reacts to its inputs and produces the corresponding outputs
  - reactions are synchronised with real-time by input time signals
- deterministic behaviour
  - bounded execution paths, calculable in advance
  - with guaranteed pre-scheduled resources:
  - programs have predictable timing and reproducible behaviour
    [even in asynchronous systems]

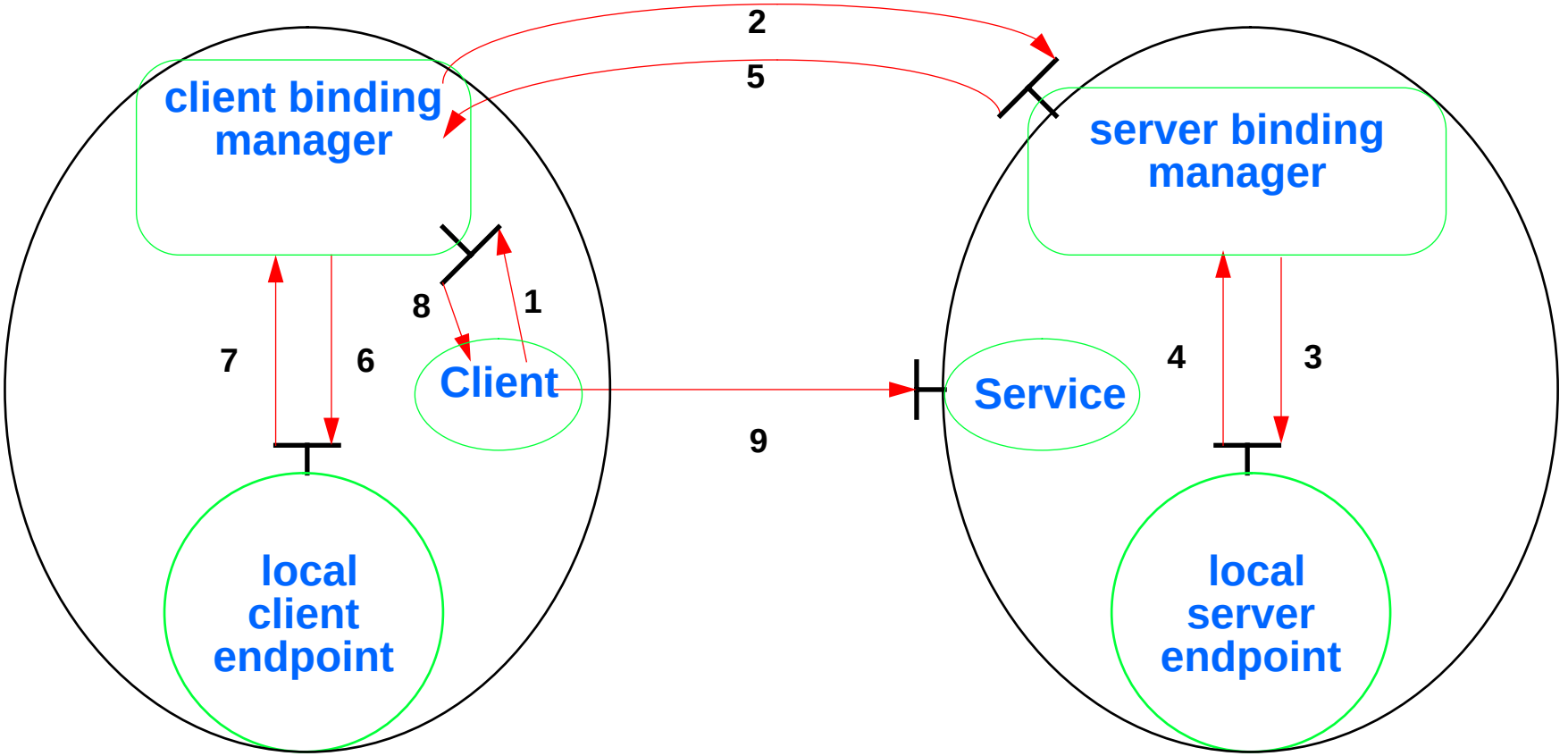
Signals

- each input or output must be a discrete message so the program can determine which instant it must react to it or emit it
- each input or output is defined by a named signal with typed arguments
- each signal has a defined direction (in or out)
- signals are not broadcast, but grouped into interfaces
  - gives same local and remote semantics
  - enables bindings to be scoped
  - enables many instances of a signal interface to be used
- reactions are synchronised with real-time by input time signals

Explicit Binding

- Generate a type specific local binder for each end of a binding
  - for client and server of an operational interface and both ends of a stream
- Perform all binding management by application program (libraries)
  - with the benefit of all distribution transparencies and tools
  - uses an implicit binding
- Each local binder is then requested to construct its end of the binding
- Provide type safe QoS specification, monitoring and control interfaces, multi-channel and multi-party bindings via matching attribute and engineering libraries
- Does not require management protocols for explicit binding to be provided in the communications technology

