St Andrews University Easter Lecture Course
21 April 1997

Dr Andy Hopper

"Multimedia and Network Computing"

References:

1. "Networked Multimedia: The Medusa Environment" - Tim Glauert, Andy Hopper, Stuart Wray - IEEE Multimedia, Vol. 1, N. 4, Winter 1994

2. "Video Mail Retrieval by Voice: An Overview of the Cambridge/Olivetti Retrieval System" Martin Brown, Jonathan Foote, Gareth Jones, Karen Sparck Jones, Steve Young- Proceedings ACM Multimedia '94 Conference Workshop on Multimedia Database Management Systems, San Francisco CA, USA, October 1994

"Smart Personalisation"

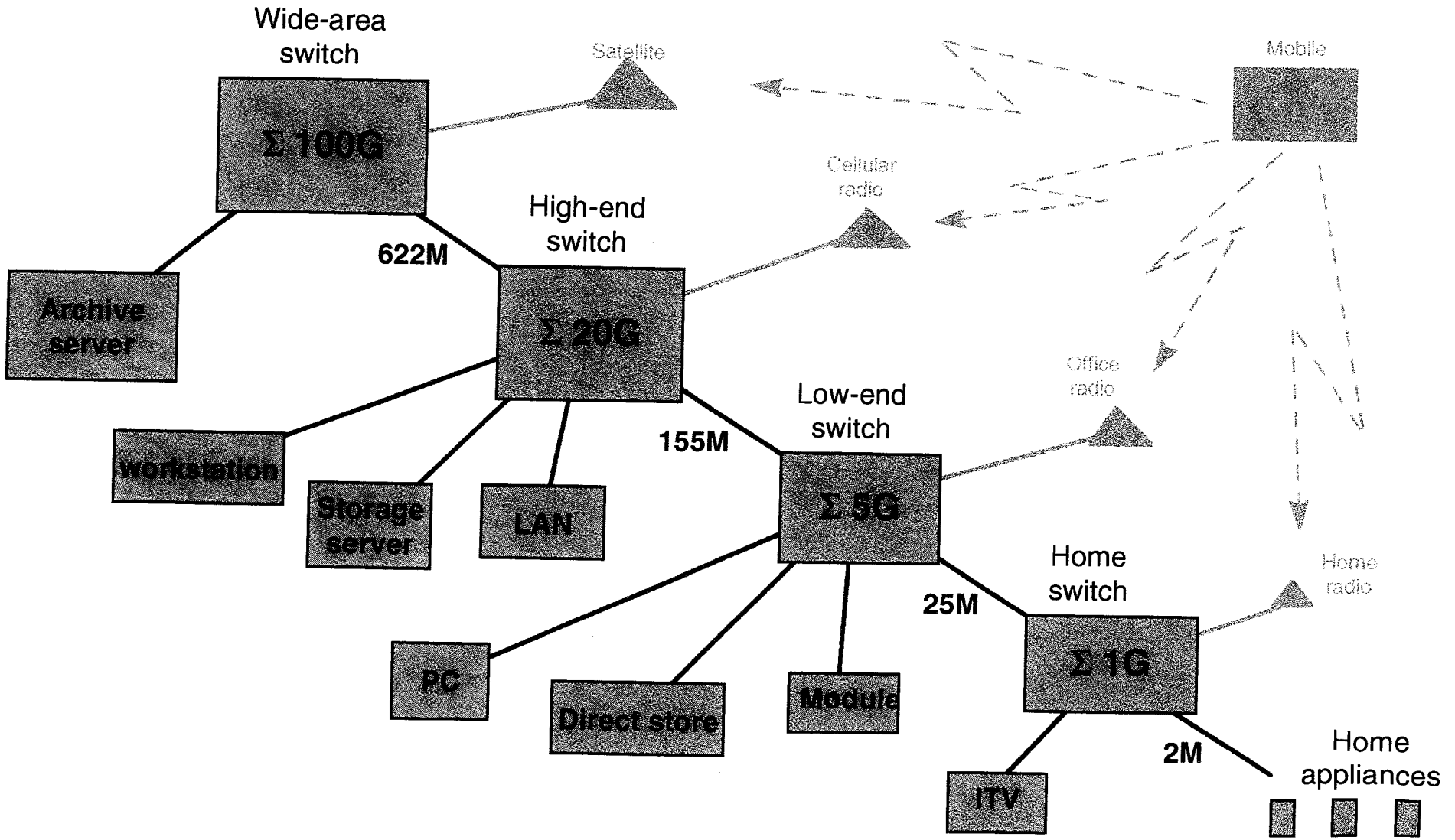
References:

1. "Global Teleporting with Java: Toward Ubiquitous Personalized Computing" - Frazer Bennett, Andy Harter, Andy Hopper, Tristan Richardson, Kenneth R. Wood - IEEE Computer, Vol. 30, N. 2. February 1997

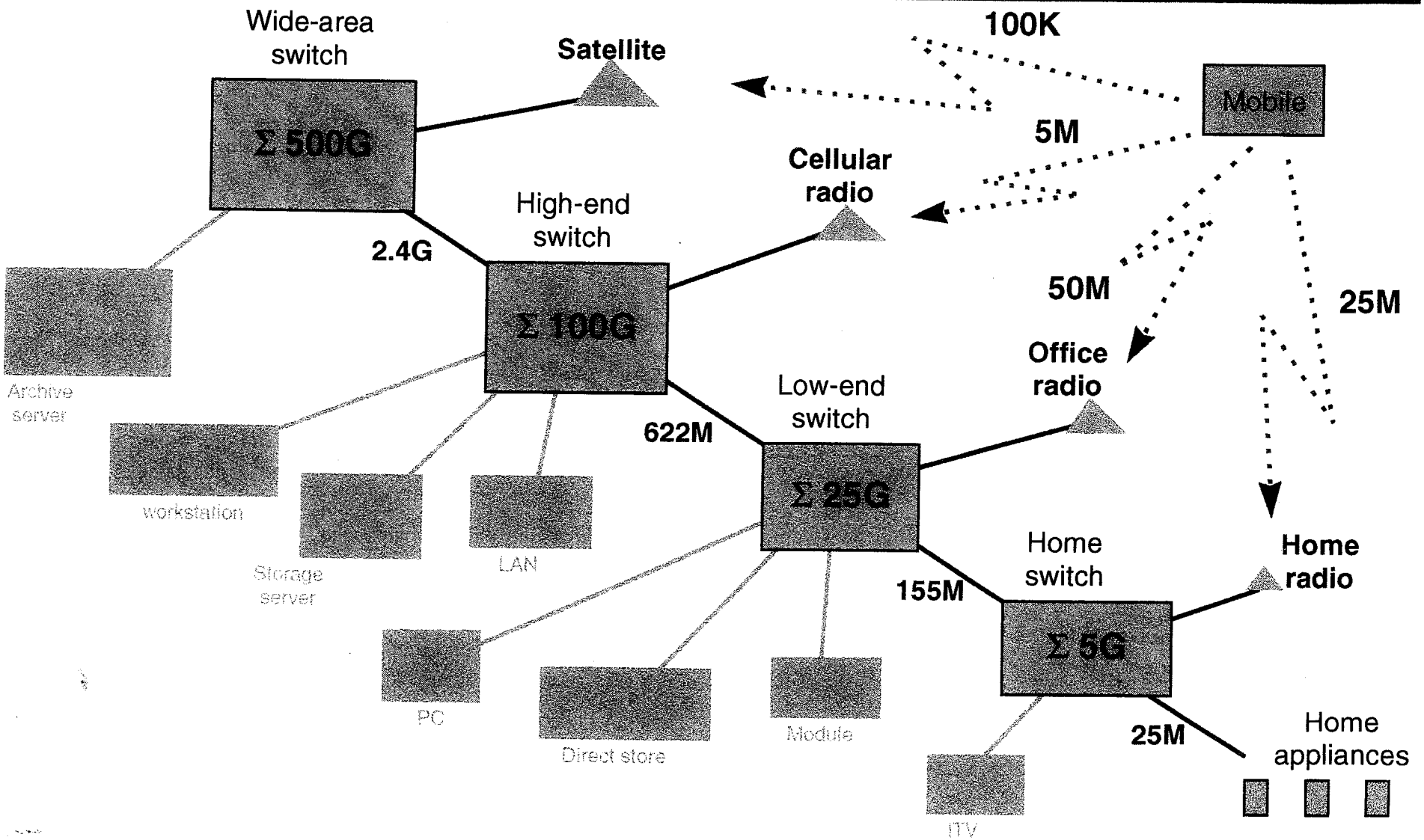
2. "A Distributed Location System for the Active Office" - Andy Harter, Andy Hopper - ORL Technical report N. 94-1, Appeared in IEEE Network, Vol. 8, N. 1, January 1994

ORL - The Olivetti & Oracle Research Laboratory
24a Trumpington Street
Cambridge CB2 1QA

ATM Everywhere (2000?)



ATM Everywhere (2005?)

