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**

Training

ANSAwise - CORBA Interoperability

Chris Mayers

Abstract

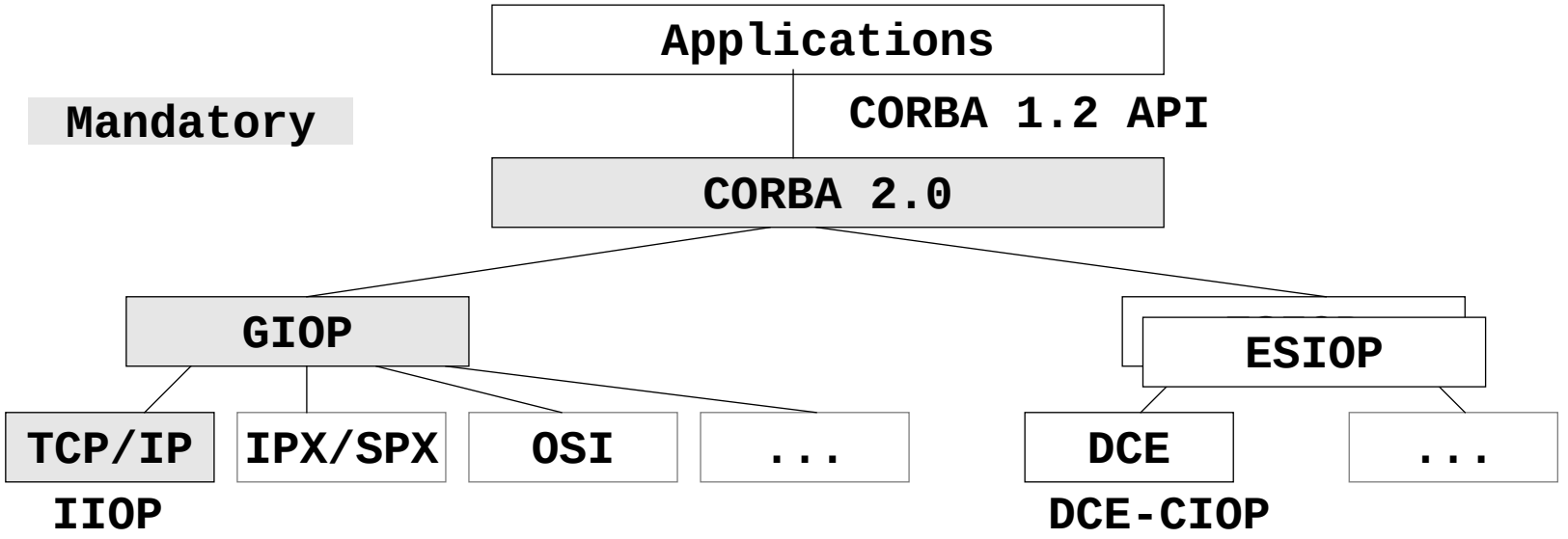
Organizations developing their own infrastructures may wish to use CORBA products, but need to interface to legacy systems using specialist transport protocols. One possibility is to use the CORBA GIOP/ESIOP framework, defining mappings for these transport protocols

This module of the ANSAwise training programme explores the implications of doing this. This module also discusses briefly other interoperability issues, and gives pointers to the ANSA Phase 3 federation work.

APM.1541.01 **Approved** 31st July 1995
Briefing Note

Distribution:
Supersedes:
Superseded by:

CORBA Interoperability



In this session

- ***Examine the implications of ORB Interoperability***
  - **for bridging between ORBs**
- ***Show how the CORBA GIOP/ESIOP framework allows mappings to new protocols***

ORB Implementations

- ***The CORBA Architecture is intended to allow a variety of ORB implementations, for example***
  - **static libraries linked into clients and servers**
  - **dynamic-linked libraries bound to clients and servers at run time**
  - **as a centralized server**
  - **built into the underlying operating system**
- ***The ORB implementations affect the way applications are built and installed...***
  - **... but applications are still source-code portable and interoperable**

