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 - Trading and Federation

Chris Mayers

Abstract

Organizations wish to be able to offer electronic services worldwide.

This creates several technical problems. First, being able to discover what services are available. Second, being able to determine which services match your specification. Third, being able to control which services are advertised to whom, and when.

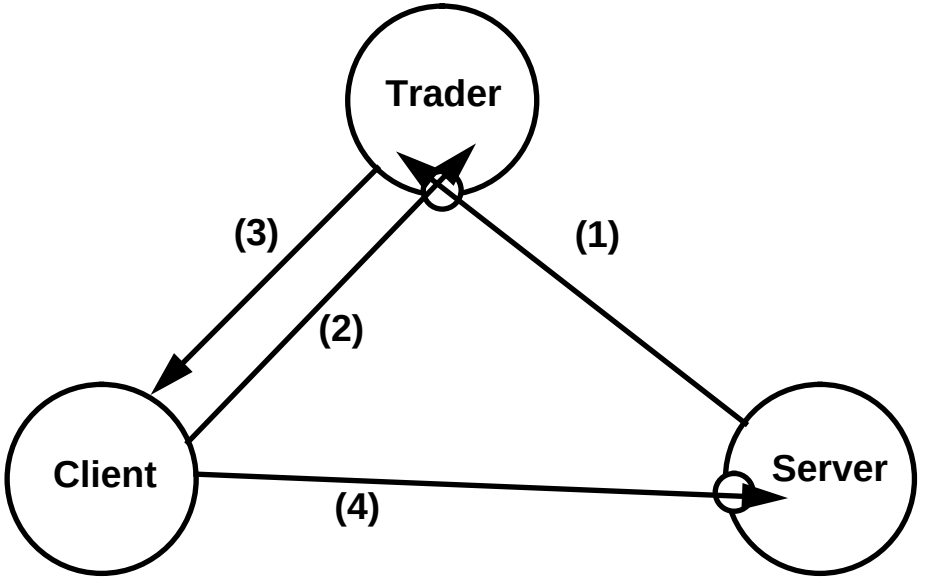
The solution being offered is the use of trading services, which can be federated across administrative boundaries.

[This module of the ANSAwise training programme is a revision of APM.1330. It assumes that the audience is already familiar with Trading concepts. It gives examples of systems which already use Trading, and then focuses on advanced trading concepts.]

APM.1741.01 **Approved** 2nd April 1996
Briefing Note

Distribution:
Supersedes:
Superseded by:

Trading and Federation

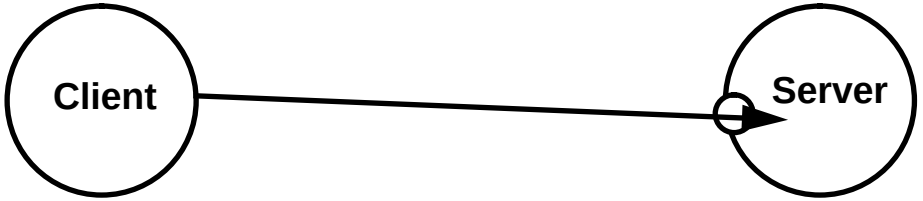


In this session

- **Show a simple form of trading in action**
  - **as applied to large distributed systems**
- **Explain the significance of federation**
- **Explain federated trading**