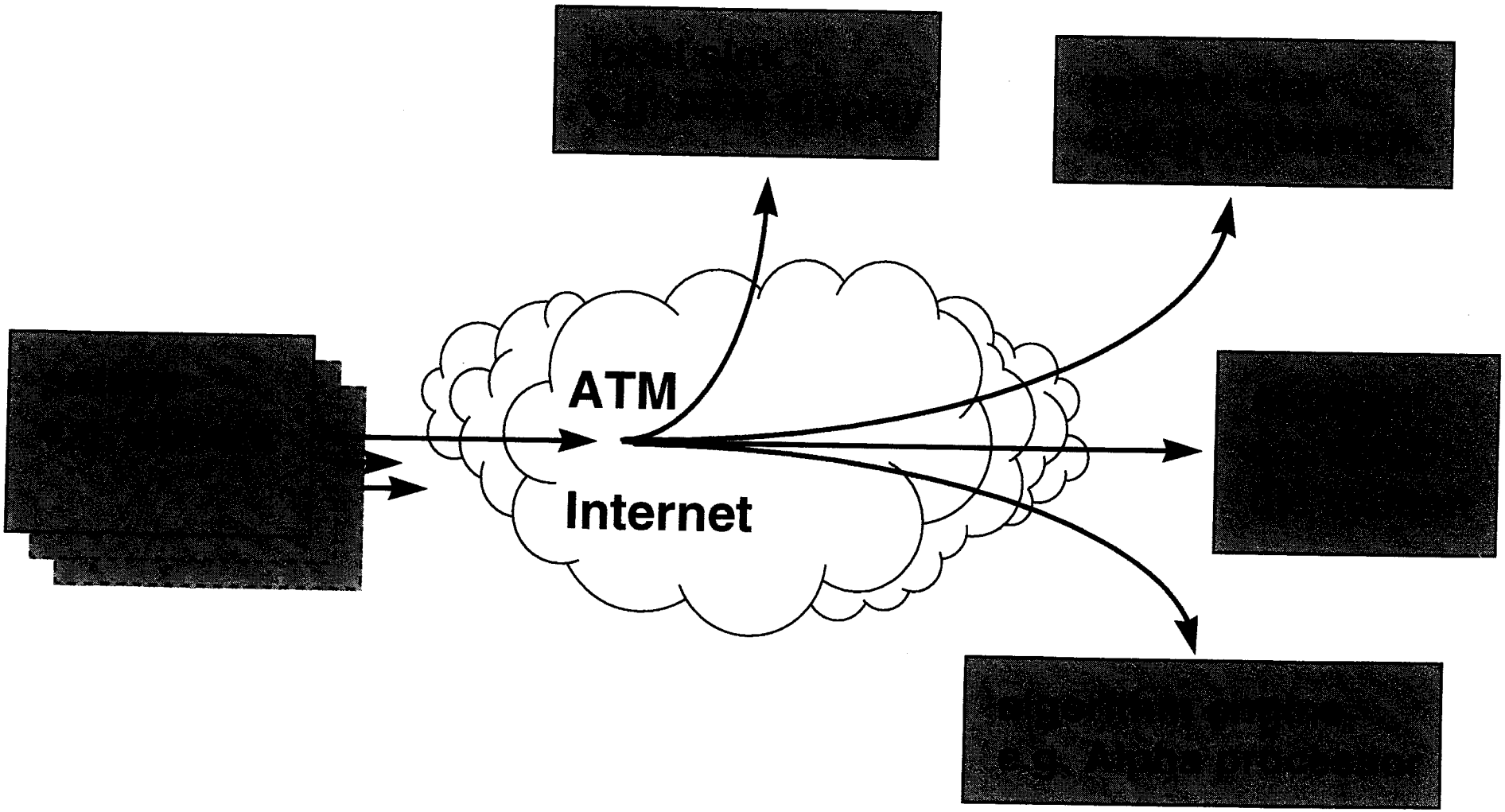


ORACLE

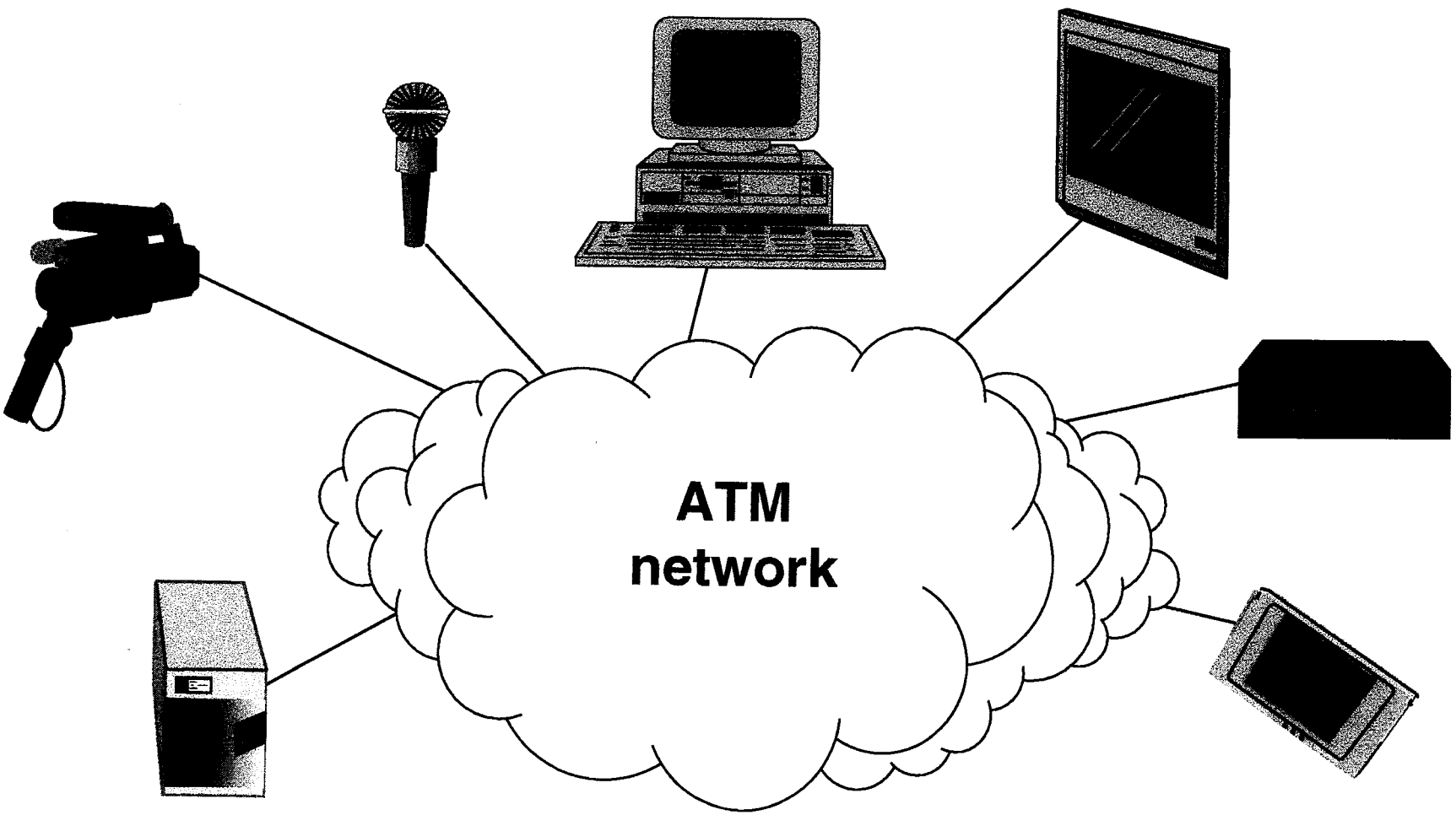
University of Cambridge Computer Laboratory

olivetti

Streams in Multimedia Applications



Network Computer Devices



2nd-Generation Applications

- Broadcast: *media server*
- Security: *look all*
- Storage: *video mail*
- Interactive communications: *video phone*

Smart feature set for all apps

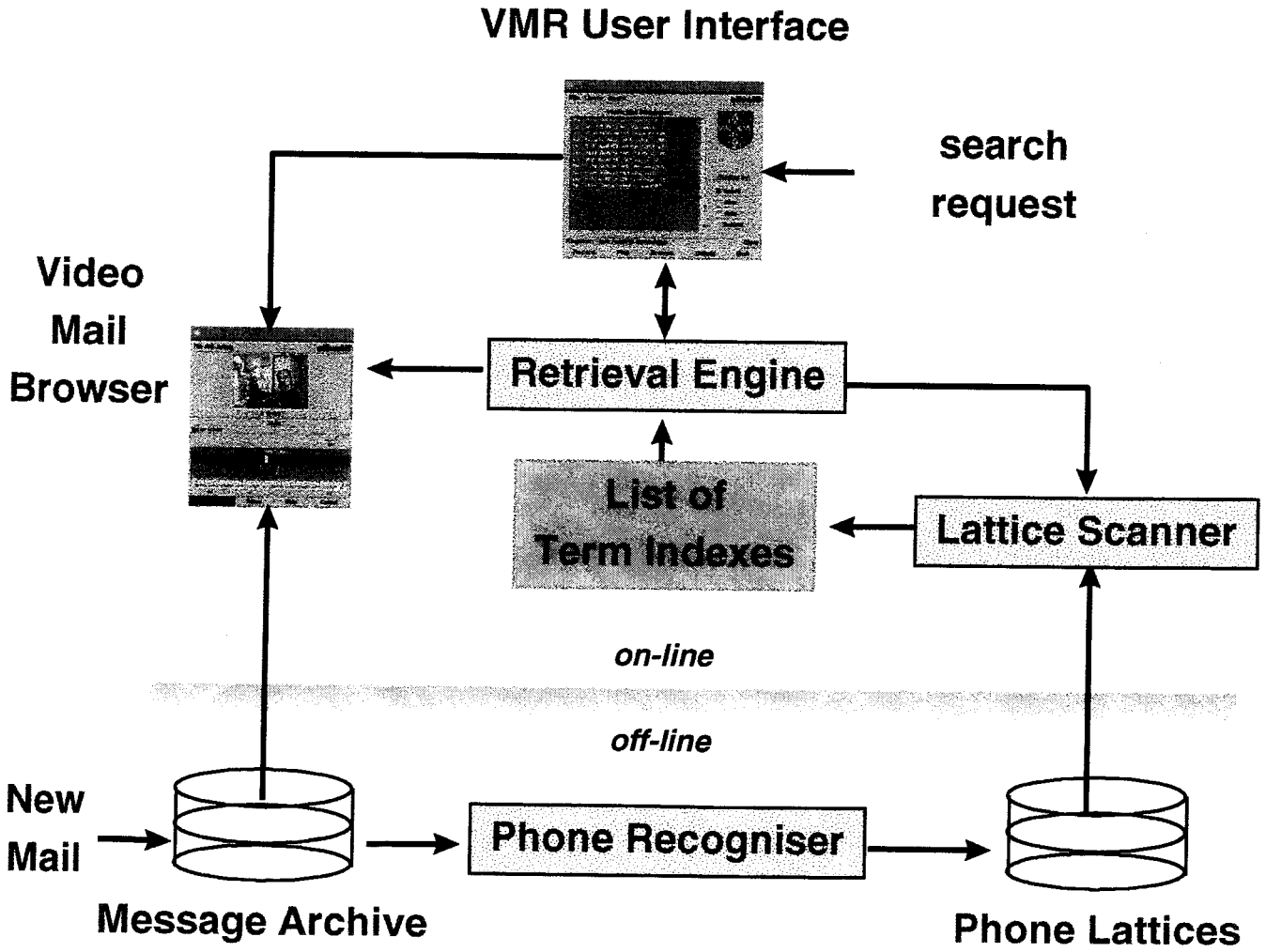
Data Analysis

Easy — Hard

Monitoring	Classifying	Recognizing	Understanding
Audio detection	Speech or music?	Speech recognition	
Motion detection	How many people?	Gesture recognition	

?

