26

Inter-operability and distributed application platform design

Yigal Hoffner

The Advanced Networked Systems Architecture Project (ANSA)

APM Ltd., Poseidon House, Castle Park, Cambridge UK, CB3 0RD. Tel: +44 1223 56 8920. Email: yh@ansa.co.uk
Web: <URL:http://www.ansa.co.uk/>

Abstract

Interception is the process which creates and inserts the appropriate gateways when a binding between a client and a server is established across domain boundaries. The inserted gateways can perform the required transformations in the case of technical differences, the checking and vetting in cases where administrative boundaries are necessary, and the monitoring where auditing is required.

The paper introduce a model of interception and then shows how it can be used to explain the implementation alternatives which face system and application integrators who are concerned with inter-operability between different distributed platforms. The paper then looks in detail at one application of interception, namely that of passing interface references through domain boundaries. In particular the structure of interface references required to facilitate the different implementations, and the role of binding in this process are explained. The implications of implementing the different approaches on the design of distributed platforms are then discussed.

This paper provides a model which explains the issues discussed in the OMG CORBA Universal Networked Object (UNO) proposal [UNO 95], as well as discussing some of the issues which are not tackled by the proposal.

Keywords

Distributed application platforms, Inter-operability, Interception, Interface references, IOR (Inter-operable Object References), Binding.

1 INTRODUCTION

Computing facilities in offices, departments, organizations and multinational companies are all being connected together. This coupled with the diversity and proliferation of information systems, and the need to rapidly adjust to business changes, make it necessary to dynamically:

- ***facilitate*** the interaction between different systems where this becomes desirable
- ***restrict*** or prevent interaction between them where this is or becomes undesirable
- ***audit*** the interactions between different systems.

A. Schill et al. (eds.), *Distributed Platforms*

Technical boundaries are caused by differences among distributed application development platforms such as CORBA [OMG 92], DCE [OSF 92] and ANSAware [ARM 93]. Such differences have to be overcome where interaction between them is desirable.

Administrative boundaries demarcate differences between the authorities in charge of systems, their policies and management procedures. These boundaries do not necessarily coincide with technical boundaries and must therefore be erected where necessary. Administrative boundaries also facilitate monitoring for auditing, billing and accounting purposes.

Technical boundaries can be bridged by agreement on common protocols or by the use of gateways (also called transformers, bridges, wrappers or fire-walls) which perform the required transformations.

Standardisation initiatives such as the CORBA Universal Network Object (UNO) [UNO 95] are a step in the right direction. They help overcome some of the technical problems by agreeing on common protocols. In spite of such standardization initiatives, gateways will still be necessary:

- to help erect and maintain administrative boundaries
- to cater for systems which do not support common protocols such as CORBA UNO, either for legacy or other technical/political reasons
- to adjust or modify the interface of an application
- to deal with special cases (niche markets, for example, are likely to have requirements not covered by standards).

Interception is the process which creates and inserts the appropriate gateways when a binding between a client and a server is created across domain boundaries. The inserted gateways can perform the required transformations in the case of technical differences, the checking and vetting in cases where administrative boundaries are necessary, and the monitoring where auditing is required.

The creation, insertion, maintenance and destruction of gateways in a dynamic fashion involves many complex issues of resource allocation and reclamation, quality of service, security, auditing, and domain management.

A model of the process of interception is a useful tool for explaining the issues and outlining the options which will be available to system and application integrators. The options differ in terms of the point in time and the manner in which gateways are created, in the resource allocated to them and in the quality of service guarantees which can be made about the bindings between clients and servers going through the created gateways.

The following chapters introduce such a model and then show how it can be used to map the implementation alternatives. The paper then looks in detail at one application of interception, namely that of passing interface references through domain boundaries. In particular the structure of interface references required to facilitate the different strategies, and the role of binding in this process are explained. The implications of implementing the different strategies on the design of distributed platforms are then discussed.

2 CREATING BINDINGS

2.1 Creating bindings between objects in different domains

The starting point to describing interception is the ***trading process***. The trading process facilitates the transfer of information about services to allow bindings between clients and servers to be set up dynamically [Deschrevel 93], [Hoffner 94]. One essential piece of information which must be passed to the client is the server's ***interface reference*** which contains the information necessary for the client to bind to the server. The passing of the interface reference is part of the trading process, either in explicit

form (through a trader) or implicitly (third party trading where interface references are passed as invocation parameters directly between clients and servers).