Client and Server are Roles, not Types of Machine

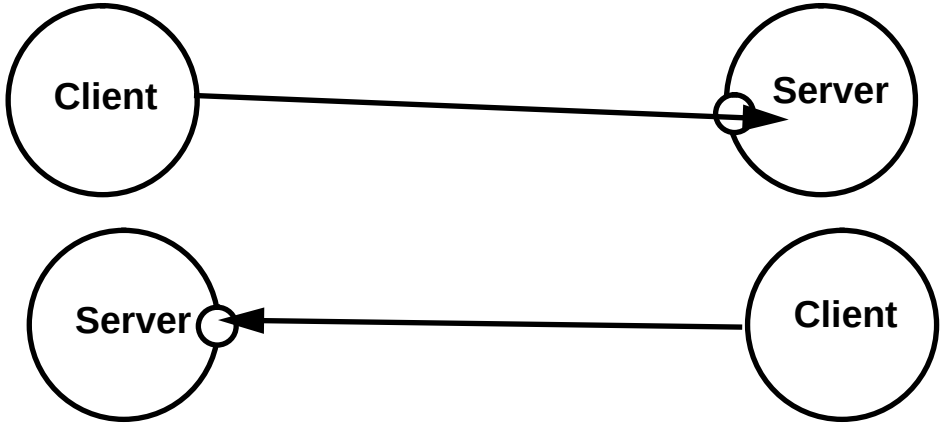
- We use the terms client and server in a particular sense...
- ... Client and Server are roles in a specific interaction between objects

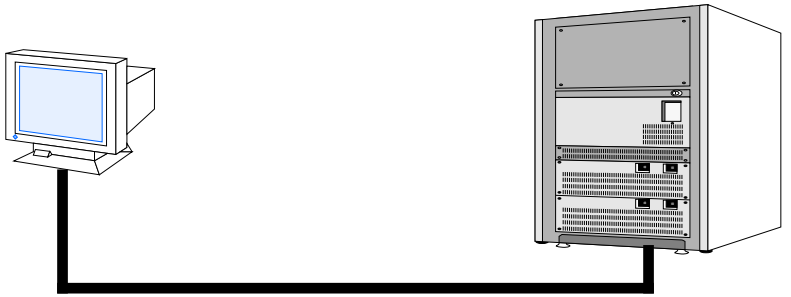


- Nothing is implied about the technology that supports them
  - the client could be a mainframe...
  - ...the server could be a PC
  - ... they could both be on the same Unix machine