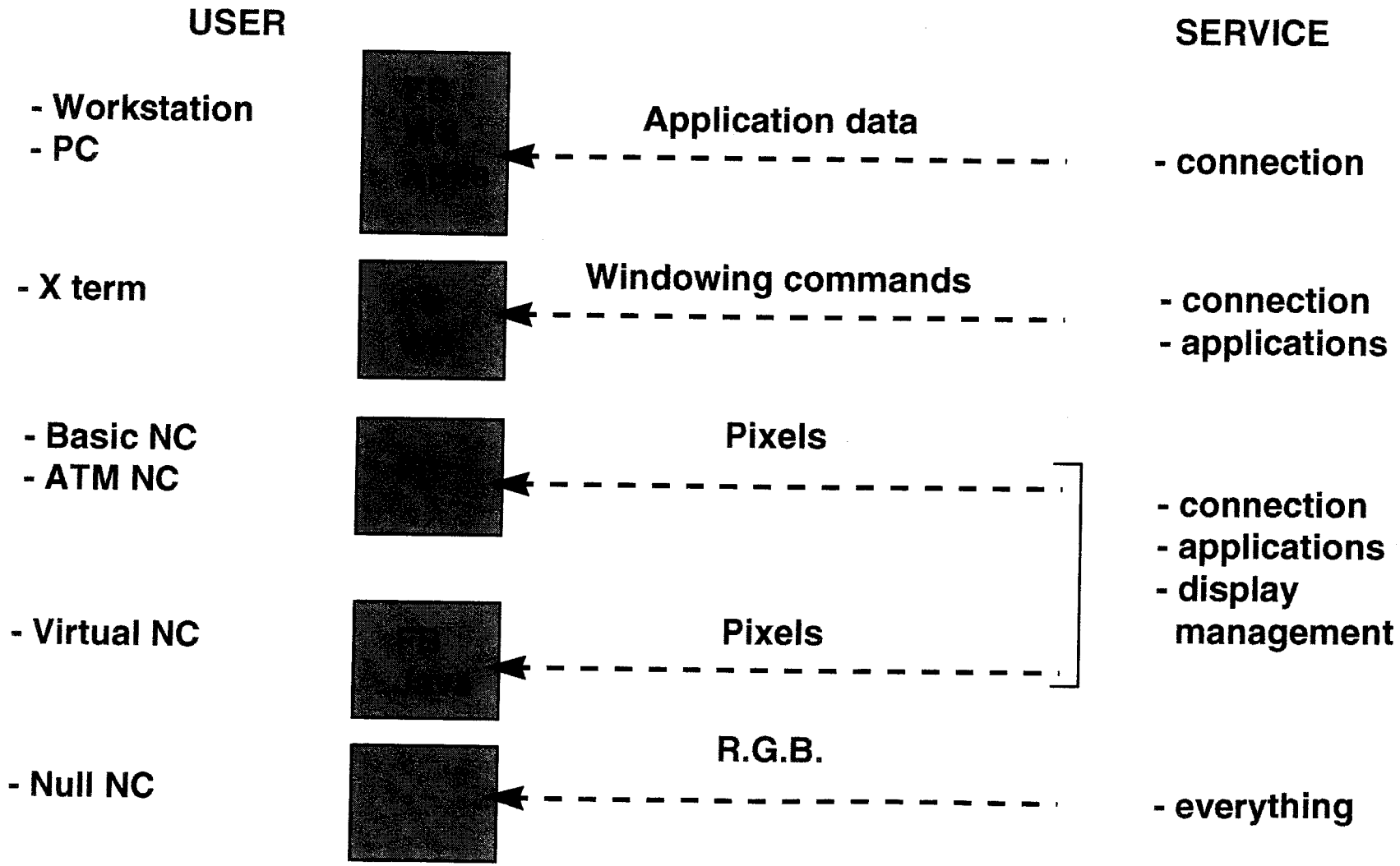
Multimedia Data Retrieval



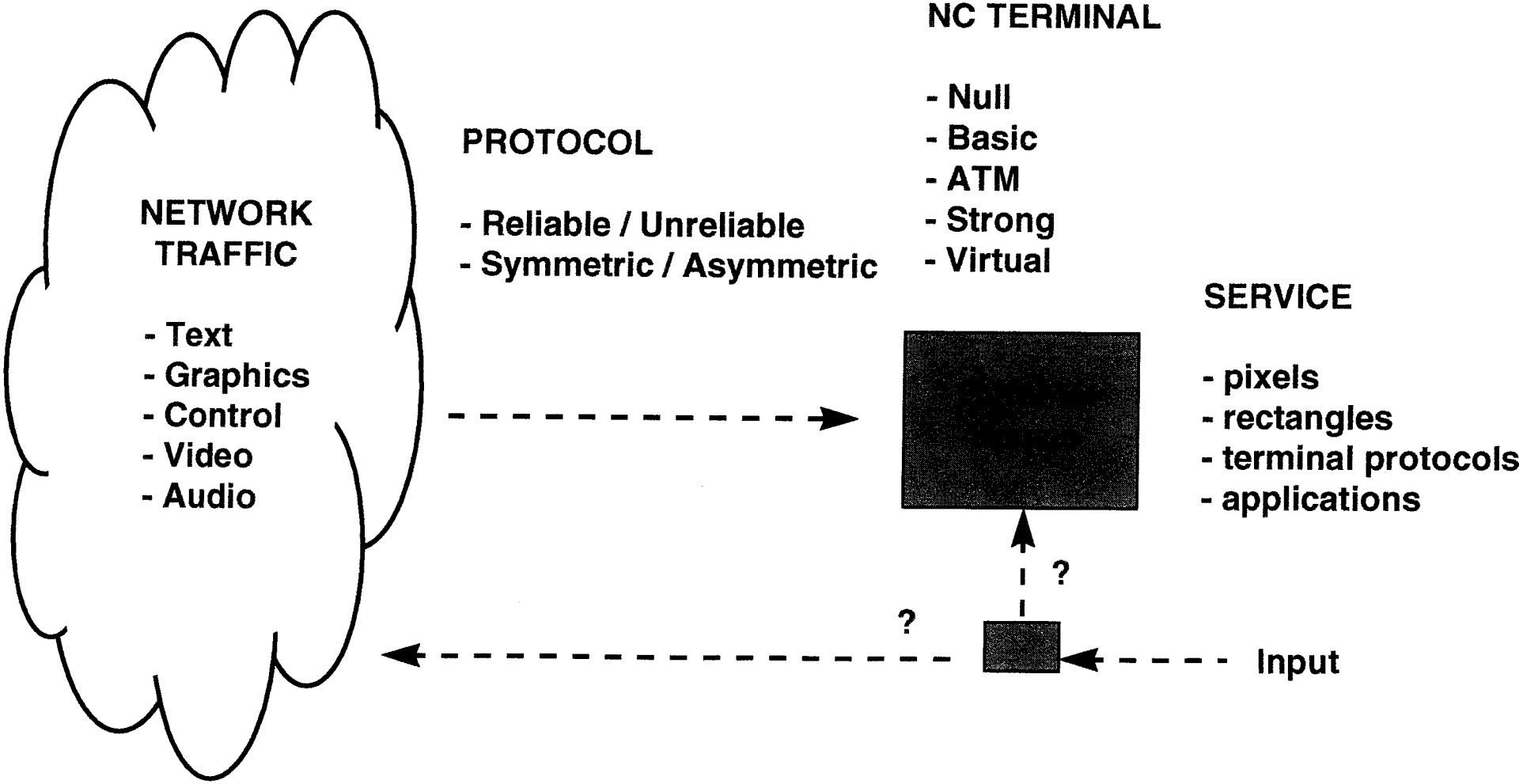
The Ideal Network Computer

- Simple no-permanent-state terminal
- Centralised services
- All media types
- Business and consumer use
- A catalyst for ubiquitous personalisation