Figure 2.1 shows a situation where a client and a server reside in two different domains A and B. Each domain has its own trader and these are connected through gateways which deal with the differences between the two domains (The federation agreement between Domains A and B has led to the establishment of trader to trader gateways embodying the policies for sharing the services across the boundaries).

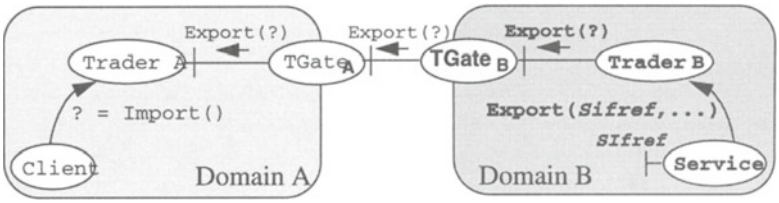

Figure 2.1 Passing interface references (Trading) across domain boundaries

The server in domain B exports its interface reference to its local trader (Trader B) which in turn exports it to the trader in domain A. As a consequence of trading for a service, a gateway for that service must be established as in Figure 2.2. This is a computational view of interception and gateways - issues of how, where and when to implement the gateways are described in the engineering model.

Figure 2.2 Binding across domain boundaries

2.2 A computational model of interception

To achieve the configuration shown in Figure 2.2, both gateways ($TGate_A$ and $TGate_B$) between the traders must be able to detect the passing of the interface reference and act upon it so that the appropriate gateways between the client and server will be set up.

It is worth noting that this would usually be made transparent to application programmers: interception may not appear in the computational model presented to them at all. Rather, this is a computational description of how the programmer would have to deal with it, if it were not provided transparently.

The ***interception process*** involves the detection of the transfer of interface references and the insertion of the necessary gateways capable of carrying out the required transformations, in the invocation path of the potential link before or when it is actually used. Figure 2.3 shows the relationship between the trader gateways and the gateways between the client and server.

The process of interception may be repeated if any of the parameters of the client-server invocations themselves contain an interface reference (third party trading). The new gateways may thus be required to carry out the interception process with regard to the invocation passing through it. Thus gateways may create gateways which in turn create more gateways. The parent-child lineal nature of the process can be seen as gateway cloning (Figure 2.4). Lineal describes a relation among a series of causes or arguments such that the sequence does not come back to the starting point [Bateson 79]. The opposite of lineal is recursive.

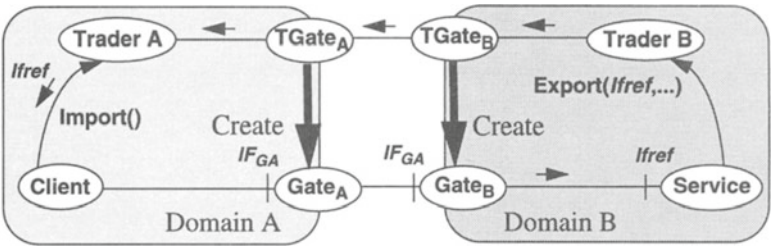

Figure 2.3 Setting up the gateways when interface references pass between different domains

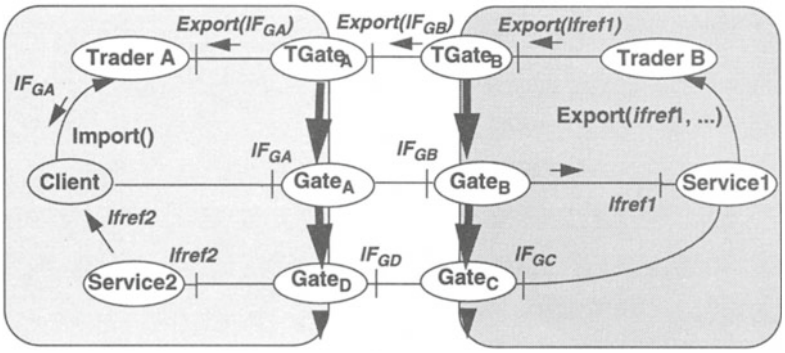

Figure 2.4 Computational model of interception: the cloning nature of setting up the gateways

Note that the computational model of interception is a ***model*** which can be implemented in a variety of different ways. This will be discussed in detail in the rest of this paper.