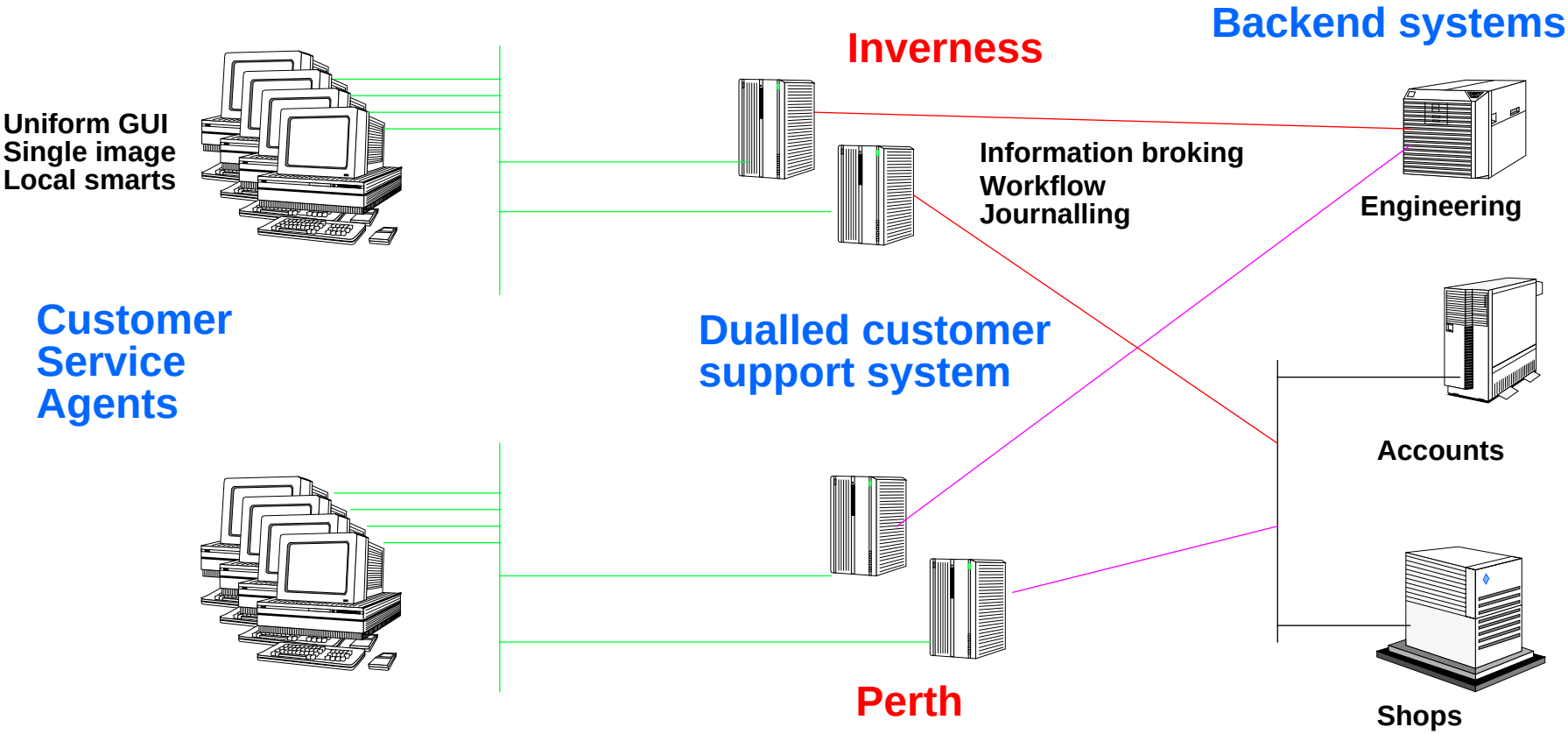
Client and Server Roles

- **The roles are determined by the objects themselves**





Utility Company



OASIS - The Partners

- Scottish Hydro-Electric Customer Information Systems
- ICL
- University of Nottingham
- Gid Ltd.

Objectives

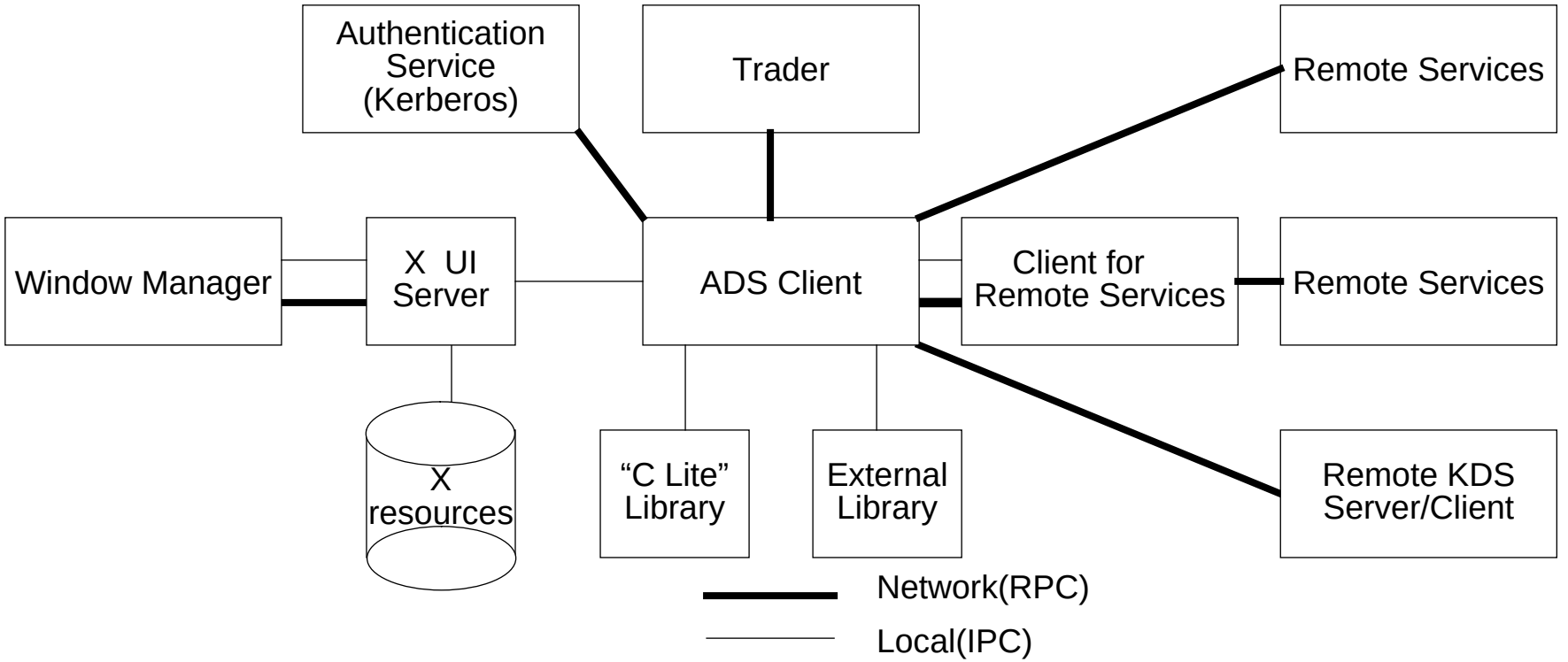
- **Provide a common front-end to customer-based systems**
  - **giving an integrated view of customer information across existing application systems and databases...**
  - **...identify the customer once**
- **Allow merging of data from within a system**
- **Allow merging of data from multiple systems**