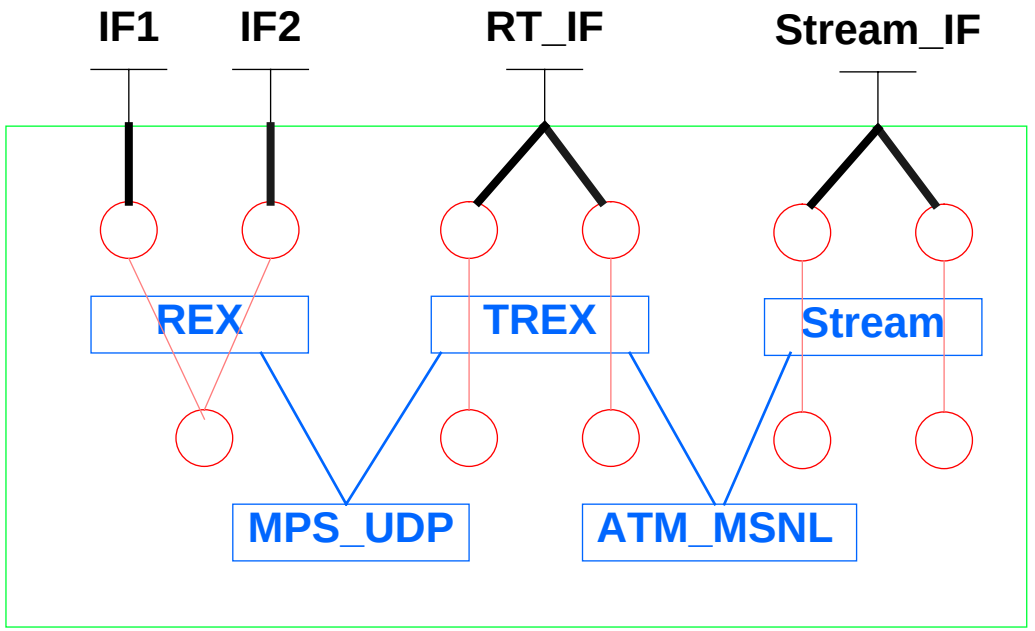
Binding Management

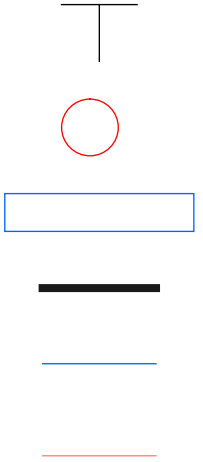


The Nucleus communication framework: concepts

- module - a layer of protocol function
  - (e.g. TCP, IP, UDP, MPS_UDP, REX, GEX)
- protocol - a stack of modules
  - (e.g. TCP/IP, UDP/IP, MPS_UDP/REX)
- channel - an instance of a module
  - (e.g. UDP socket, REX channel)
- binding - associate an interface or stream to a set of channels
- binder - create binding

Examples





Interface

Channel

Module

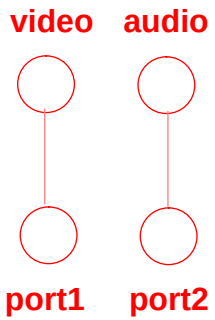
Binding

Module Link (Protocol)