3 INTERCEPTION IMPLEMENTATION ISSUES

3.1 The engineering model of interception and gateways

Taken at face value, the computational model of interception shown in Figure 2.4 implies that a private gateway is created for each interface reference which is passed through bindings across each domain boundary. This may of course be unacceptable in engineering terms because of inefficient resource usage. The computational gateway, generated by a parent gateway, can have different manifestations, depending on the engineering decisions taken when implementing the interception process [Crawford 95]. Gateways can range from heavy-weight implementations such as a process (capsule in ANSAware) per client-server binding, to a single capsule shared among multiple gateways of different interface types. These are examples of implementing gateways as separate capsules to the clients and servers; it is also possible to implement a gateway inside the same client or server capsule. An example of such implementation are stubs as implemented in ANSAware [ARM 93]. This would give a lighter weight implementation at the expense of lost run-time flexibility.

A ***gateway*** can be viewed as holding information about:

- what transformations have to be applied at the gateway
- how to bind to the server (e.g. hold its interface reference) or the actual bindings.

A gateway can thus be implemented in a variety of ways and the implementation options can be derived from the different possible answers to the following questions:

What resources are allocated, when, for how long and how are they allocated?

In practice, there is a spectrum of implementation options:

1. **how** are resource allocated - per client-server binding
2. **what** resources are allocated per binding and what resources are shared - in terms of capsules, objects, interfaces, table entries
3. **when** are resources allocated - early or late allocation:
   - immediate allocation of resources as interface references cross a domain boundary
   - deferral of allocation of resources to a later point when some object in recipient domain
     - — wishes to use the service
     - — on first invocation
     - — on every invocation, i.e. deferred to each invocation. Resources allocated at invocation are discarded at end of invocation
4. **where** are the resources allocated and kept: the information about the server interface can be kept either in the form of a binding in the gateway or carried by the interface reference. This determines whether resources are allocated in the gateway to maintain the binding, i.e. whether the gateway holds state or not.

Intermediate flavours between the extreme options described above exist.

3.2 Choosing implementation options

Which of the implementation options described in this chapter are most appropriate in any system will be determined by enterprise issues (such as security considerations or real-time constraints) and by the facilities which are available in terms of software and hardware. The spectrum of options outlined in subsequent sections offer design and implementation trade-off which will be determined by the following issues:

- ***QoS*** guarantees and performance: early versus late allocation of resources may effect the QoS guarantees which can be given, particularly availability and performance
- ***Management*** of domain boundaries: in some cases it may be important to know which bindings were created as a result of which client-server interactions. In other words, the lineal relationship or parenthood of gateways may be important for distinguishing between them for management, security or billing purposes. In such cases, it may be necessary to make distinctions between client-server bindings and their gateways on the basis of issues other than the IDL definition of the interface to be supported
- ***Technical*** issues: for example, the availability of interface specific versus type generic stubs can have a significant effect on gateway design and implementation (e.g. generic stub facilities such as CORBA DSI/DII [UNO 95])). Sharing of resources as in the case of type generic gateways may also effect performance and influence the QoS guarantees
- ***Application semantics***: the specific requirements of the application may indicate which option is most appropriate.

3.3 Implementation options

The following sections describe in detail the options outlined above.

3.3.1 Interface type specific versus type generic stubs

Gateways can be implemented as (in order of increasing resource sharing):

- ***private***: gateway for each client-server binding
- ***type specific***: one capsule or object and interface per type of interface
- ***type generic***: one gateway interface capable of supporting any type of service interface.

Type generic stubs allow the use of a single interface to act as a gateway for different types of bindings, thereby circumventing the need to create more gateways as interface references of different service types cross the boundary.

Note that type generic stubs will not necessarily be available in all platforms (although they are part of the CORBA specification in the form of DSI/DII [UNO 95]). Also, in some cases such as processing overload, the overheads of sharing gateway resources will necessitate the creation and use of private gateways.

3.3.2 Resource allocation options

There are different ways in which resources can be allocated to gateways and these describe to what extent are resources shared (order goes from heavy to light-weight use of resources per gateway):

1. each gateway is implemented as a ***capsule*** per client-server binding: binding state is in capsule
2. each gateway is implemented as an ***object*** and ***interface*** per client-server binding: binding state is in object (share a capsule)
3. each gateway is implemented as an ***interface*** per client-server binding: binding state is in interface state (share an object)
4. each gateway is implemented as (binding) state in a ***table***: one interface/object/capsule for all bindings. This requires the client to have some information for indexing the table uniquely (share an interface). The important thing is that there is a single interface for all bindings
5. each gateway is implemented as (binding) state in ***invocation message*** together with a gateway which can use this information to direct the invocation to the appropriate server: one interface/object/capsule for all bindings. State denoting client-server binding is distributed with the server in the ifref and passed in the RPC header.