Challenges and Successes

- **Reduce time to identify customer**
- **Reduce time to train new users**
- **Cope with change**

...delivered on time, to specification, within budget

NASA Astrophysics Data System (ADS)



Objectives

- **Provide the science investigator**
  - **efficient and broad access to NASA's current and future data holdings**
  - **tools for ease of data interpretability**
- **Provide NASA**
  - **a common information infrastructure for science analysis**
  - **increased scientific return from missions, reducing the duplication of effort**

Challenges and Successes

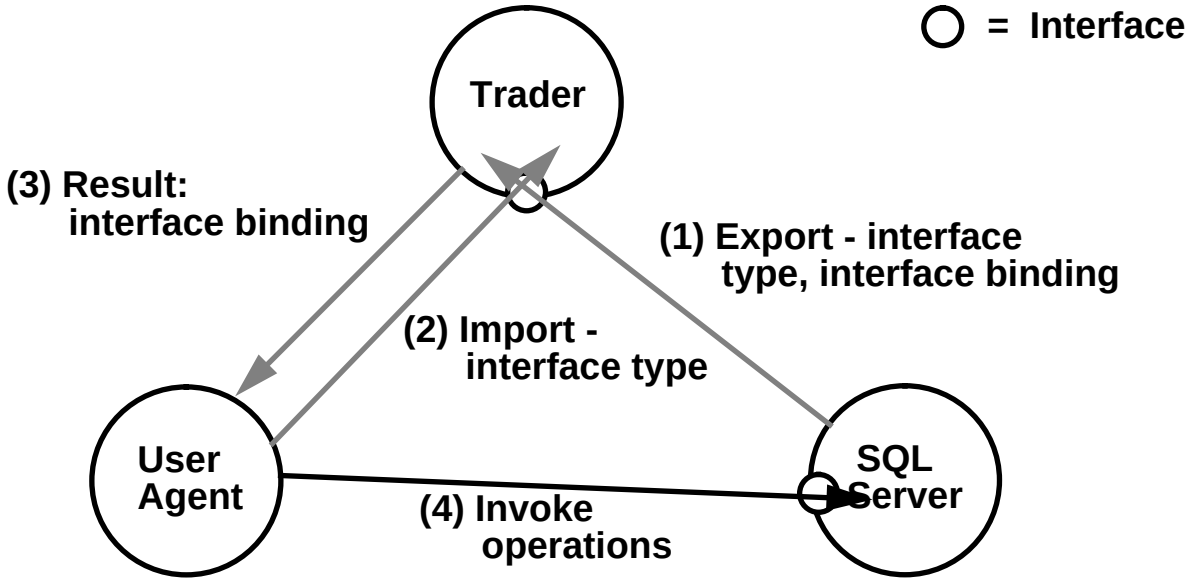
- **Turn the ultimate legacy system into the world's largest open distributed system**
  - **7 large mainframes (IBM 3081, CDC Cyber 180, DEC VAX 8600, Britton-Lee IDM,...)**
  - **15 heterogeneous databases**
  - **terabytes of existing data (text, data, images,...)**
- **Support a variety of client machines, including low-cost PC**
- **Deploy a solution rapidly**