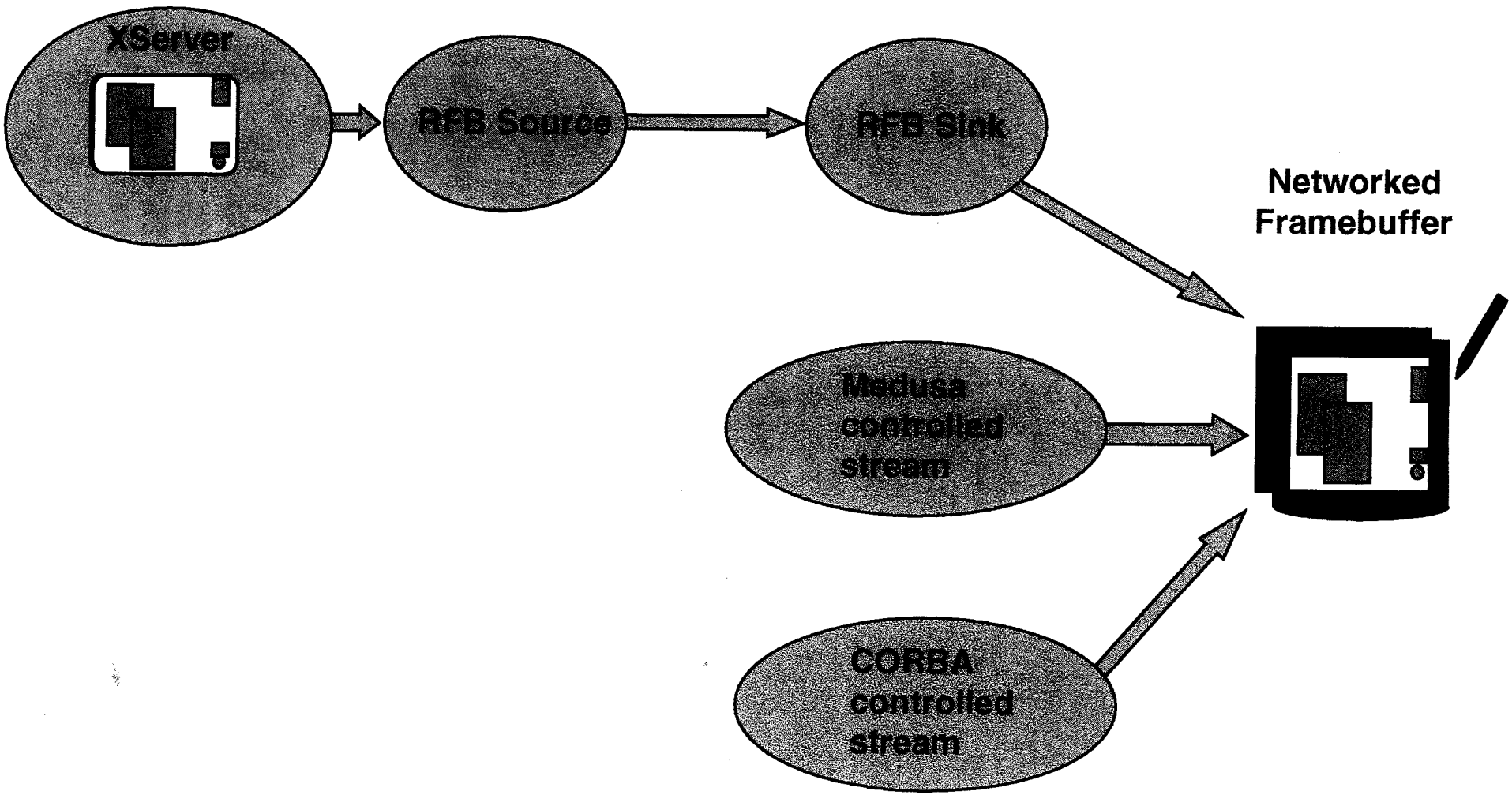
Terminal Architectures



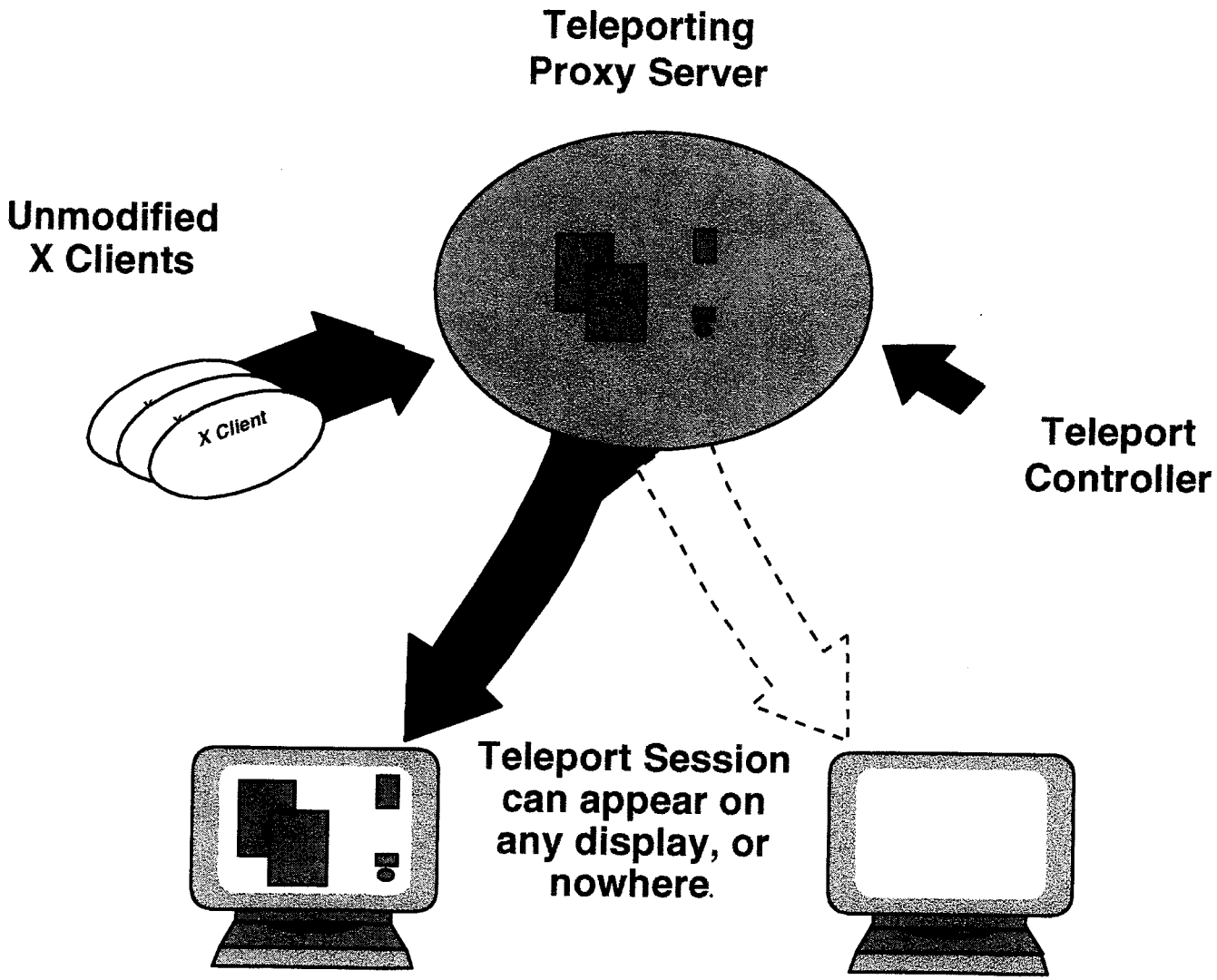
Terminal Architectures



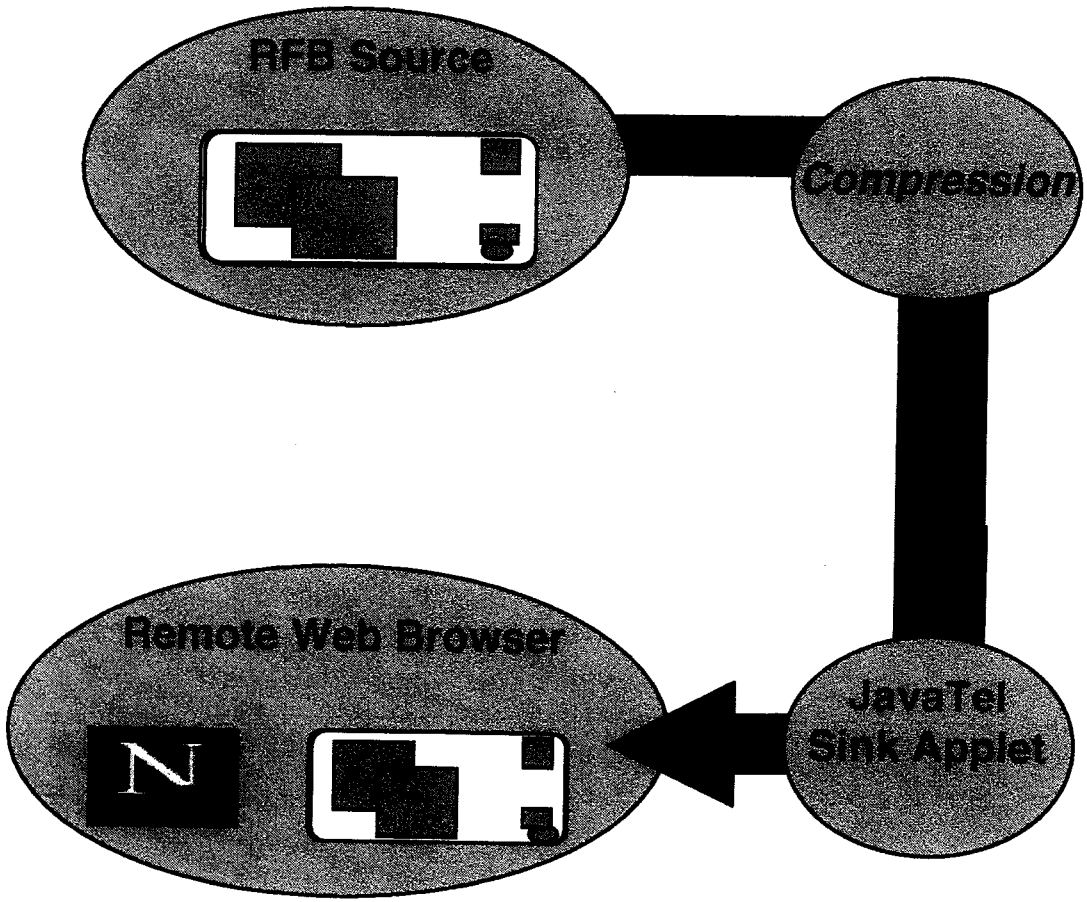
ATM NC



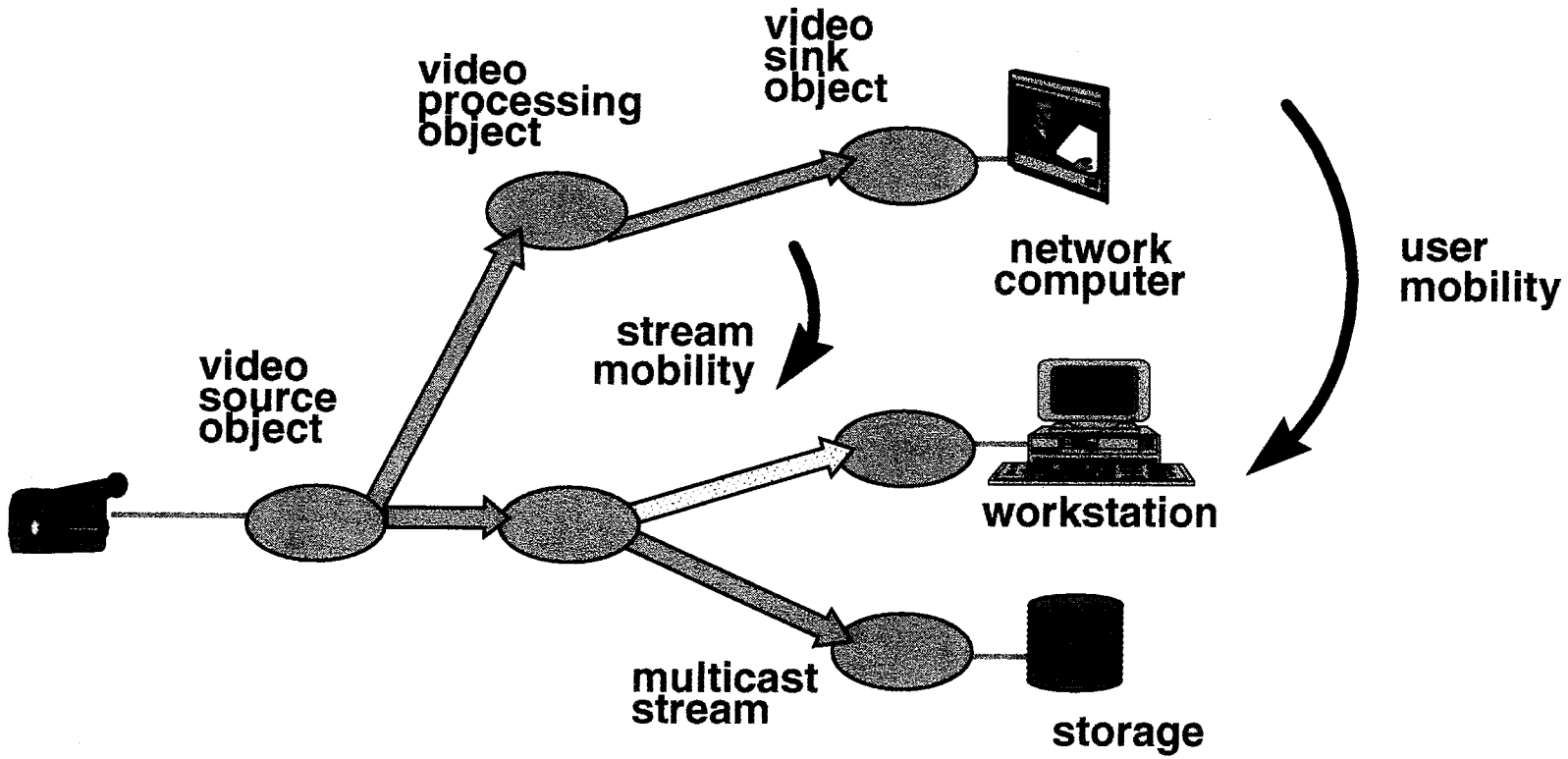
Teleporting



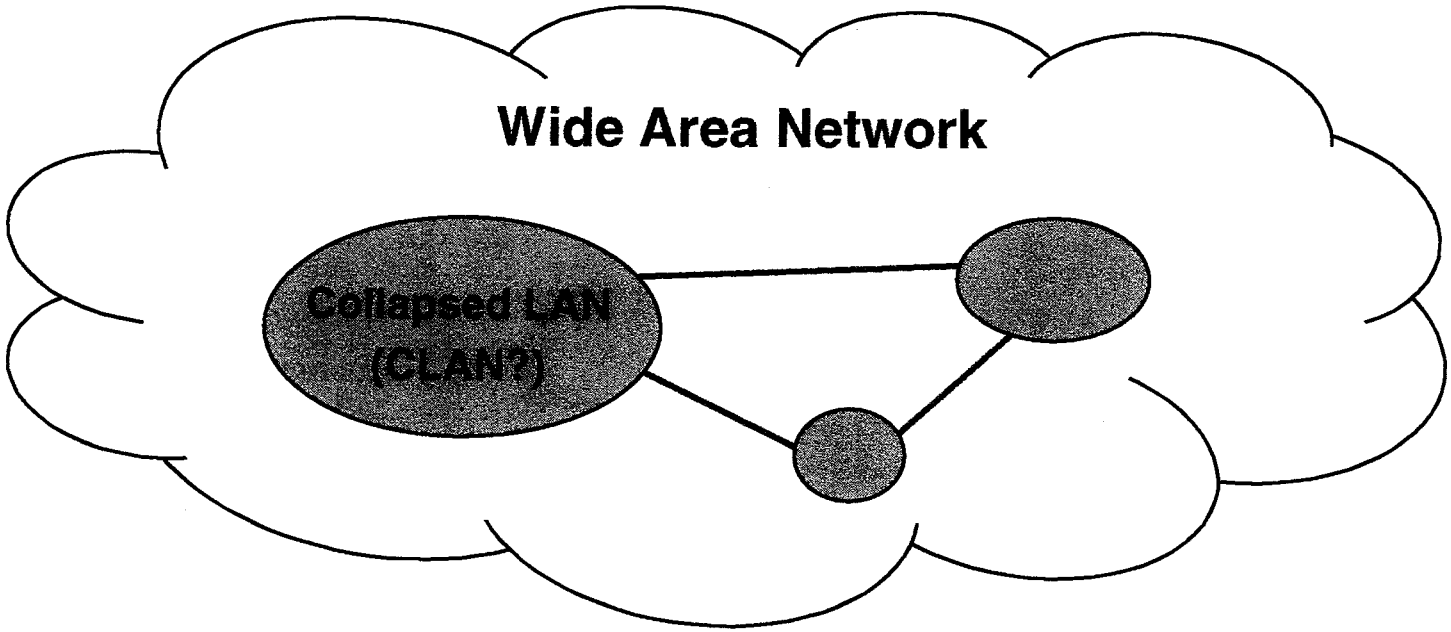
Virtual NC



Mobile Streams

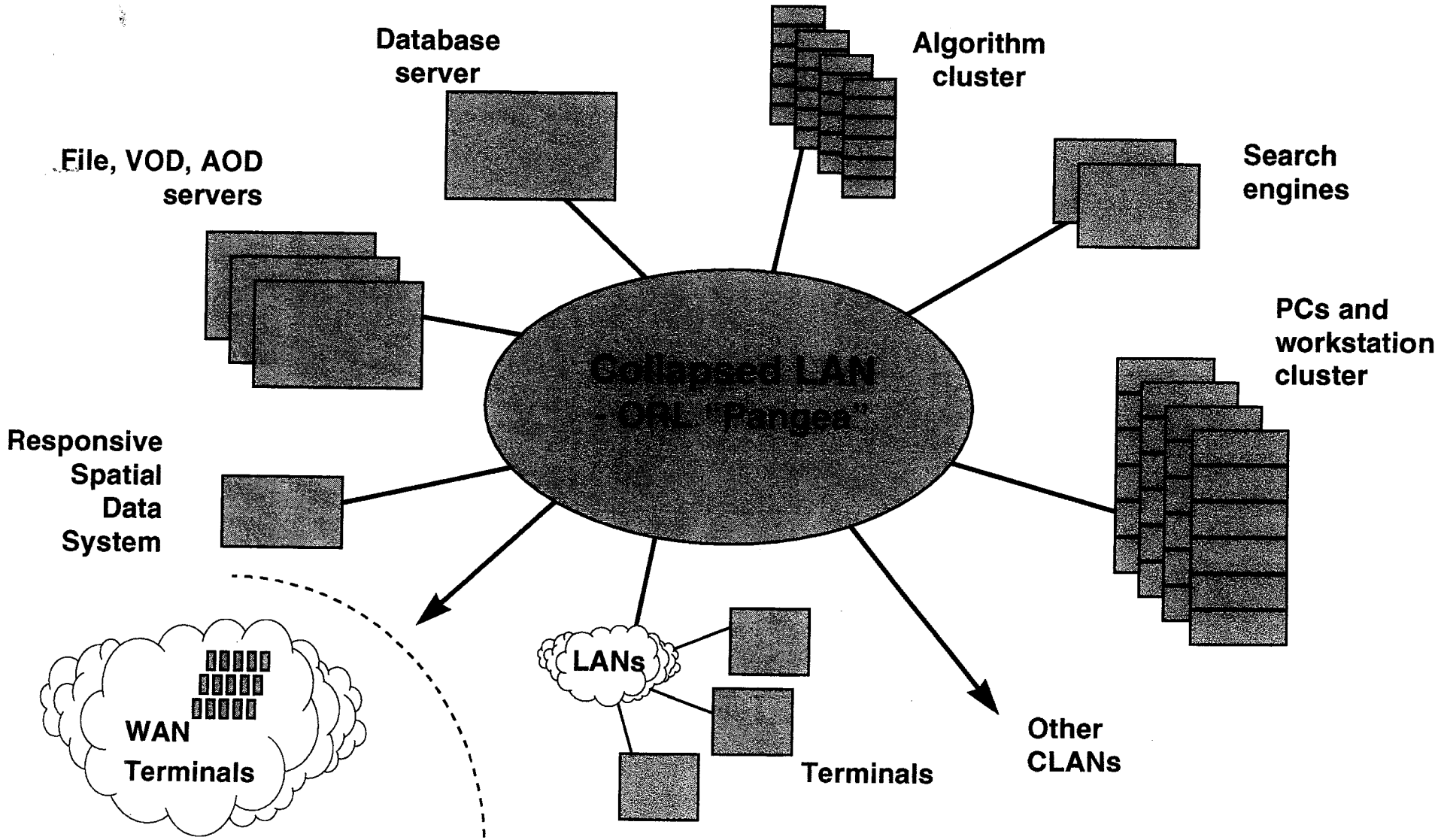


Beyond ATM ?



- Fibre point-to-point
- Fixed-time payload
- Dedicated protocols

The Internet Service Cluster

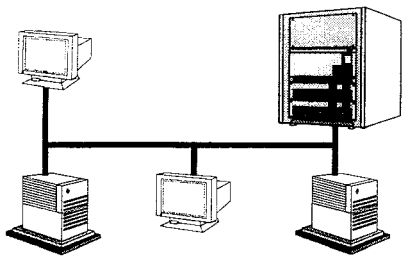


ORACLE

University of Cambridge Computer Laboratory

olivetti

Distributed Computing Technology



Andrew Herbert (ajh@ansa.co.uk - http://www.ansa.co.uk)

In the next two lectures.....

- Explain distributed computing
- Examine a real application
- Review distributed computing technology
  - focus on distributed objectcomputing with CORBA
- Distributed Systems: Concepts and Design, George Colouris, Jean Dollimore & Tim Kindberg, Addison-Wesley, 1994, ISBN 0-201-62433-8
- CORBA Fundamentals and Programming, Jon Siegal, John Wiley, ISBN: 0-471-12148-7, April, 1996

What's the real business challenge?

Coping with change

The pressures for change

- Political, economic, social, and technological...
  - Globalization

  - Rapid organizational change

  - Increased customer expectations

  - Inexpensive computing and telecommunications

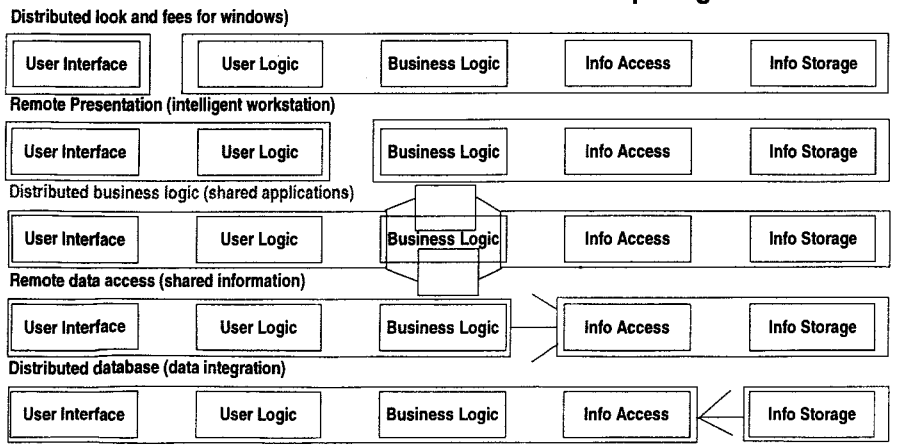
Coping with change

- The key is *evolution not revolution*
  - build on existing functional systems
  - provide new services to meet changing business needs
- Use networks to interconnect systems
  - LANs interconnected by backbone WANs
- Different parts of the system can evolve independently
  - this requires standards for *interoperability*
  - *Portability* can be useful to enable functions to mgrate from one computer to another

Architecture

- Looking at systems in high level terms
  - design principles
  - framework of components
  - interface specifications
  - assembly rules
- Hereis a simple architecture for distributed systems

Gartner models for distributed computing



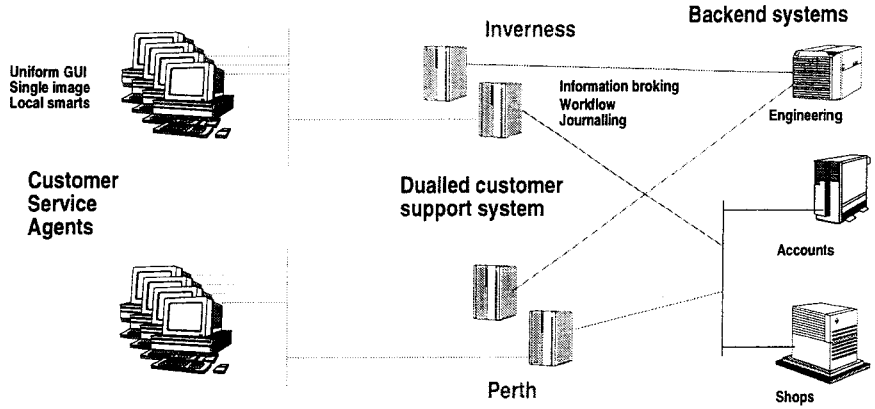
General issues for distributed systems

- Scalability
  - can the system expand as needed?
  - can the system be deployed in small and large configurations?
- Interoperability
  - can the system interwork with other systems?
- Dependability
  - can the system be made reliable?
  - can the system be made secure?
- Internationalization
  - can the system be deployed anywhere in the world?

Applications of distributed systems

- Diverse business areas
  - Airline reservations
  - Retail point-of-sale
  - Banking
  - Command and control (military, emergency services, air traffic control)
  - Telecommunications (network switching control, network management)
  - ... and many more

A Real Distributed Application: Scottish HYDRO



Distributed systems technology

- *CORBA from the Object Management Group (OMG)*
- DCE from the Open Software Foundation (OSF)
- Distributed OLE from Microsoft
- Other *proprietary* technologies (e.g., message queues, remote SQL database access)

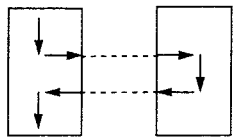
Inherent features of distributed systems