3.3.3 When are resources allocated to gateways

The allocation of resources can happen at different points:

- immediate resolution strategy: immediate allocation of resources to create gateway when interface reference crosses domain boundary
- deferred resolution strategy: deferring gateway creation to
  - — any object which recognizes the deferment of the resolution and knows how to extract the information necessary to request the resolution
  - — when the client wishes to use the server or on first invocation
- leave-and-forward strategy: may result in no permanent allocation of resources for a client-server binding if used on a per invocation basis.

3.3.4 Maintaining binding state

When the information about the client is carried by each invocation, it is possible to create a binding between the client and server in the forwarding gateway and maintain it over long sessions, or it can be

discarded at the end of each invocation and re-created on each invocation. This provides a choice of allocating the resources for the binding and the overhead of discarding and re-creating it.

The rest of this paper is concerned with the different implementation strategies of the computational interception model and the implications of some of the strategies for the design of distributed application development platforms.

4 INTERFACE REFERENCES CROSSING DOMAIN BOUNDARIES

4.1 Crossing domain boundaries

In order to describe what happens when interface references cross domain boundaries, the example of connecting two different distributed application platforms is used. Most existing platforms (DCE [OSF 92], CORBA based [OMG 92], ANSAware [ARM 93] for example), have different RPC's and also different notions of the information and structure of interface references (interface references in ANSAware, Binding handles in DCE and Object pointers in CORBA). Both differences have to be accommodated by the gateways bridging the two domains.

Three strategies are described:

- immediate resolution strategy
- deferred resolution strategy
- leave-and-forward strategy.

In the following figures, interface references are shown in a graphical manner. The exact nature of the structure and information will be discussed in detail in chapter 5.

4.2 Immediate resolution strategy and interface reference passing

The ***immediate resolution strategy*** shown in Figure 4.1 supports the creation of the necessary gateway as the interface reference is passing the boundary.

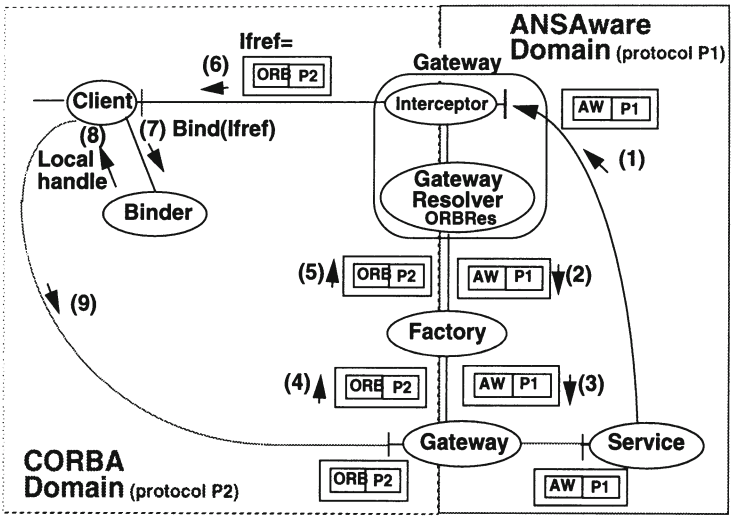

Figure 4.1 The immediate resolution strategy

4.2.1 The immediate resolution strategy sequence

The sequence of operations in the immediate resolution strategy (Figure 4.1) is:

1. the server interface reference is sent from the sender domain to the recipient domain but gets intercepted by the gateway
2. the gateway resolver, according to the immediate resolution strategy, requests a gateway factory to create an appropriate gateway, and passes it the interface reference of the server in the sender domain
3. the factory creates the requested gateway with its interface in the recipient domain. The gateway can bind to the server at this point
4. created gateway passes its interface reference to the factory
5. factory returns the gateway interface reference to the originating gateway resolver and interceptor
6. the new interface reference is passed to the client in the recipient domain
7. client passes the gateway interface reference to the Binder
8. Binder creates and returns a local handle for the client
9. client performs invocation on created gateway.

4.2.2 Advantages and disadvantages of the immediate resolution strategy

The main disadvantage of the immediate resolution strategy lies in the fact that resources are allocated for constructing the gateway regardless of whether the service will be used from the recipient domain.