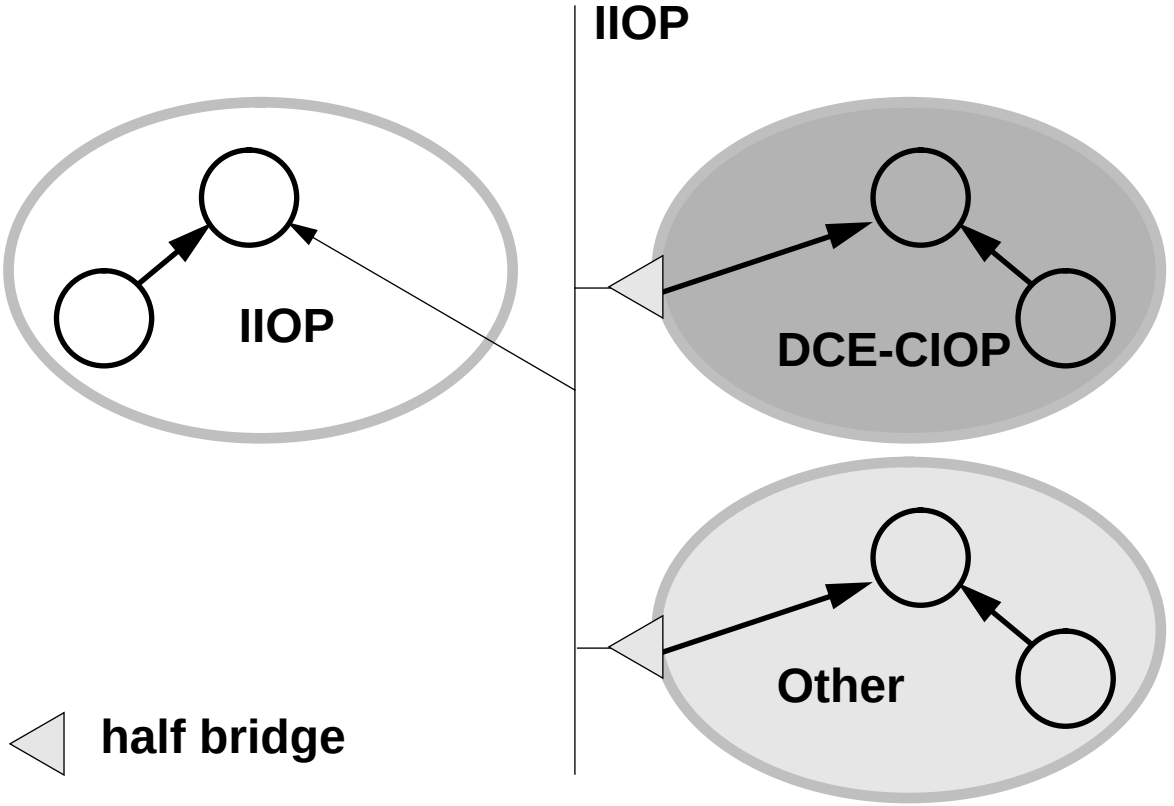
Multiple ORB Implementations

- *An ORB installation may consist of several interoperating ORBs*
  - supplied by different vendors
  - implemented in different ways
- *These ORB implementations may use*
  - different representations for object references
  - different means for performing invocations
- *These differences are transparent to the application*
  - the ORB implementations must be capable of resolving these differences automatically

Implications of ORB Implementations

- *We should consider the distributed system as a set of domains*
  - a domain is a set of objects sharing a common set of characteristics
- *Domains may be*
  - administrative: for resource management and security,...
  - technological: for media, protocols,...
- *Interoperability is concerned with bridging these domain boundaries*
  - transparently to applications
- *Typically, there will be one domain per ORB*
  - but one ORB may support multiple domains, and domains may span ORBs

ORB (Technology) Domains



Domains to consider

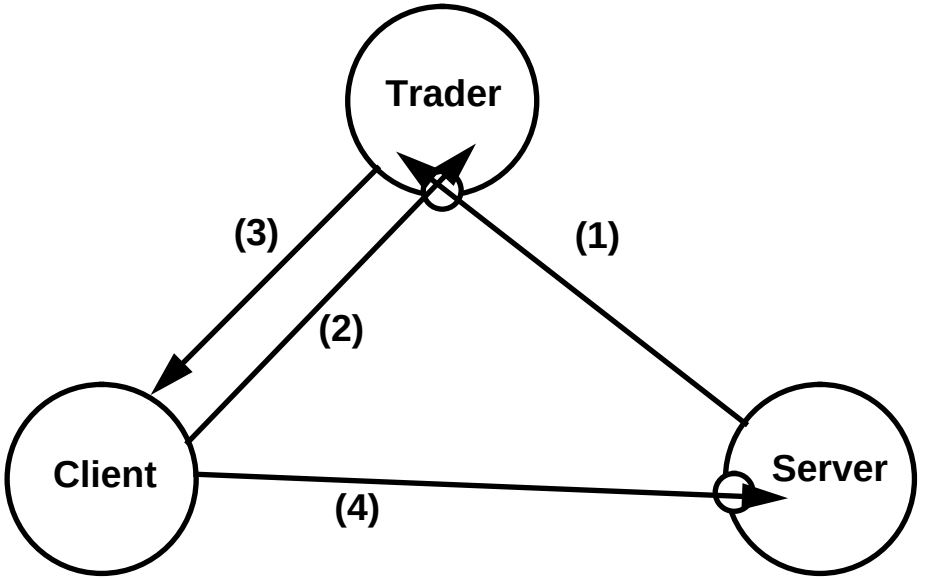
- *referencing domain: the scope of an object reference*
- *representation domain: the scope of a message transfer syntax and protocol*
- *network addressing domain: the scope of a network address*
- *network connectivity domain: the potential scope of a network message*
- *security domain: the extent of a particular security policy*
- *type domain: the scope of particular type identifier*
- *transaction domain: the scope of a given transaction service*

