**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 Framework ORB
# (slides for MC 7/95, TC&MC 9/95)

**Guangxing Li**

**Andre Kramer (updated for BT 11.95 visit)**

**Abstract**

It is well understood that real world applications require many ORBs to cover diverse functional and non-functional requirements, and to cope with increasing complexity of technology improvements (such as high performance communication, real-time and multimedia applications). To help manage the complexity and save engineering effort of different ORBs, a framework ORB can be used as a (micro) kernel, which provides a common base for the construction of fully functional, application domain specific ORBs.

The main design challenge of a framework ORB is modularity, which is the key to evolution, flexibility, portability, high performance, configuration and scalability. Typical ORB modular features are generic communication engine, generic threading scheme, flexible buffer management and generic event processing scheme.

The DIMMA nucleus is such a framework ORB, on which real-time and multimedia objects can be readily supported.

APM.1522.03 | **Approved** | 1st November 1995

Briefing Note

**Distribution:**
**Supersedes**:
**Superseded by**:


# A Framework ORB

**Guangxing Li**

**Andre Kramer**

**(ak@ansa.co.uk)**

**documented in APM.1553 :**

**An overview of the DIMMA nucleus**


# Why a framework ORB?

- real world needs many application specific ORBs:
  - high performance, downsize for RT, scalable
  - security, transactional / reliable, real-time / multimedia
  - object services
  - W3, Agents
- technology advances - must be reflected in future ORBs
  - more functional scheduling and advanced memory management support
  - more powerful communication support
- connectivity:
  - many protocols, OSs
  - interoperability with many existing ORBs
  - support alternative object APIs


# Modularity is the Key

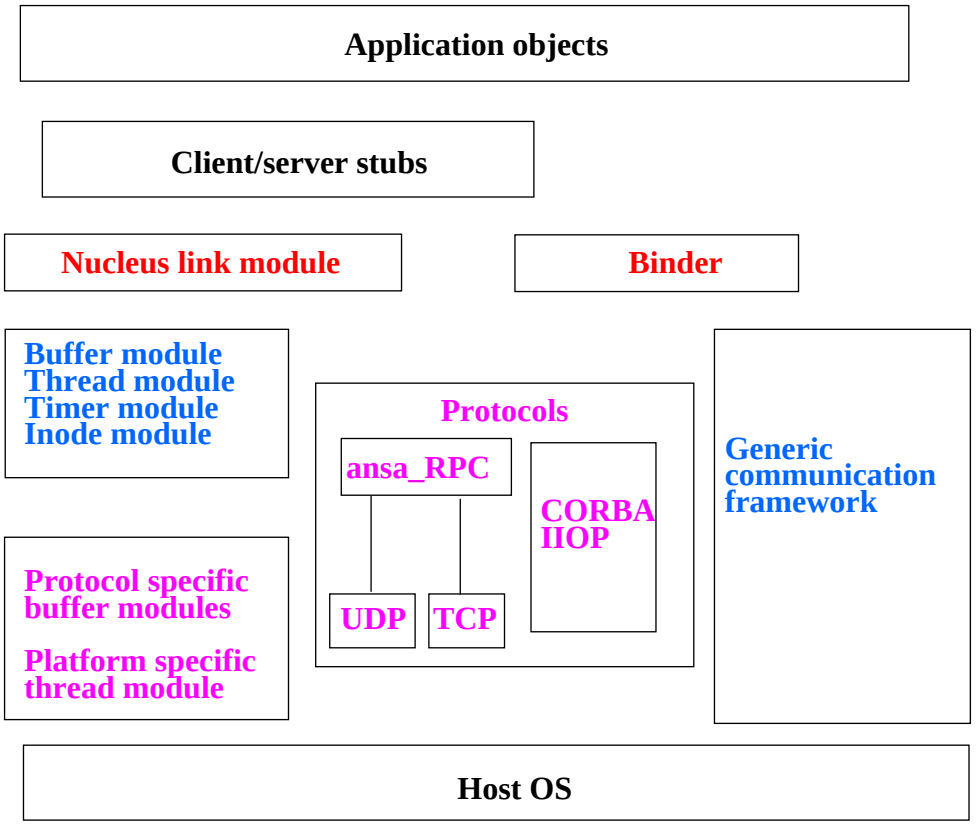
- to future-proof evolution,
- to efficiency and optimization ,
- to portability and flexible configuration,
- to scalability.
- for resource sharing / multiplexing policies,
- for Binding and QoS.