- Separation: physical and logical dispersal
- Diversity: many types of machines in the same system
- Legacy: evolution and interworking of existing systems
- Scalability: low cost of computing per machine
- Decentralization: no single point of control
- .... these differences are fundamental

Transparency for distributed systems

- These inherent features make distribution complex for the application programmer
- Keep distributed systems code separate from applications code
  - => the distributed systems technology should make network issues transparent
  - using *programming abstractions*
  - *mask complexity* but *retain control*

Abstraction #1: Remote Procedure Call in Distributed Systems

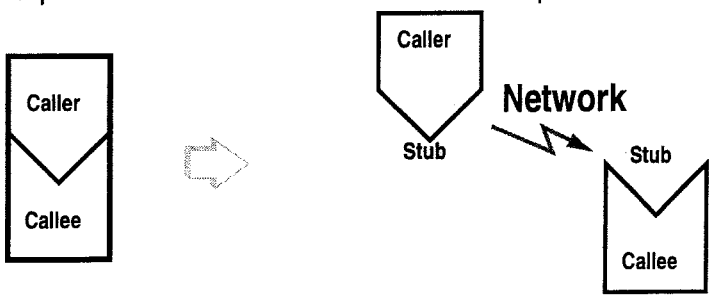


In this segment

- Explain the function of remote procedure call (RPC) in distributed systems
- Explain the importance of understanding RPC execution semantics

Remote Procedure Call (RPC)

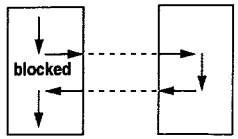
- Local procedure call can be transformed into a remote procedure call



- The caller is the client, the callee is the server

Caller Waits for Callee

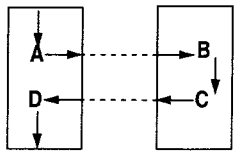
- Remote procedure calls are normally synchronous...



- ... just as in a local procedure call
- The difference is how long you may have to wait

Stubs in RPC

- The stubs in RPC are responsible for packing and unpacking the call parameters, and the call results
  - this is called marshalling (A,C) / unmarshalling (B,D):



- Stubs must allow for the fact that client and server may be machines of different types

Differing data representations

- For example, integers may be represented differently (byte-ordering)

CPU	Ordering
Intel 80x86	b0 b1 b2 b3
Digital PDP-11	b2 b3 b0 b1
Digital VAX-11	b3 b2 b1 b0
Motorola M68K	b3 b2 b1 b0

- ... and there are also different representations for floating-point and other types

RPC and transparency

- Different data representations must be allowed for
- There are three basic possibilities, only two of which scale
  - ...a single canonical 'on-the-wire' representation
  - ...'receiver-makes-right' (include representation code in messages)
  - ...'sender-makes-right' (how does the sender find out?)
- Stubs can handle the different data representations transparently
- It is worth considering whether RPC could be transparent...
  - ...so that all remote procedure calls looked like local procedure calls

Remote Procedure Call Isn't Local Procedure Call

- In an ordinary local procedure call you need not be concerned about independent failure of client and server
  - ... in a remote procedure call you must be able to handle this
- Ultimately, it is impossible to hide failures
  - ... therefore, remote procedure call *cannot* be made fully transparent
- There is no way of avoiding this issue. The conclusion is:

Design for distribution - assume remote procedure call as much as possible

RPC execution semantics

- This is reflected in the request reply exchange, which may be
  - at-least-once
  - at-most-once: the realistic case
  - exactly-once: the ideal case
- An RPC system may offer a choice of the above
  - different RPC products offer different choices with different defaults