Information retrieval in the NASA ADS

- Database front-ended by the Knowledge Dictionary System (KDS)
  - Factor spaces (initially 30 fields, 600 terms) for retrieval
  - SQL for remote data access
- Trader supports location transparency for databases

Trading in the NASA ADS

- **In the NASA ADS**
  - **the client is the ADS User Agent (KDS)**
  - **the server is the SQL server**



Benefits of a Trader service

- **The main benefit is location transparency**
  - **clients need not know the location of server objects**
  - **new server objects and new clients can be introduced independently**
  - **server objects may later move**

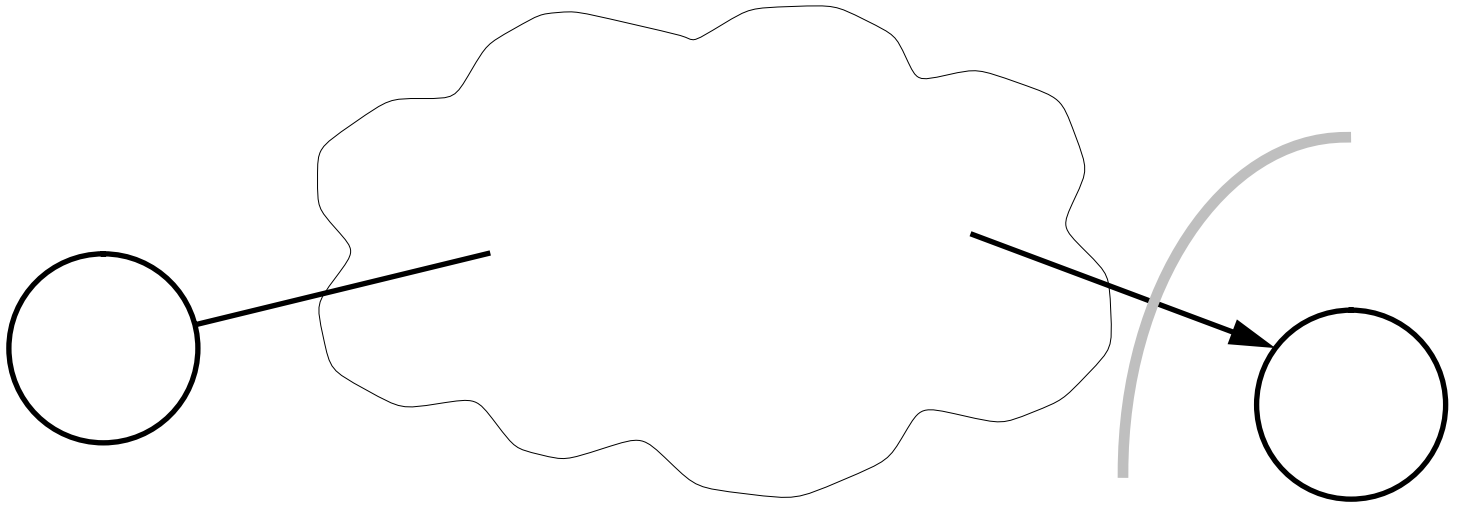
Trading in large distributed systems

- **Because there will be millions of servers in the world:**
  - **there will be many Traders providing the Trading service**
  - **... Traders must be interconnected**
  - **... the Trading service must itself be distributed for scalability**
  - **... and cannot be centralized**
  - **this is called *federated trading***
- **And also because organizations will wish to control their own Traders:**
  - **to determine who sees which services**

The need for federation

- **Organizations want to offer electronic services to other organizations**
  - **services provided by applications in the usual way...**
  - **...services provided by distributed objects**
- **Organizations want to make money**
  - **to *sell* services provided by distributed objects**
- **Organizations need to keep control of their objects**
  - **control over their security, for instance**

Federation is concerned with boundaries



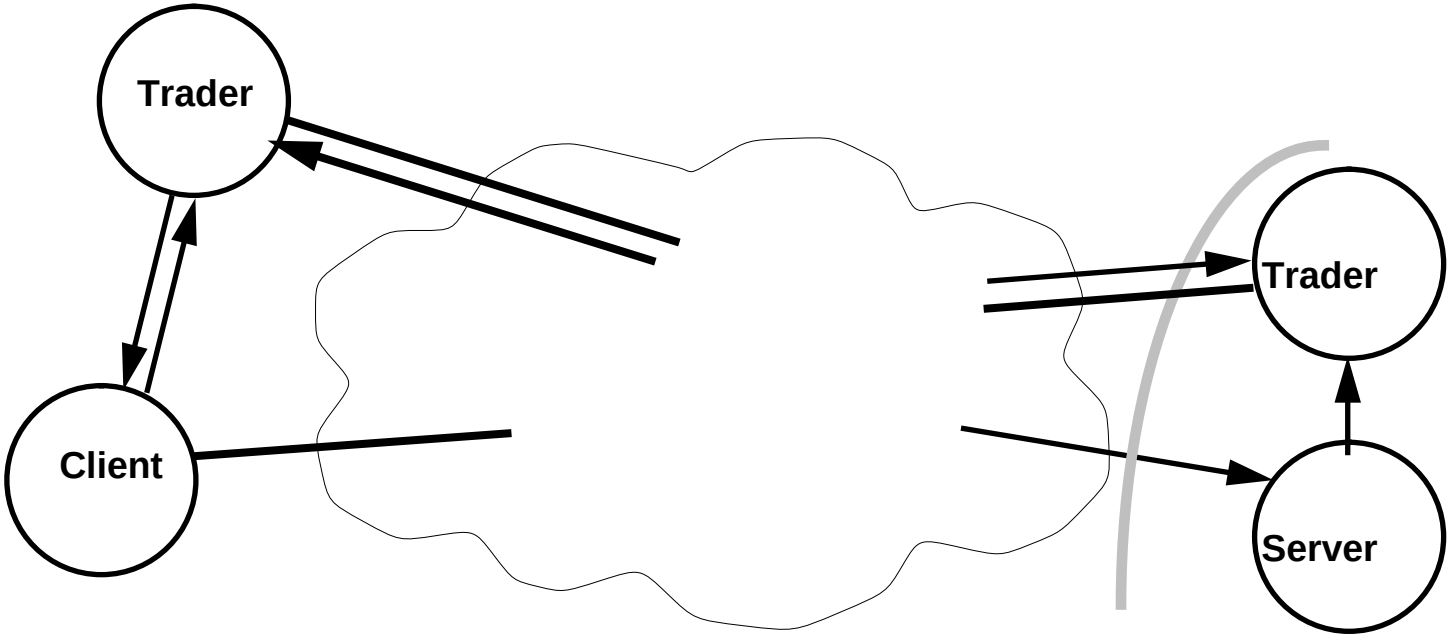
- Controls can be enforced naturally at boundaries
- Distributed objects are ideal
  - their encapsulation provides a boundary