The main advantage of the strategy is that once the resources are allocated to the gateway, it is possible to guarantee that the gateway will be available when necessary. Also, no additional time has to be spent in setting up the gateway when the binding is created. This may be important in real-time applications.

This strategy has the advantage that it provides a way of dealing with legacy systems as it does not require any changes to be made to the supporting platforms or applications in order to deal with boundary crossing. To the recipient domain, the interface reference will appear as if it is supported by a service in the same domain.

4.2.3 Requirements placed on distributed platform design

None!

4.3 Deferred resolution strategy and interface reference passing

The ***deferred resolution strategy*** (Figure 4.2) intercepts the passing of the interface reference through the domain boundary, marks the interface reference to indicate that it has crossed a domain boundary and that it will ultimately require the creation of the appropriate gateway before being used to invoke the server in the other domain. At some time before the invocation takes place, the marked information will have to be passed to a gateway-resolver capable of creating the gateway.

The request to resolve the deferred interface reference can take place when client requests the Binder to set up the binding or it can take place on first invocation attempt.

4.3.1 The deferred resolution strategy sequence

The sequence of operations in the deferred resolution strategy (Figure 4.2) is:

1. the server interface reference is sent from the sender domain to the recipient domain but gets intercepted by the gateway

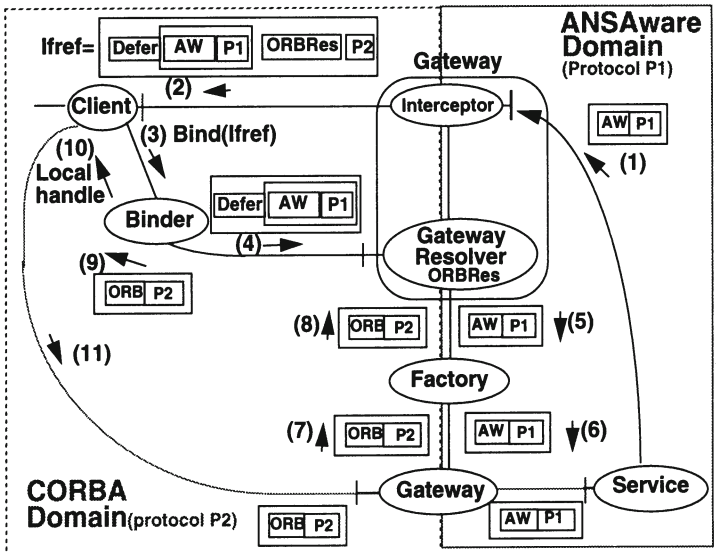

Figure 4.2 The deferred resolution strategy

2. the gateway resolver, according to the deferred resolution strategy, marks the information as deferred ("defer"), adding the reference of the gateway resolver where the deferred interface reference can be resolved when required. The marked interface reference is then passed to the recipient domain
3. client requests Binder to bind to the interface using the passed interface reference
4. Binder recognizes the interface record marked as deferred, extracts the gateway resolver reference and sends the deferred interface record (or the entire interface reference) to the gateway resolver
5. gateway resolver requests a gateway factory to create an appropriate gateway, passing it the interface reference of the server
6. the factory creates the requested gateway with its interface in the recipient domain; the gateway can bind to the server at this point
7. created gateway passes its interface reference to the factory
8. factory returns the gateway interface reference to the gateway resolver
9. the new interface reference is passed to the Binder in the recipient domain
10. Binder creates binding to the gateway and returns a local handle for the client
11. client performs invocation on created gateway.

4.3.2 Advantages and disadvantages of the deferred resolution strategy

The deferred resolution strategy is more efficient with regard to resource utilization than the immediate resolution strategy as the creation of gateways only takes place when a client requests their creation, i.e. when it wishes to use the service.

The main disadvantages of this strategy is that it places overheads on the creation of the client-server binding in terms of the time taken to set up the gateway. Also this strategy requires changes to the supporting distributed platform.

4.3.3 Requirements placed on distributed platform design

This method places the following requirements on distributed platform:

- the interface reference has to be able to hold foreign interface references within it together with gateway-resolver reference
- marking the interface reference to indicate deferred strategy is to be used
- the Binder has to be able to recognize the marking and act on it accordingly.