At-least-once RPC semantics

- At-least-once semantics are appropriate for operations that have the same effect when invoked more than once
  - these are called *idempotent*
- For example
  - "add 50 units to stock level" is not idempotent...
  - ... "set stock level to 100 units" is idempotent
- OSF DCE allows you to specify an operation as idempotent
  - it will be executed more efficiently
  - ... but DCE does *not* support at-least-once semantics at all!
- At-least-once has a straightforward implementation
  - 'retransmit until acknowledged'

At-most-once RPC semantics

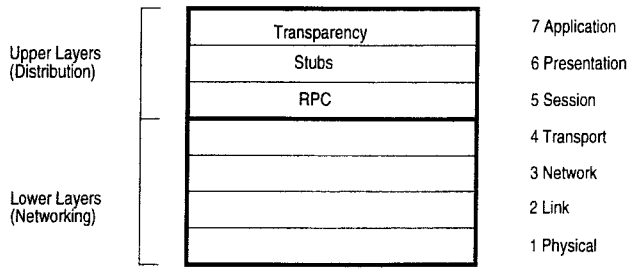
- At-most-once RPC semantics are appropriate for non-idempotent operations
- Clients must allow for operation not having occurred
  - i.e., after failure go back and check to see if the server did the last operation
- Sometimes described as best-effort
- Trivial implementation ('fire and forget')
  - in practice, retransmit until acknowledged, with duplicate suppression (via sequence numbers in messages)

Exactly-once RPC semantics

- At-least-once + At-most-once = Exactly-once
- Straightforward implementation under normal conditions...
  - 'wait for acknowledgment'
  - ... in practice, periodic retransmission and duplicate suppression
- ... much harder under failure conditions
  - requires either replication or persistent logs
  - still an active area of research; some commercial solutions are available

RPC in the communications protocol stack

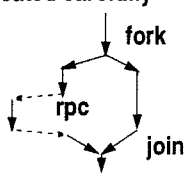
- RPC is the lowest level building block of a distributed system



- RPC is a general purpose protocol - it is not a replacement for optimised protocols (e.g. for bulk transfer)

Abstraction #2: Threads

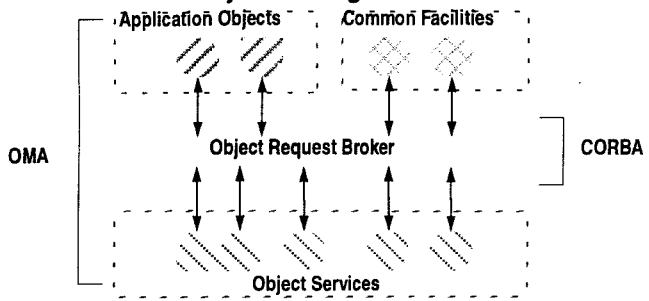
- RPC is blocking - wait for the reply
  - how can we avoid end-to-end delays?
- Use *threads* to overlap RPC and
  - local processing, local I/O, other RPCs
- Synchronisation becomes an issue
- Error handling needs to be treated carefully



Abstraction #3: Object Request Brokers

- Using RPC to distribute OO programs
- ORB = RPC + threads + local object management
- Interface Definition Language (IDL)
- Object Services

The OMG's object management architecture



- The ORB "standard" is the OMG "Combined ORB Architecture" (CORBA) specification
- Objects are Object Services, Common Facilities, or Application Objects

In this segment

Interface Definition Languages

- Explain the purpose of interface definitions
  - an abstraction to hide RPC marshalling
- Explain how to write service specifications in CORBA IDL (Interface Definition Language)
- Explore the basic constructs of CORBA IDL

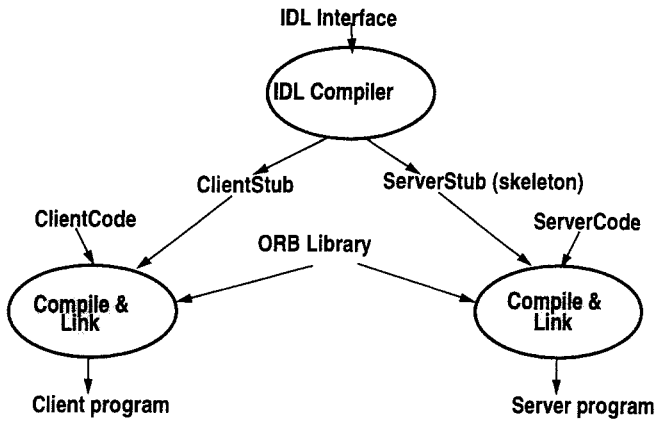
Interface Definitions

- Interface Definitions are a 'contract' between service provider and service user
  - client object -> ORB -> server object
- Each side of the interface, and the ORB,abides by the contract
- A complete contract would define the interface's
  - type
  - behaviour
  - characteristics (for example, quality-of-service)
- CORBA IDL intentionally only defines interface types

CORBA Interface Definitions

- Contain only definitions and declarations
  - no statements
  - ...they cannot be executed
  - ... they are 'sophisticated header files'
  - ... they *are* compiled to produce *stubs*
- Support many programming languages (C, C++, Smalltalk, Ada,...) from one interface definition
  - any programming language for which there is an IDL *language mapping*

CORBA IDL in a simple build process



CORBA applications and interfaces

- One CORBA application can
  - use many IDL interfaces (as a client)
  - implement many IDL interfaces (as a server - an *object implementation*)
- One CORBA interface specification can
  - be used by many applications
  - be implemented by many applications

A simple service in CORBA IDL

- This looks rather like C++

```
interface Echo {
// Comment lines start with two slashes

     string Echo (in string Message);

     string Reverse (in string Message);

};
```


Example C++ Interface Mapping

- This mapping is typical

```
class Echo;
typedef Echo *Echo_ptr;
typedef Echo_ptr Echoref;
class A : public virtual Object
{
  // Inherits from the C++ class Object defined by CORBA
  public:
  ...
  virtual char *Echo (const char *Msg) = 0;
  virtual char *Reverse (const char *Msg) = 0;
  ...
};
...
```

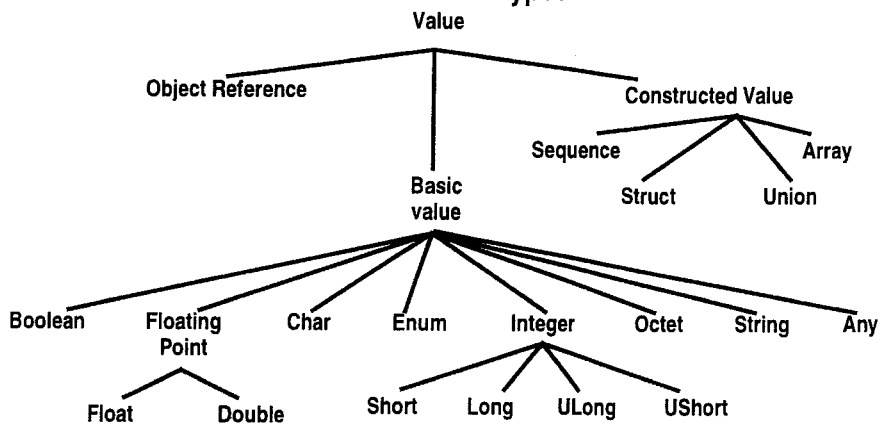
CORBA Operations

- Operations are much like function prototypes...

```
Status create_request (
   in Context        ctx,
   in Identifier     operation,
   in NVList         arg-list,
   inout NamedValue  result,
   out Request       request,
   in Flags          req_flags
);
```

- ... note the use of modes in, out, and inout
- An operation may return only a single result
  - ... this is a concession to the language mappings; most programming languages do not support multiple results

CORBA Data Types

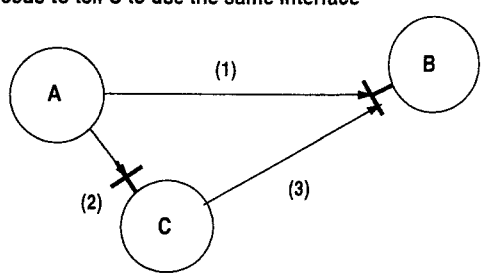


CORBA Types

- Unlike some programming languages, CORBA types have a specified set of values
  - for example, short has exactly the range $-2^{15} .. 2^{15}-1$
  - ... no more, no less
- CORBA Types Do Not Have Specified Representations
- The representation of the values is not specified
  - it will differ between machines (e.g byte ordering)
  - ... the stubs resolve this (when marshalling/unmarshalling)
  - ... the application programmer does not have to worry

Transfer of Object References

- In a distributed system, we need to be able to transfer object references
  - Object A is using Object B
  - Suppose it needs to tell C to use the same interface



  - It must be possible to pass a *reference* to B's interface between A and C

CORBA Object References

- For an object reference, just use the name of the interface

```
interface B {
   ...
};
interface C {
   ...
   void Op (in B my_B);
   ...
};
```

  - A calls C's Op operation passing a B parameter; the implementation of C can now call operations of B

- Object references can be used freely in CORBA IDL
  - not just as parameters - also in structs, sequences, unions, arrays

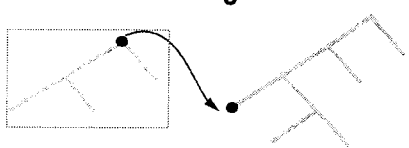
The absence of pointers

- CORBA IDL deliberately does not support pointers
  - they are not meaningful in a distributed system
  - use object references instead
  - or migrate a composite object (serialisation)

CORBAservices: Common Object Services Specification

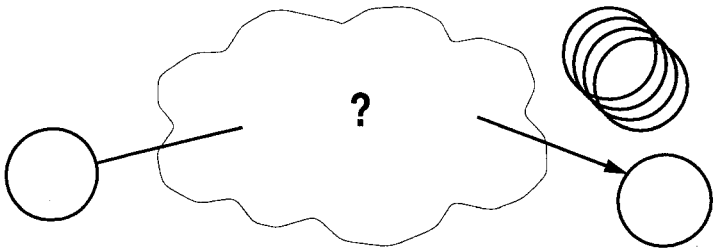
- Naming, Trading
- Persistence, LifeCycle
- Concurrency, Externalization
- Relationships, Transactions
- Security, Time
- Licensing, Properties
- Query, Event management

The Naming service



- The naming service is the simplest possible directory service
  - given a name, it will return you an object reference...
  - ...any kind of CORBA object can be named
- The naming service is a 'white pages' service...
  - it is not a 'yellow pages' service for finding objects from a description
  - ... that is the function of a separate trading service

The need for Trading



- How can clients find servers that provide the services that they need?
  - in the future, there will be millions of interconnected servers around the world
  - clients will come and go dynamically
  - servers will come and go dynamically

Naming is not enough

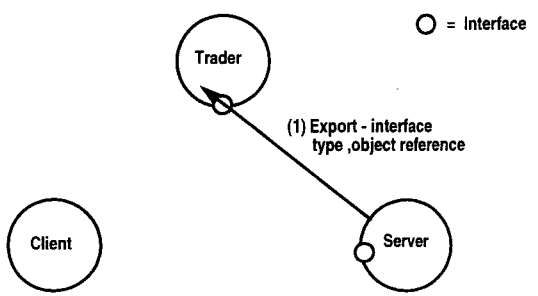
- Trading is necessary
  - we cannot rely on clients being able to name servers...
  - ....the server may not even exist when the client was created
- Trading works by matching descriptions provided by clients and servers

Trading - Basic needs

- Server must state what it provides
  - it must export a service offer
- Client must state what it requires
  - it must import a service offer
- Trading must find a service offer that matches the request
  - there may be many such offers...
  - ...there may be none

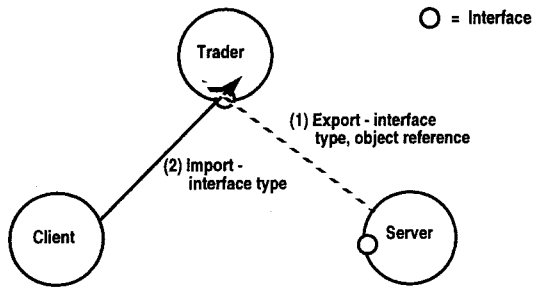
Steps in Trading - (1)