# MicroKernel ORB: the Nucleus

- Apply advanced OS structuring techniques to ORB:
- separate policy from mechanism,
- identify the key abstractions in O-O Framework,
- dynamic configuration,
- QoS guided resource multiplexing.


# Modularity in the Object Nucleus

- Generic communication framework (cf. x-kernel)
- concurrent processing:
  - many threading schemes (POSIX multi-threading,
  - real-time threading), even non-threaded.
- memory management:
  - many buffering schemes (eager, continuous, linked (mbufs, fbufs)),
  - many Protocol Data Unit (PDU) formats,
  - many presentation protocols (ANSA REX, GIOP).
- event and message processing:
  - multiplexing and layering,
  - synchronous or asynchronous or mixed.


Application objects
Client/server stubs
Nucleus link module
Binder
Buffer module
Thread module
Timer module
Inode module
Protocol specific buffer modules
Platform specific thread module
Protocols
ansa_RPC
UDP
TCP
CORBA IIOP
Generic communication framework
Host OS


# Abstractions

- Protocol Module
- Protocol: stack of Modules
- Channel  and Session
- Binding Objects
- QoS Objects
- Protocol independence: generic addressing (IOR Profiles).
- Infrastructure: Threads/Timers,  Buffers,  Inode.

**Figure 0.1: Client channel multiplexing and concurrency (1)**

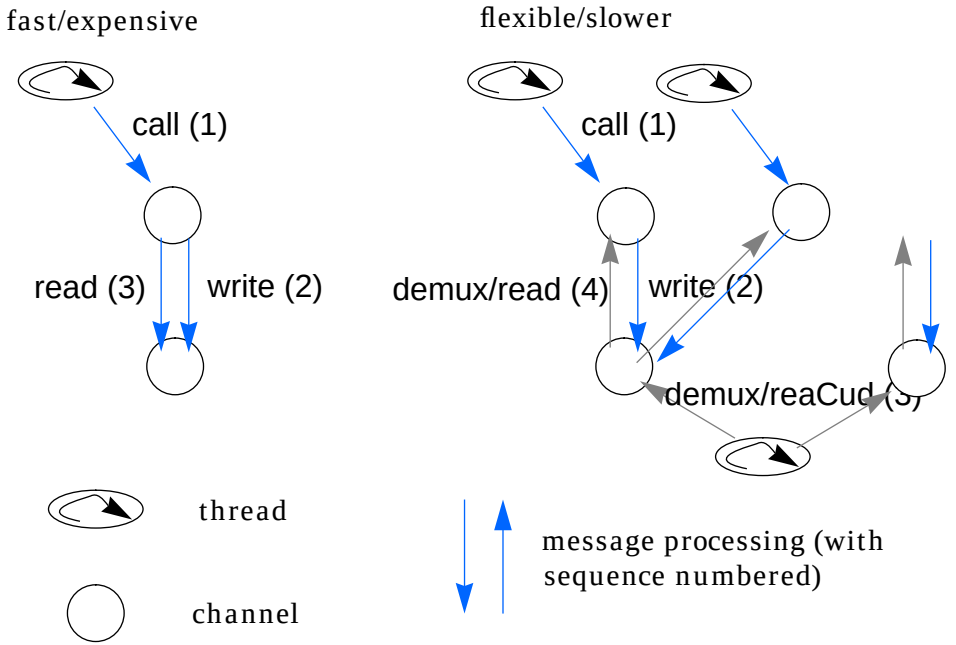


**Figure 0.2: Server channel multiplexing and concurrency (1)**

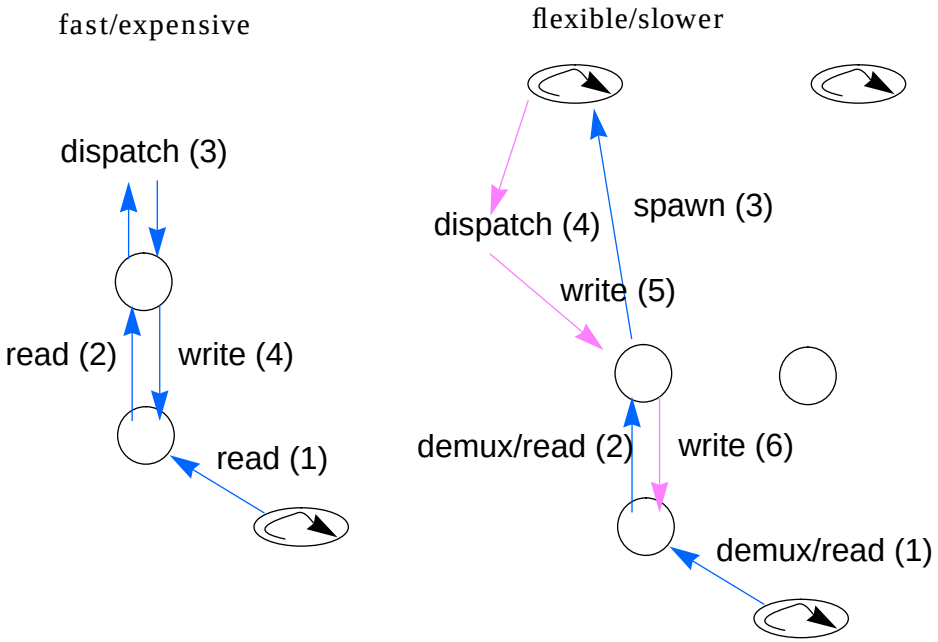


**Figure 0.3: Client channel multiplexing and concurrency (2)**

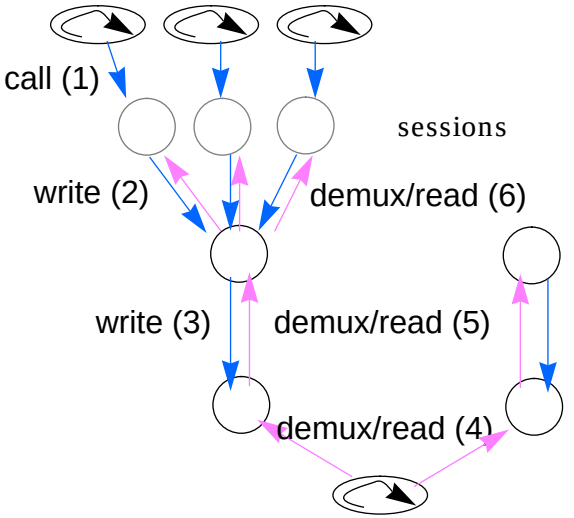
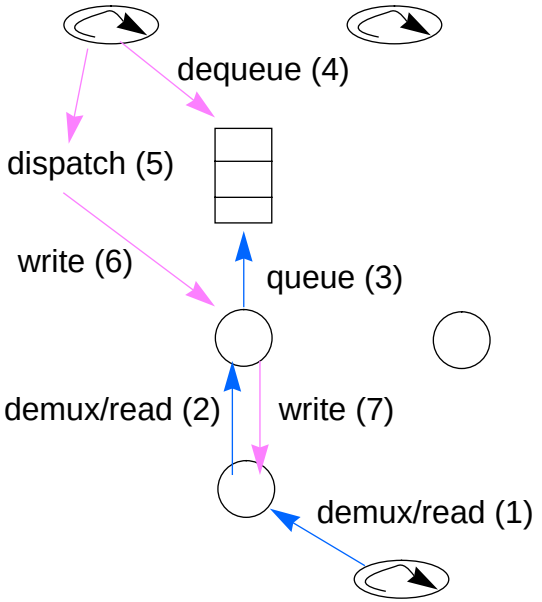


**Figure 0.4: Server channel multiplexing and concurrency (2)**




# Current status (1)

- generic communication framework
  - multiple protocols
  - protocol composition/layering
  - protocol configuration
  - independent of concurrency , arbitrary multiplexing
  - QoS and implicit binding
- a native ansa_RPC protocol - ultra light-weight
  - sessions - allow client concurrency
  - channels - multiplexing with lifetime support
  - automatic server stub dispatch (new thread for server concurrency)
  - driver module for TCP - over cached connections ,
  - UDP (simple to add LIPC).