Kinds of boundaries

- There may be many kinds of boundaries
  - Administrative
  - Judicial
  - Political
  - Technological
- Openness entails bridging boundaries (in a controlled manner)
  - not abolishing them
- But we want these boundaries to be transparent to applications

Federated Trading

- **The traders are federated, but the client and server need not be aware of the boundary**



Different policies for different organizations

- Federation raises policy issues including
  - Authority
  - Billing
  - Security
- Organizations that sell electronic services must be able to enforce their policies
  - they cannot rely on trust

Federated Trading

- Distributed objects already satisfy most of the needs of federation...
- ... but there are new issues
  - federation of naming contexts
  - technology boundaries
  - security
- These issues affect all services, including the trading service itself

Finding a starting place

- You use the Trader to find other services
  - but how does the client find the Trader to start with...
  - ... as there may be many Traders?
- This is a bootstrap problem
- You could use the Naming service to find a Trader
  - ...if the Trader uses the Naming service...
  - ...there is then a bootstrap problem with the Naming service!

The CORBA solution for finding a starting place

- In interface ORB is a standard, simplified, local name service

```
module CORBA {
   ...
      interface ORB {
            ...
            typedef string ObjectId;
            typdedef sequence <ObjectId> ObjectIdList;
            exception InvalidName {};
            ObjectIdList list_initial_services ();
            Object resolve_initial_references
                     (in ObjectId identifier)
                raises (InvalidName)
      }
}
```


Obtaining initial object references - a CORBA approach

- Use `list_initial_services` to get the names of initial services
  - currently, CORBA reserves the names `InterfaceRepository` and `NameService`
- From this list, pick out the `NameService`
- Use `resolve_initial_references` to get the object reference of the Naming service
- Use the Naming service to find the Trader service

Advanced trading - negotiation

- **The 4-step approach described earlier will satisfy many needs, but not all...**
  - **...if several offers match, which one should be chosen?**
- **Different offers may have a different quality-of-service, or cost**
  - **This will require more sophisticated patterns of *negotiation*...**
  - **... for trade-offs between quality and cost**
  - **... involving another service, a *negotiation service***
  - **the Trading service is neither accountable nor responsible for the quality of service described in service offers**

Advanced trading - different lifecycle epochs

- The description so far implies that trading happens at run-time
  - this is typical
- Does it make sense for trading to happen at earlier epochs during the service development lifecycle?
  - at application link time?
  - at application compile time?
  - at design time?
  - at specification time?

Advanced trading - compile/link time epoch

- At compile/link time, special tools could be used...
  - to test that a set of services could trade successfully
  - to determine an optimum set of bindings within this set
  - to replace the trading operations by this set of bindings
- ... to optimize a group of closely-related services