4.4 Leave-and-forward resolution strategy

When an interface reference is passed through a gateway, it is possible to set up the gateway-resolver as a forwarding agent, mark the passing interface reference to indicate this, and add a reference to the forwarding gateway-resolver. When the recipient object in Domain B wishes to use the interface reference passed to it, it includes the interface reference of the server with the RPC header sent to the gateway-resolver which acts as a forwarding agent. The result is the ***leave-and-forward strategy*** (Figure 4.3).

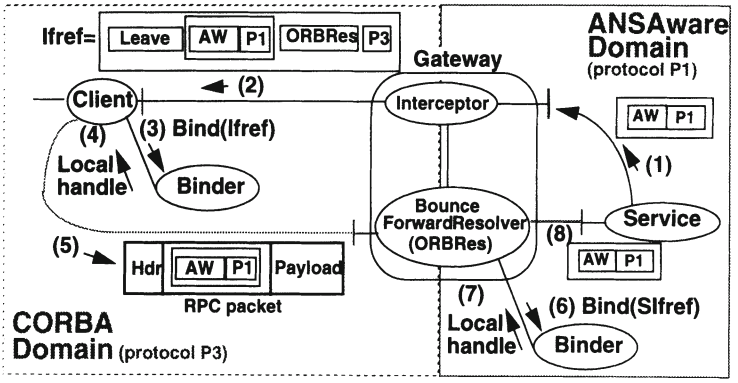

Figure 4.3 The Leave-and-Forward strategy

The allocation of resources in this strategy leaves the following options to the designer:

- a binding between the gateway and server can be established and held over more than one invocation from the client. This implies the gateway-resolver holds client-server specific state
- a binding between the gateway and server can be established for each invocation and discarded after response is delivered to client.

4.4.1 The Leave-and-forward strategy sequence

The sequence of operations in the leave-and-forward strategy (Figure 4.3) is:

1. the server interface reference is sent from the sender domain to the recipient domain but gets intercepted by the gateway
2. the gateway resolver marks the information as "leave", adding the reference of the gateway-resolver where the invocations are to be sent to
3. client passes the Binder the interface reference

4. Binder returns a local handle having set up the appropriate comms
5. client invokes the gateway-resolver passing it the interface reference of the server as part of the RPC header information
6. gateway forwarding resolver requests the Binder in ANSAware Domain to create a binding to the service specified by the interface reference carried in the RPC header
7. a local handle is returned to the gateway forwarding resolver
8. which can then forward the invocation from the client to the server.

4.4.2 Advantages and disadvantages of the leave-and-forward strategy

The main advantage of the leave-and-forward strategy is that it does not require holding any state in the gateway concerning bindings between clients and servers. This will make the re-starting of such a gateway after failure easier as no mappings between clients and servers will be lost.

If the binding between the server and gateway is established and discarded on per invocation basis, the overheads of establishing a binding for each invocation will be prohibitive in some real-time applications.

4.4.3 Requirements placed on distributed platform design

This method places the following requirements on distributed platform:

- the interface reference has to be able to hold foreign interface references within it together with a forwarding gateay reference
- marking the interface reference to indicate leave-and-forward strategy is to be used
- the Binder has to be able to recognize the marking and act on it accordingly
- an RPC protocol supported by the platform must carry the full interface reference of the destination at an agreed part of its header.

5 INTERFACE REFERENCE STRUCTURE AND CONTENT

5.1 Requirements of interface references

From the discussion of the different interception strategies and from considering other inter-operability issues it emerges that interface references should allow (For the sake of simplicity as well as for constraints on space, mechanisms for dealing with re-location, migration, passivation/activation of objects, and which also have an effect on interface reference content and on binding, are not discussed here. A more comprehensive discussion of the issues is provided in [Hoffner 95]):

- sequences of interface records to provide alternative routes: to allow the same service to be accessed from different platforms by different routes
- interface records containing other interface records so as to be able to incorporate foreign interface reference and mark them (as deferred or leave-and-forward, for example)
- language of interface reference to express: Platform name, Deferred ("Defer"), leave-and-forward ("Leave"). This requires domain reserved words which do not have to be global.

The proposed generic structure for interface references which satisfies the above requirements is the following:

- the generic interface reference may have zero or more interface records

- each interface record is marked with the type of the platform on which the service resides and may also include information on the type of record within the platform such as resolver, relocator, deferred, leave-and-forward etc.
- each interface record has information concerning the protocols supported by the service and its platform
- interface records can be nested inside other platform's interface records and marked appropriately
- interface records must also be able to contain information on gateway-resolvers.