Domain boundaries

- *In general, the boundaries of these different domains will not coincide*
- *Typically, administrative domains are inherently more dynamic than technology domains*
- *Domains may be related by*
  - containment
  - federation

Bridging Must Be Bidirectional

- *Because object references can be transferred anywhere...*



- *... ORB bridges must allow bridging in both directions*

Approaches to interoperability

- ***There two general approaches***
  - ***request level bridging*: built into the application**
  - ***in-line bridging*: built into the ORB**

Request-level bridging

- ***Request-level bridges may be***
  - ***interface-specific*: supporting only some predetermined IDL interfaces**
  - ***generic*: supporting any IDL interface**
- ***Anyone can build a generic request-level bridge***
  - **it does not require access to the internals of an ORB**
- ***The key interfaces are:***
  - **Dynamic Invocation Interface (DII) and Dynamic Skeleton Interface (DSI)**
  - **Interface Repositories and Object Adapters**
  - **Object references themselves**

In-line bridging

- ***Normally, this requires access to the ORB internals***

- ***However, particular ORB vendors may provide limited access***

  - **for example, for integrating a new transport protocol, or a new RPC protocol**

The GIOP Specification

- *It consists of the following elements*

  - Common Data Representation (CDR) definition: a transfer syntax for OMG IDL data types

  - GIOP Message Formats: message exchanges

  - GIOP Transport Assumptions: also covers connection management and message ordering

The IIOP Specification

- *This is a mapping of GIOP onto the TCP/IP transport protocol*
  - it is a specialization of GIOP, not a separate specification
- *ORBs must support IIOP for interoperability*
  - they can use any protocol internally: this may or may not be IIOP

IIOP Implementations

- ***CORBA2/Interoperable vendors may support IIOP either natively or via a half bridge***
- ***If supported via a half bridge***
  - **both the ORB and the half bridge must be commercially available**
  - **the ORB and half bridge may be bundled or sold separately (the half bridge may come from a third party)**
  - **at least one node must run the half bridge**
  - **the specific combination of ORB and half bridge must have its own branding stamp**
- ***Support via a half bridge means that only the customers that need interoperability must pay for it***

Why use anything other than IIOP?

- *To support legacy protocols*
- *Because TCP/IP is not an ideal fit to GIOP*
  - TCP connection overhead may be unacceptable
  - the length of a request may not be known at the point of call

Mapping your own transport protocol within GIOP

- *You must*
  - use the Common Data Representation
  - implement the GIOP Message Formats
  - comply with the GIOP Transport Assumptions

GIOP Transport Assumptions

- ***Connection-orientated transport***
  - **GIOP uses the connections to define the scope and extent of request IDs**
- ***Reliable transport***
  - **ordering, at-most-once delivery, and an acknowledgement**
- ***Byte-stream***
  - **no arbitrary size, fragmentation or alignment limits imposed**
- ***Standard method of initiating a connection***
  - **the listen, connect, accept/reject model of TCP/IP**

GIOP Connection Management

- *Clients generate RequestIds*
- *RequestIDs must unambiguously associate replies with requests within the scope and lifetime of the connection*
- *Connections are not symmetrical*
  - clients can send Request, LocateRequest, Cancel, and MessageError
  - servers can send Reply, LocateReply, CloseConnection, and MessageError
- *Connections are dedicated*
  - only GIOP messages may be sent over GIOP connections

Common Data Representation and Marshalling

- *The CDR is only a transfer syntax*
  - it does not say what techniques should be used to marshal each IDL type
- *One approach is to marshal simple IDL types directly, and to use an interpreter to marshall compound types*
  - the SunSoft IIOP implementation uses an IDL TypeCode interpreter
- *ANSA recommends that you always use one level of procedural indirection, even for simple IDL types*

ESIOPs (Environment-Specific Inter-ORB Protocols)

- *Currently, the DCE-CIOP is the only ESIOP*
- *The DCE-CIOP uses*
  - the DCE RPC packet formats
  - the CORBA CDR representation (*not* the DCE NDR representation)
- *Thus, it treats DCE RPC as a pipe*
  - DCE has such a mechanism already; but not all implementations support it...
  - ...so DCE-CIOP can use conformant arrays instead