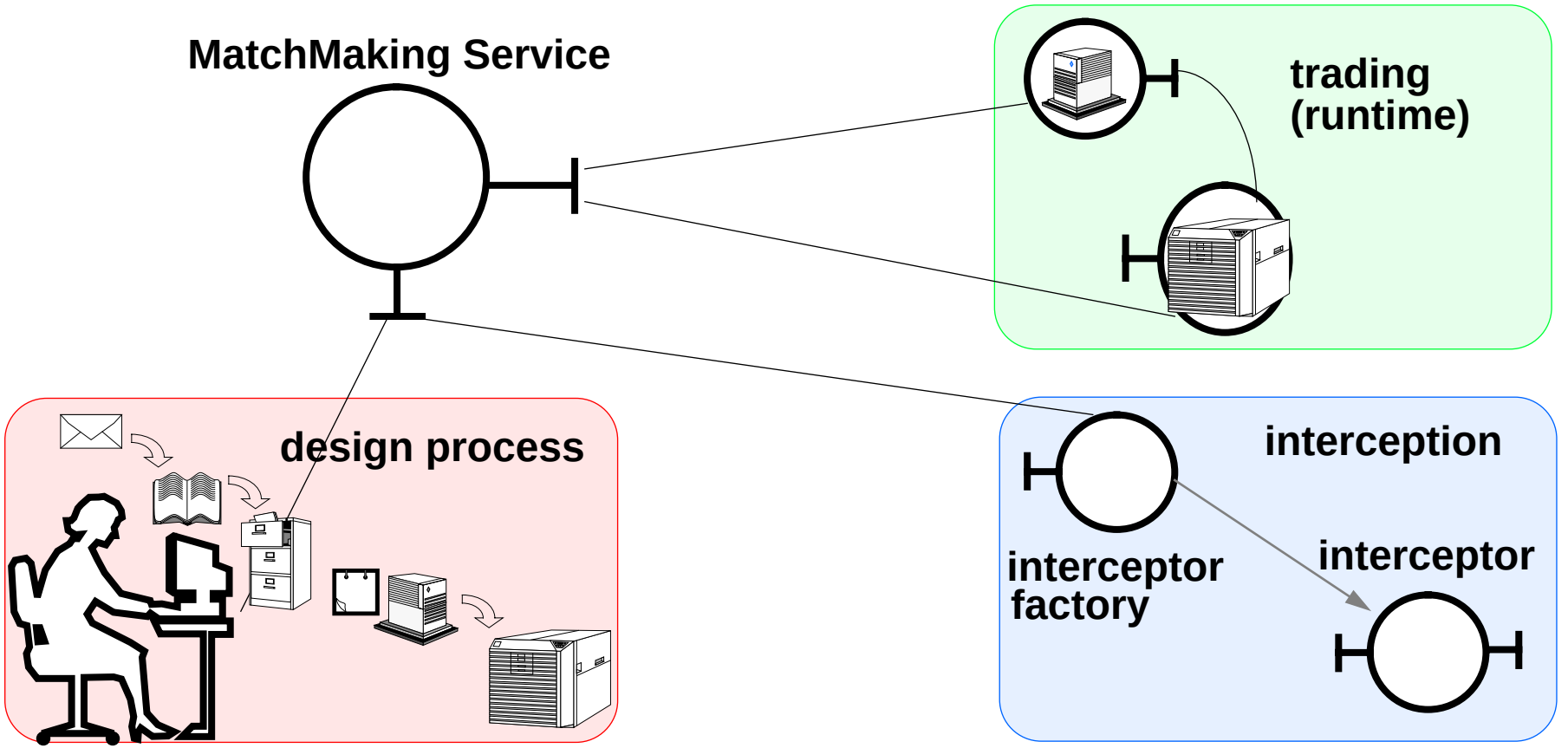
Advanced trading - design/specification epoch

- **Applications and service designers need to know what services already exist**
  - **to avoid re-inventing the wheel**
- **This information is already available in the trading service**
  - **intelligent browsing tools can aid the designer**

Advanced trading - new service wanted...

- Suppose the designer does not find the service they require...
  - ...but they do not want to implement the service themselves
- They can simply place a service request for this non-existent service
  - ...like a Service Wanted advertisement
- A special service can intercept this request
  - and create a dummy service

Use of MatchMaking Service



Summary

- **Trading is necessary to support large distributed systems**
  - **these will use federated traders**
- **Advanced trading features may emerge in due course**

Trading and Federation - more information

- For a general discussion of trading and federation, see *The ANSA Model for Trading and Federation* (AR.005.00)
- For security issues, see *A Framework for Federating Secure Systems* (AR.008.00)
- For a discussion of Quality of Service issues in multimedia applications, see *Integrating Multimedia into the ANSA Architecture* (TR.028.00)