# Current status (2)

- CORBA IIOP - interoperability
  - also retains most of the above ansa_RPC features.
- truly scalable data structures - from few interfaces to millions
- threading framework:
  - POSIX thread interface (extensible to real-time POSIX thread interface)
  - null thread package
  - timers
- integrated event processing
- an object adapter for the ODP API (developed by Dave Otway)
  - CORBA2 IOR
  - implicit binding


# Future work

- **Services:**
  - CORBA - BOA supporting full CORBA API
  - ANSAware - REX
- **Real-time extensions:**
  - real-time object API
  - QoS API
  - timed RPC, multicast.
- **Multimedia:**
  - Stream API
  - MPEG/JPEG, H.261 over ISDN or MBONE
  - Reservation Protocols
  - ReTINA tORB

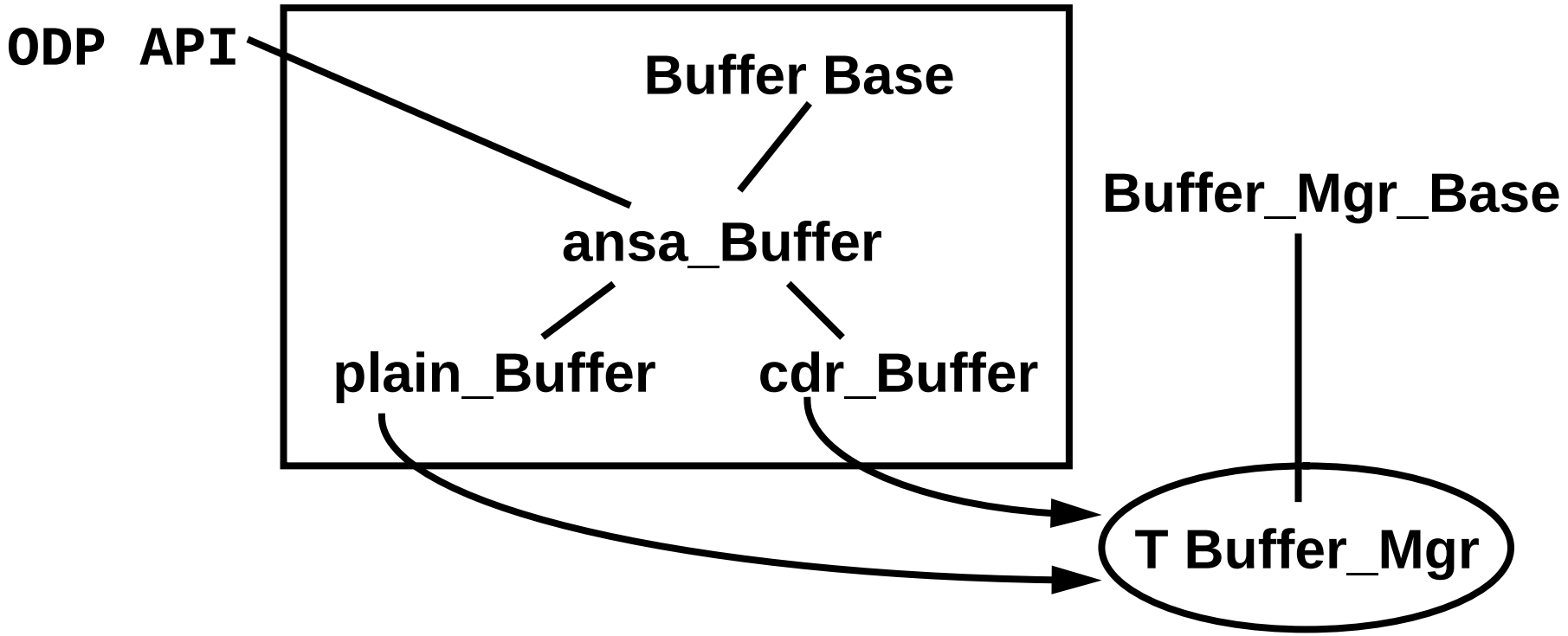



**Fig 1.1 Buffer Classes**