The following sections describe the proposed structure in detail.

5.1.1 Alternative paths: interface reference options

An interface reference may have zero or more records providing information about possible paths between the client and server (Figure 5.1).

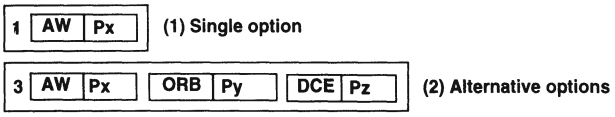

Figure 5.1 Interface Reference structure - one or more altenative paths

The Binder will be able to choose whichever option matches the available comms infrastructure, QoS constraints and cost in terms of resources.

5.1.2 Deferred resolution and Leave-and-forward resolution

The deferred resolution method is used, the information about the cascade can be represented in a nested fashion with the reference to the gateway-resolver appended (Figure 5.2).

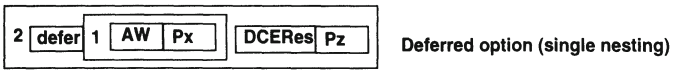

Figure 5.2 Deferred resolution information inside an Interface Reference

- ***deferred*** ("defer"): the information about the gateways can be represented in a nested fashion with the reference to the gateway-resolver appended (Figure 5.2). This tells the Binder that the marked information is not to be interpreted by the Binder but passed to the resolver whose reference is appended to the marked information
- ***"leave"***: the information is marked thereby telling the Binder that it should be used in the RPC header of the invocation sent to the gateway whose reference is appended to the marked information. The gateway in this case could be a type generic gateway implemented so that it receives the destination of the invocation from the RPC header and passes the invocation on.

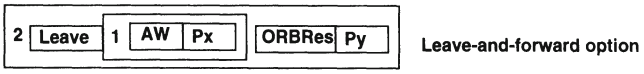

Figure 5.3 Leave-and-forward information inside an Interface Reference

5.1.2.1 Short cuts and optimization

When using the deferred resolution method, a Binder can try and skip gateway nesting of interface references and go directly to the service, provided of course that it has the suitable communication protocols. There are therefore two cases to consider:

- where the nested records are NOT options: Binder must not try to skip a gateway and bind directly to an interface nested inside the structure
- where short cuts can be made: this is particularly important where circularity of reference is to be prevented.

Whether the nesting indicates a compulsory path or not depends on the type of boundary crossed and the enterprise/management issues associated with the crossing of the boundary. Either way it should be left to the gateway-resolver to decide.

The information marked as deferred can also be encrypted so as to prevent its resolution by unauthorized agents. This can help prevent taking any short-cuts where it is necessary to force the use of a gateway.

5.2 OMG CORBA Inter-operability proposal

The UNO (Universal Networked Objects) proposal of the OMG Inter-operability Proposal [UNO 95] includes the IOR (Inter-operable Object References) which defines an inter-operable wrapper of the kind described above.

5.2.1 The IOR (Inter-operable Object References)

The structure of the IOR is shown in Figure 5.4. This is simply a sequence of *TaggedProfiles*, which will

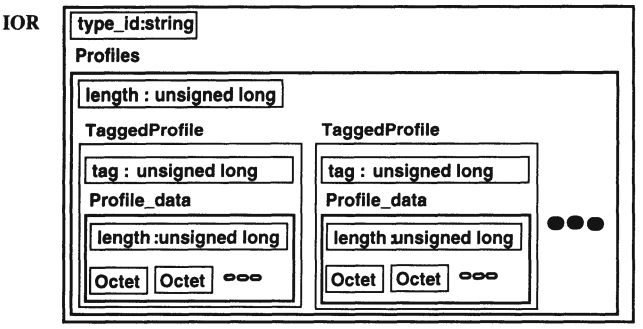

Figure 5.4 Interoperable Object Reference (IOR)

of course include the length of the sequence. The *TaggedProfile* structure leaves the contents of *profile_data* unconstrained. The *profile_data* should hold sufficient information to enable interaction with some object, but this may vary from a complete interface reference to something much more lightweight, e.g. a protocol identifier and an address.

Profiles in a single IOR may hold information describing several interfaces, in arbitrary forms; this allows free interchange of interface references between different distributed application platforms.