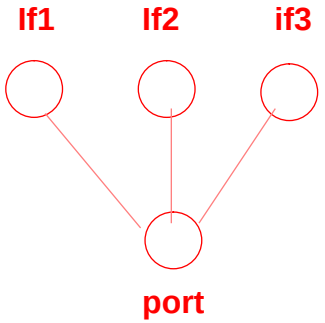
Channel multiplex

Channel Multiplexing

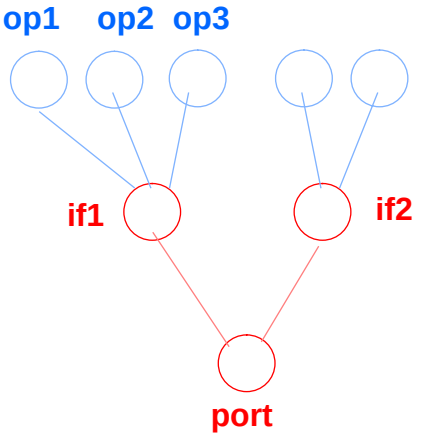
no multiplex



maximum multiplex

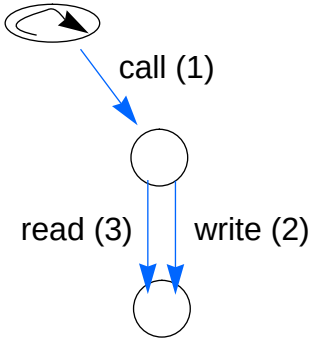


maximum multiplex with concurrency

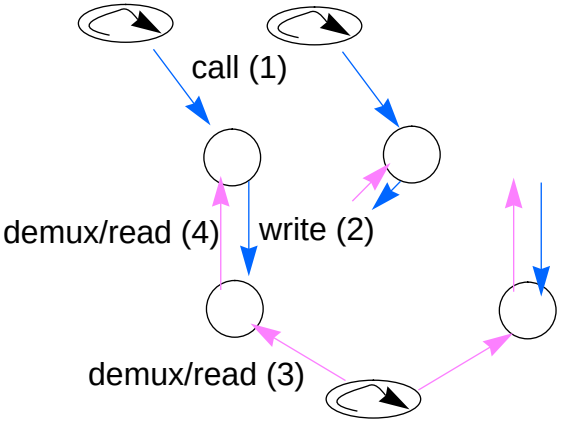


Client concurrency

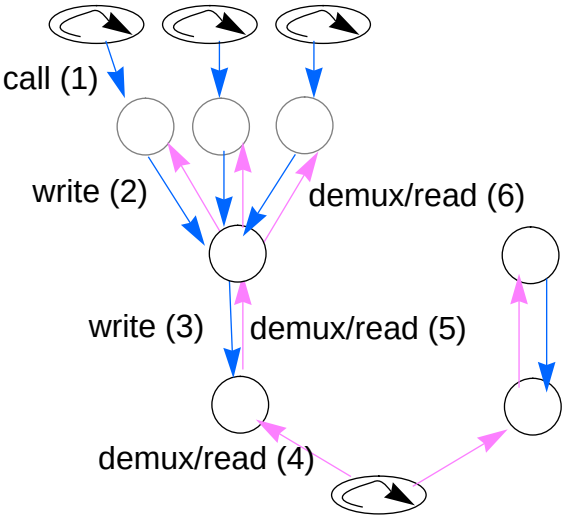
fast/expensive



flexible/slow

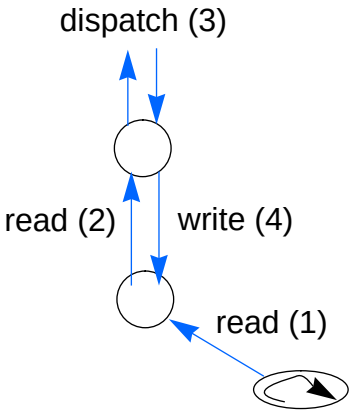


very flexible/slower

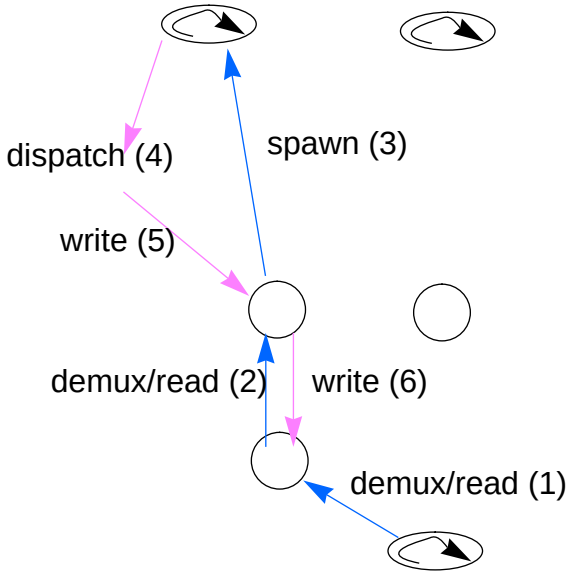


Server concurrency

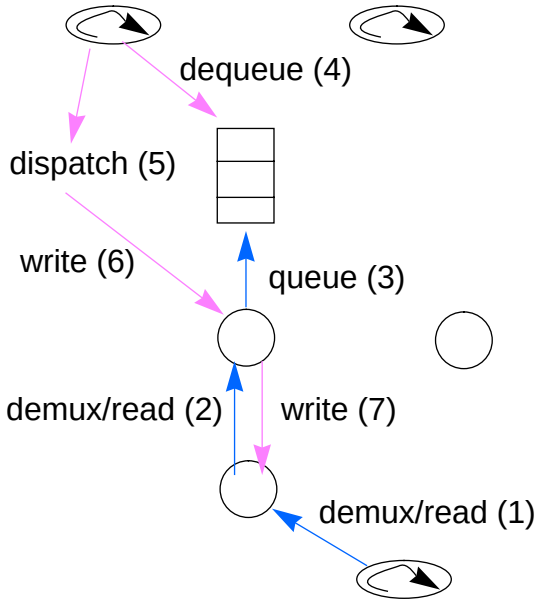
fast/expensive



flexible/slow



very flexible/slower



ANSA Documents

APM.1555: ODP C++ API Design Overview

APM.1554: DIMMA Stub Generator Design and Implementation

APM.1553: An Overview of DIMMA nucleus

APM.1476: ANSAware/RT 1.0 Manual

APM.1393: Streams and Signals

APM.1392: The ANSA Binding Model

APM.1295 An Overview of the Distributed Interactive Multi-Media Architecture