- (1) Server exports a service offer to the Trader



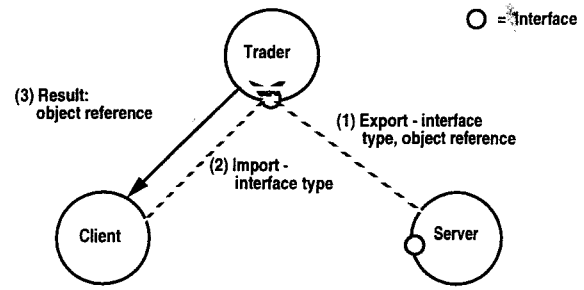
Steps in Trading - (2)

- (2) Client requests a service offer from the Trader



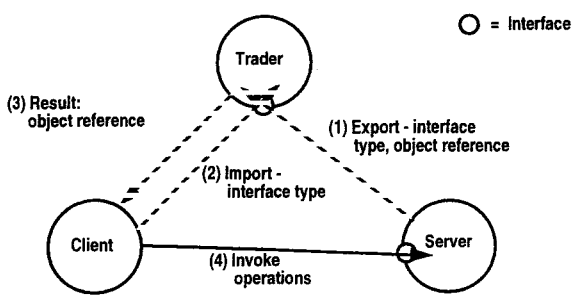
Steps in Trading - (3)

- (3) Trader returns a matching service offer to the client
  - it returns the object reference given by the server



Steps in Trading - (4)

- (4) Client uses object reference to invoke the server's operations
  - Trader takes no further part in the interaction
  - an object reference from the Trader is invoked just like any other



Matching requests with offers - type conformance

- How does Trading decide whether a client request matches a server offer?
  - it uses the interface *type conformance* concept
  - the interfacemust support at least the operations the client requires -it can provide more, but they won't be used
  - exceptions make the rules more complicated....
- If the client request and server offer interface types do not conform, they are incompatible, and cannot match

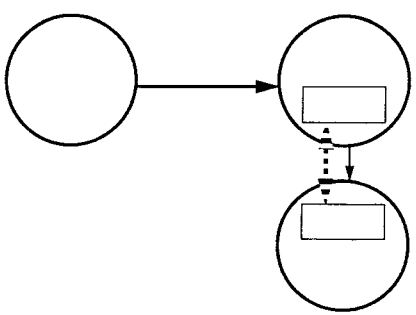
Other matching criteria

- Type conformance is not sufficient
  - who owns the service?
  - what will it cost to use?
  - where is the service, and can it be reached?
- These criteria are known as properties
  - (name, value) pairs
- Preference criteria sort matching offers into order
- Scope criteria control where to look for offers

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

Persistence Service

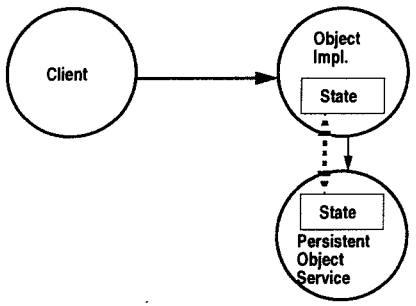


In this segment

- Examine how data storage is supported in CORBA
  - in the Persistent Object (persistence) service
  - in the Externalization service

Roles in the Persistent Object Service

- Persistent Object Service stores persistent state



Persistent state

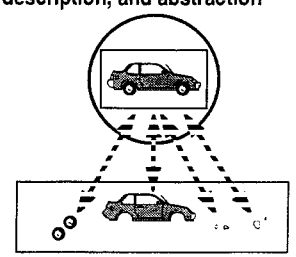
- The state of an object can be treated as two parts
- Dynamic state
  - typically in memory
  - lost if the object crashes
- Persistent state
  - typically on disk
  - preserved over crashes, and can be used to reconstruct the dynamic state

Preserving state

- It is an object's responsibility to preserve its dynamic state as persistent state
  - and to recover it after a crash
- The object may use the Persistent Object service for this purpose
  - or any private mechanism it chooses instead...
  - ... flat files
  - ... direct access to a relational database
  - ... direct access to some other kind of database
- The object may delegate the responsibility back to its client

Advantages of using the Persistent Object Service

- Typical data storage mechanisms do not have object characteristics
  - uniform interfaces, self-description, and abstraction

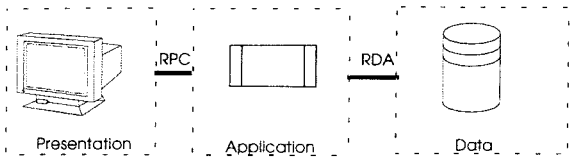


- This is sometimes known as an 'impedance mismatch'

Client Control

- Clients may need to control or manage persistence of the objects they use
- In particular, the client may need to control
  - exactly when persistent state is saved and restored
  - which copy of persistent state is to be used
- Object implementations do not have to provide client control
  - they may choose to hide the complexity from the client

Digression on databases in distributed systems



In this segment

- *Explain the Remote Data Access (RDA) technique*
- *Examine OMG approach distributed database and transaction processing*

Two worlds

- *Two approaches to distributed systems are predominant*
  - Remote Data Access (RDA), generally used for databases
  - Remote Procedure Call (RPC), generally used for distributed computing
- *RDA is based on the SQL structured query language*
  - usually to access a relational database
- *RPC is based on IDL interface definition languages*
  - usually to access an application server
- *Both are 'client-server approaches'...*
- *OMG unifies them through relationship, query and property object services*

Remote Data Access (RDA) Overview

- *Application interacts with database using SQL statements*
- *Statements can retrieve, update, delete, or insert records:*
  - a single record
  - a set of records at once
  - a set of records, one at a time
- *Statements can be grouped into transactions*
  - a transaction succeeds or fails as a whole
- *The location of the remote database is transparent*

RDA Advantages

- *Flexibility for clients to define application-specific views and ad hoc queries*
- *Vendor-supported interfaces to application development tools*
  - data browsers
  - report writers
  - 4GLs
  - spread sheets
  - graphing packages
  - ...and so on

SQL (Structured Query Language)

- Example

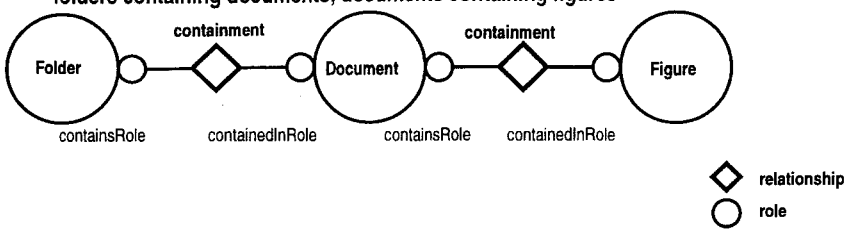
```
SELECT employee_name, title, commission
   FROM    employees
   WHERE   employee_name IN
           (SELECT   employee_name
           FROM      employees
           WHERE     commission > 8000
           AND       title = 'account_manager');
```


Databases and the OMG

- Represent data elements as *objects* (with methods)
- Label objects with *properties*
- Structure sets of objects using *relationship* objects (e.g. containers)
- Use persistence service to move objects to/from disc
- Permit queries to match by property within a relationship
  - this is an *Object Database* model
  - the relational (SQL) model is a special (optimised) case

Relationship Service

- An example of containment
  - folders containing documents, documents containing figures



  - 2 relationships, 4 roles, 9 objects involved
  - represent relationships as container objects
- Heavily used in OO document architectures (Microsoft OLE, OMG OpenDoc)

The Properties service

- Allows typed, named values to be dynamically associated with objects
  - attributes defined in CORBA IDL are static...
  - ... adding a new attribute means changing the IDL
- Client applications can get and set both properties and attributes
  - properties can be created and deleted...
  - ... attributes cannot

The Query service

- Is a general query service and framework, embracing
  - SQL, for relational databases
  - OQL, for object databases
- Allows queries to be prepared, and their status checked later
- Allows queries to be nested

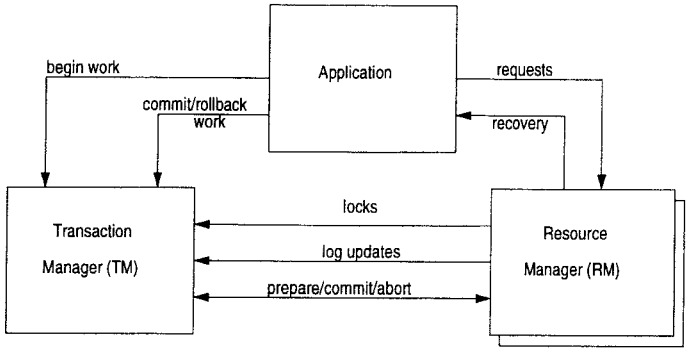
The Transaction service

- How can we ensure database updates really happen if there are partial failure?
- Use online transaction processing (OLTP)
  - extend RPC with two-phase commit
- OLTP can also be used to make method invocation robust
- OLTP has a high performance overhead compared to simple RPC
  - trade fault tolerant application design cost against performance cost of OLTP

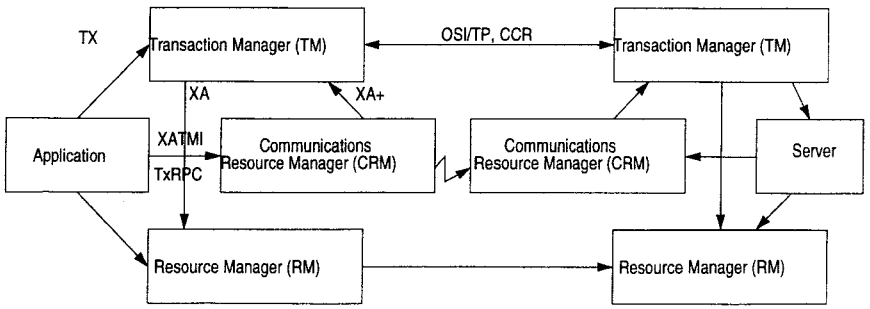
Transactions - the ACID properties

- *Atomicity*
  - all-or-nothing
- *Consistency*
  - transactions must be self-contained logical units of work
- *Isolation*
  - concurrent transactions must not affect each other
- *Durability*
  - what's done must not be undone

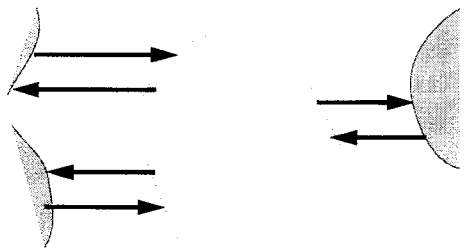
Typical TP Core Components (much simplified)



X/Open and ISO TP Model



The Event service



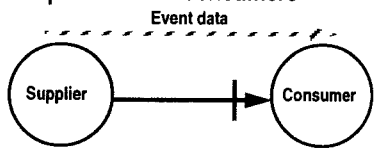
- Events are concerned with asynchronous communication between objects

Events

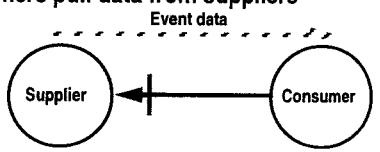
- Events support asynchronous notification
  - ...'alerts', 'change notification'
  - for example, when disk space is getting low
- Suppliers and consumers of events are decoupled
  - via an event channel
- There can be multiple suppliers and multiple consumers
  - so-called "publish and subscribe"
- Events could be used to build an RQM (Robust Queued Messaging) interface
  - or to interface to an existing one

Push and pull models for communicating event data

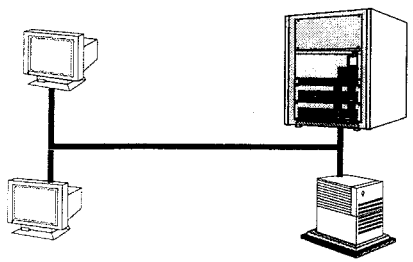
- Push model: suppliers push data to consumers



- Pull model: consumers pull data from suppliers



Secure Commerce on the Internet



Andrew Herbert (ajh@ansa.co.uk - http://www.ansa.co.uk)

In the next two lectures.....