6 BINDING

6.1 Binding and interface references

The binding process described in [Otway 95] is extended to deal with the options described in chapters 4 and 5:

- recognize interface records within interface records
- recognize interface records marked as deferred
- recognize interface records marked as leave-and-forward
- extract the information about where to resolve the marked information, i.e. the gateway resolver (or use a default gateway-resolver) in the case of deferred resolution
- extract the information about where to send the forwarded invocation in the case of the leave-and-forward strategy
- allow for other information concerning relocation, passivation/activation, for example.

6.2 Binder algorithm

A Binder dealing with an interface reference will:

- try each interface record:
  - — immediately usable record: set up comms and pass the client a local handle to do invocation on
  - — record marked as deferred: extract gateway-resolver interface reference and pass it the currently processed interface reference. The returned interface reference is treated in the same manner as the current one
  - — record marked as leave-and-forward: extract gateway-resolver interface reference and set up the appropriate comms and RPC with the destination interface reference in the RPC header, and pass the client a local handle on which invocations can be performed.

6.3 Binder policy

As the order of interface records is not necessarily indicative of preferences or priorities from the point of view of the interface reference producer, the Binder's policy has to determine:

- whether to deal with each record at a time or search first for immediately usable references before trying to resolve deferred ones
- whether to look for short cuts in marked interface records.

7 CONCLUSIONS

Different approaches to interception have been outlined in this paper, offering system integrators different options concerning how to connect platforms. The options are concerned with what resources are allocated, when, for how long and how they are allocated. The choice of the appropriate option depends on performance, resource allocation, security and manageability.

The immediate resolution strategy does not require any changes to the supporting platform and will therefore be suitable where such changes are not possible for legacy reasons. The deferred resolution and the leave-and-forward strategies require changes to the interface reference structure and content and subsequently changes to the Binder to enable the platform to support such strategies. Implementing these changes will enable platforms to offer a wide variety of options to system integrators and federated system administrators.

The resources allocated to each client-server binding can vary depending on what is shared between different bindings. For example, using type generic gateways has the advantage of not having to generate type specific gateways as interface references pass through domain boundaries. This can reduce the resources which have to be allocated per client-server binding. On the other hand, in some implementations, type generic gateways may incur some performance overheads.

For a more comprehensive discussion of the topics mentioned in this paper as well as related ones see [Hoffner 95]. For a discussion of implementation issues see [Crawford 95].

8 ACKNOWLEDGEMENTS

The author would like to thank the following people for their input and help in carrying out the work described in this paper: Professor Peter Linington and Chris Scott from the University of Kent at Canterbury, and Nic Holt from ICL (UK) Ltd. at Manchester.

Many thanks also to Nigel Edwards, Mark Masden, Rob Van der Linden and Andrew Herbert from the ANSA core team at APM in Cambridge, UK, who reviewed and commented on the document.

Special thanks are also due to David Iggulden, Dave Otway and Ben Crawford from the ANSA core team for their help and suggestions.

9 REFERENCES

[ARM 93] "The ANSAware 4.1 manual set", Architecture Projects Management, Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1993.

[UNO 95] "CORBA 2.0/Interoperability - Universal Networked Objects", BNR Europe Ltd., Digital Equipment Corporation, Expersoft Corporation, Hewlett Packard Corporation, IBM Corporation, ICL, plc., IONA Technologies, Sunsoft Inc., OMG Document number 95.3.10, March 20, 1995.

[Bateson 79] Bateson, G., "Mind and Nature: a Necessary Unity", Bantam Books, ISBN 0-553-34575-3, 1979.

[Deschrevel 93] Deschrevel, J-P., "The ANSA Model for Trading and Federation", APM.1005, Architecture Projects Management, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1993.

[Crawford 95] Crawford, B., "Gateway Design and Implementation", APM.1303, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[Hoffner 94] Hoffner, Y. "A Designers' Introduction to Trading", APM.1387, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[Hoffner 95] Hoffner, Y. and Crawford, B., "Federation and Interoperability", APM.1514, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[IONA 93] Orbix: Programmer's Guide, Iona Technologies Ltd., Dublin, Republic of Ireland, 1993.

[OMG 92] "The Common Object Request Broker: Architecture and Specification", Document Number 91.12.1, Object Management Group and X/Open, 1992.

[OSF 92] OSF, "DCE Application Development Guide", Open Software Foundation, 11 Cambridge Centre, Cambridge, MA 02142, USA, 1992.

[Otway 95] Otway, D. J., "The ANSA Binding Model", APM.1392, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995