Object Location

- *A transport address does not necessarily correspond to the location of any ORB component*
- *Instead, it implies the existence of an 'agent' to which a connection can be made*
  - but the agent may not be able to be handle invocations directly...
  - ...it may need to forward them
- *Both the GIOP and the DCE-IIOP specify LocateRequest messages that allow the 'agent' to respond with the forwarding address*

Multithreading

- ***Multithreaded GIOP/ESIOP implementations will require care***
- ***Because the effect multithreading will have on message ordering, interoperability testing will be vital***
- ***You will also need a multithreaded implementation of the transport protocol libraries***

Implicit context

- ***Some Object Services require 'implicit context' to be passed along with a request***
  - **for example, transaction ids for the Transactions service...**
  - **... security context information for the Security service**
- ***The GIOP Message Formats provides for this***
- ***However, the implicit context is not available via the DSI and DII***
  - **because it is not part of the IDL**
- ***Therefore, request-level bridging between administrative domains may be currently impractical***

Portability and transport-level APIs

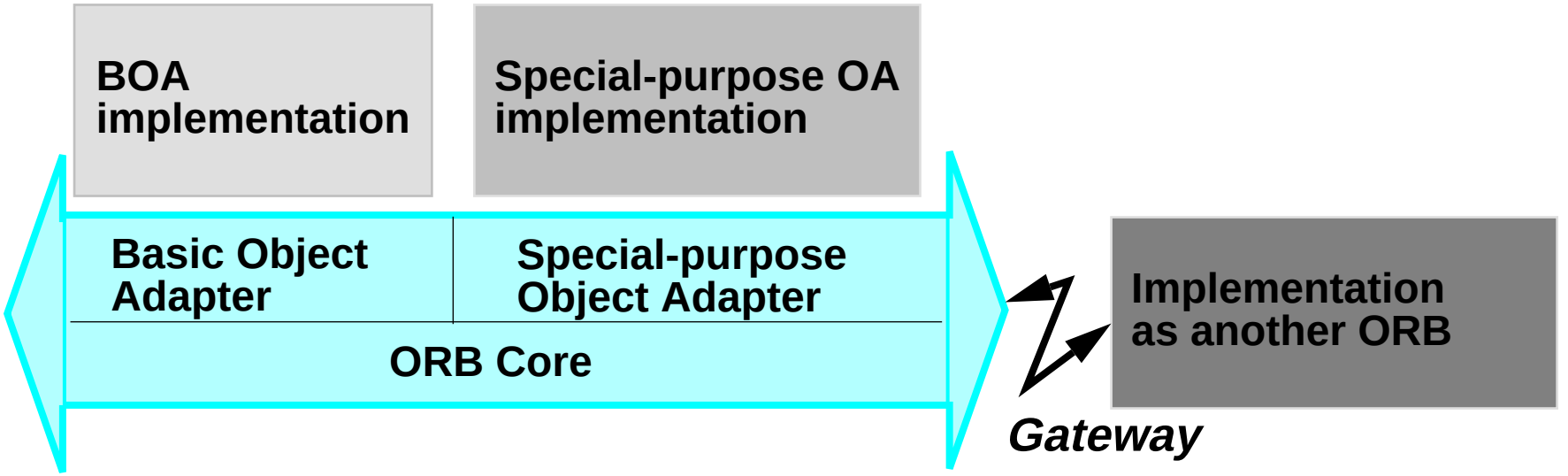
- *The GIOP/ESIOP framework does not specify any transport APIs*
- *For portability of your protocol implementation, using a portable transport API is desirable*
- *Candidates are*
  - XTI (X/Open Transport Interface)
  - Windows Sockets (version 2 of the specification includes QoS support and multiple simultaneous transports)

A suggested approach

- ***Consider carefully your interoperability requirements***
  - **is support for multiple protocols sufficient?**
- ***Select your vendor carefully***
  - **does the ORB have an open infrastructure to enable you to add in-line bridging?**
- ***Select your API for portability carefully***
- ***If you are defining your own protocol, model it on the GIOP or DCE-CIOP***
  - **using CDR**
  - **using similar messages**
  - **with provision for object location, if you need it**

ORB Integration with other object systems

- *As well as wrapping non-OO applications, an ORB can integrate with non-CORBA object systems*
- *This can be done in these ways*



Summary

- ***For more on this topic***
  - **see *CORBA2/Interoperability* (OMG document 95-3-10)**
  - **for a wider view of the interoperability issue, see *ORB Interoperability* (ANSA TR.043)**