- Introduce WWW and Java
- Introduce network security
- Look at an example of secure electronic commerce

In this segment

- Describe the World Wide Web (WWW) and its essential components
  - HyperText Transfer Protocol (HTTP)
  - HyperText Markup Language (HTML)
  - Java and JavaScript
- How the WWW is affecting the development of distributed systems
- What effect this will have on electronic businesses

What is the WWW?

- The WWW is the largest information system ever created
- It is truly distributed
  - there is no centre
- The essentials are standardised by use and the IETF

Precise definition not possible

- No narrow definition of the WWW entirely fits
- The WWW includes and simplifies access to
  - ftp
  - Gopher
  - Mail
  - News
- For the purposes of this presentation, the WWW is
  - A network of webservers and browsers...
  - ...using HTTP+HTML+Java

HTTP

- HTTP is the HyperText Transfer Protocol
- Defined by an ongoing series of internet-drafts
- Built to on top of TCP/IP, and using TCP/IP transparently
  - TCP/IP is the Transport Control Protocol/Internet Protocol
- HTTP is a *connectionless* RPC
  - no state is stored between connections
  - alternatively: connections do not remember previous connections

HTML

- HTML is the HyperText Markup Language
  - originally defined as a subset of SGML
  - some vendors have made nonstandard extensions
- Defined by an ongoing series of internet-drafts
  - latest version is HTML 3.0
  - HTML 2.0 is currently heading for becoming a standards track RFC
- HTML does not define presentation of a document rigidly
  - HTML tags describe logical content (paragraphs, headings)
  - exact presentation is under control of user's browser
  - allows for access from wide range of terminal types

WWW Browsers

- The browser is a client in distributed systems terminology
- Responsible for retrieval and presentation of HTML documents
- Communicates with WWWserver via HTTP for retrieval
- The user's local machine resources are used for the presentation

Web Servers

- Web server delivers documents in response to browser requests
- Server is a *process* that listens on a predefined port
- When it hears an incoming HTTP request it
  - forks a new child process to handle the request
  - hands the request and the connection to the child
  - resumes listening on its port
- The child process then
  - locates the requested document
  - returns it to the browser by HTTP
  - drops the connection and expires

The Common Gateway Interface

- Many services are based on delivery of information that must be
  - live: delivered in real time
  - customised on a per-use basis
- CGI was developed to allow the server to handle these cases
- CGI mechanism is standardised and controlled by NCSA

CGI Programs

- CGI programs are pieces of code runnable by the Web Server
- They are specially privileged
  - the server can only run them from its cgi-bin directory
  - they must be extremely well trusted before installation
- They can be written in any programming language
  - the WWW community favours scripts (tcl, perl, csh, Java)
  - since these can easily be distributed and shared
  - and checked for security holes

Introduction to Java

- Java is a normal programming language
- Basically a lot like C++
  - with a few improvements added...
  - ...and a few undesirable features removed
- Java is special with regard to the WWW because
  - it runs on a *virtual machine*, so is platform independent
  - bytecodes for the virtual machine can be moved across the WWW...
  - ...and run remotely, in the user's browser

Java Language Design

- The Java language specification is public
  - although owned and controlled by Sun
- Java is object-oriented
  - uses C++ based ideas and syntax
- Java omits C++ assumptions about memory space structure
  - no pointer arithmetic or conversions
  - means Java programs are "secure" in a programming sense, but not in a systems sense!
- Java is designed for distributed systems technology

Java Application Anatomy

```
import java.util.Date;
        class DateApp {
          public static void main (String args[]) {
            Date today = new Date();
            System.out.println(today);
          }
        }
```

- This example, DateApp, is from Sun's Java Tutorial
- Main concepts shown are
  - class membership hierarchy ("." is membership operator)
  - type hierarchy (DateApp inherits "Date" type from java.util)

Java Virtual Machine and Byte Codes

- Java compiler generates byte code for a virtual machine
- This virtual machine is the Java byte code interpreter
- The interpreter is architecture neutral
  - widely ported (mostly by third parties)
  - no implementation dependencies permitted
- Only byte codes are shipped from web server to browser
  - first verified by compiler
  - source code is not distributed
  - but byte codes include information about interfaces

RMI and Serialisation

- Remote Method Invocation (RMI)
  - inherit fromRemote class to pick up RPC library and bind (=trading) methods
  - no IDL, generic stub uses interface type (held in byte codes) to steer marshalling
  - constants marshalled as data
  - references marshalled as remote references (c.f. CORBA)
  - much better integration than CORBA/C++
- Serialisation
  - inherit from Serialization classes to snapshot state and byte codes
  - can be sent as RMI arguments and results
- Together give mobile objects

Verification and Security in Java

- Java is intended for networked distributed environments
- Language restricted to disable most illegal memory accesses
- Byte codes verified for safety during compilation
- Byte codes authenticated using public key encryption techniques (see later)
- Byte codes executed inside a safe interpreter
  - client can restrict access to local filesystem and communications

WWW Security

- Web sites must guard against security compromise
- Generic problems
  - remote filesystem accesses
  - remote client executing privileged programs on Web Server
- Wide range of holes must be plugged

Firewall Defences

- Firewalls are the standard defence for webservers
- Two-firewall configurations are favoured
- Webserver is placed inside outer firewall
  - protected by packet filters
- Corporate Intranet is behind an inside firewall
  - protected with a huge array of defences

More information?

- For more general information on WWW searching
  - <URL:http://www.altavista.digital.com/>
  - <URL:http://www.yahoo.com/>
- For more on HTTP
  - <URL:http://www.w3.org/pub/WWW/Protocols/>
- For more on HTML
  - <URL:http://www.w3.org/pub/WWW/MarkUp/>
- For more on CGI
  - <URL:http://hoohoo.ncsa.uiuc.edu/cgi/>
- For more on the Java language
  - <URL:http://java.sun.com/>
  - <URL:http://java.sun.com/tutorial/index.html>

Even More Information

- For more on Firewalls
  - see *Firewalls and Internet Security* by Cheswick & Bellovin (Addison-Wesley 1994).
- For more on relevant IETF standards activities
  - <URL:http://www.ietf.org/>
- For more on W3C activities
  - <URL:http://www.w3.org/>

Network Security

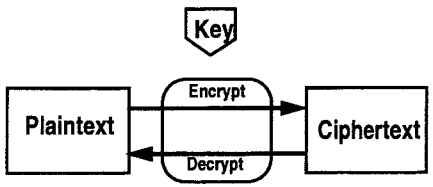
- In this segment
  - understand security issues in network systems
  - understand basic concepts of cryptography
  - understand key management
- PGP: Pretty Good Privacy, Simon Garfinkel, O'Reilly & Associates
  - play with PGP software too - widely available on the Internet
- Applied Crytography: Protocols, Algorithms and SourceCode in C, Bruce Schneier, John Wiley & Sons, 1994

Security

- Authentication
  - how can you prove who you are to a distant host in a way which can't be mimicked by an imposter
- Integrity
  - how can you prove a message is the right one to process next
  - how can you prove a message hasn't been forged
- Confidentiality
  - how can you stop other people reading your messages
- Non-repudiation
  - how can you stop people denying you sent them messages?
- -> CRYPTOGRAPHY!

Cryptography

- Message you wish to encrypt (plain text)
- The message after it is encrypted (ciphertext)
- Encryption key
- Encryption algorithm



Encryption Algorithm

Keys

- Keys are just numbers
  - hence many government restrict import or export of algorithms!
- The longer the key, the more security the encryption system delivers
  - hence many governments restrict key length
  - key escrow/recovery may be the way out.....
- Breaking the code
  - brute force - try every possible key (128 bit key at 10**9 trials/sec on 10**9computers = 10**13 years; age of the unverse = 1.2*10**10!)
  - known plain text (e.g. email headers, sequence numbers, etc)
  - chosen plain text (fool user into sending a message of your chosing)
  - differential cryptanalysis of similar plaintext

Private Key (Symmetric) Cryptography

- Both sender and recipient share a single common key which both keep secret
  - problem of key distribution has to be solved by physical means (e.g., couriers)
- Data Encryption Standard (DES) - 56 bit
- Triple-DES (DES three times with two keys) - used by banks
- RC2, RC4 - 1 to 1024 bit - Prof Ronald Rivest MIT / RSA Security
- International Data Encyption Algorithm (IDEA) - 128 bit - used in PGP

Private Key Distribution

- Need a different key for each pair of communicating parties
  - n(n-1)/2 keys for n people!
  - doesn't scale
- Use a *secured* key distribution centre
  - everyone has secret key with the KDC
  - any pair can ask the KDC to issue a short term secret "session" key
  - KDC can do mutual authentication as part of key distribution
  - but beware "replay" and "man-in-the-middle" attacks
- Disadvantages
  - KDC is obvious place to attack (physically)
  - KDC has to be online to all potential users, all the tme
- Works best for "many-to-one" communication

Public Key Cryptography

- Based on two keys: public key and secret key, generated as a pair
  - public key can be told to anyone (e.g. published in a directory, a newspaper, etc)
  - secret key must be physically secure
  - a message encrypted with one can be decrypted with the other (either way)
  - sometimes called asymmetric cryptography

Rivest-Shamir-Adleman (RSA) Algorithm

- Encryption and decryption algorithm are the same
- Benefits:
  - no need for a secure KDC, put you private key anywhere
  - no need even for KDC to be online
  - good for many-to-many communication - only *n* key pairs needed
- RSA is slow
  - use RSA to establish *symmetric* session key

What encryption can't do

- Can't protect your unecrypted information!
- Can't protect against stolen keys
- Can't protect against descructive attacks (denial of service)
- Can't protect against a booby-trapped encryption program
- Can't protect you against a traitor
- Can hide the fact a message was sent (traffic analysis)

Digital Signature

- How do I know a public key is valid and not a spoof?
- Have the key digitally signed by a certification authority with a very well-known key
  - i.e., published in lost of independent places so I can cross check them all
  - or operated by some agency I trust (my bank? my government? my friends?)
- To sign a message
  - compute a message digest (e.g. MD5 or SHA algorithms)
  - a one-way function which distills the message into a large (e.g. 128bit) number
  - distribute message and the digest
  - recipient also computes digest and compares with transmitted digest
  - guarantees integrity of whole message

But who made the digest?

- Encrypt the digest with your secret key
- Recipient can decrypt this with your (well-known) public key
- If decrypted digest = digest of plain test the recipient knows this message came from you
  - it has been digitally signed
- Can use other keyed one-way functions than encryption
  - allows use of strong keys for integrity even if not allowed for confidentiality

Applications of digital signature

- *Certification authorities* publish signed public key to user name mappings (sometimes called "credentials")
  - Verisign Inc and others
- PGP model users exchange signed keys to build up Web of trust
  - Accept validity of keys signed by e.g. 3 or more people you trust
  - Only add your signature if you've physically checked....
- Digitally sign Java applets or other downloaded code
  - do you trust the author to write "honest" code
- Digitally sign email messages or HTML forms to validate orders

Netscape Secure Socket Layer (SSL)

- Server owner has a credential for its server name - e.g. WWW.XYZ.COM
  - issued by e.g. Verisign for a fee
  - Verisign's public key is widely known
- User invents session key and sends it encrypted in server's public key
  - Only server can decrypt session key
  - thereafter messages exchanged using session key to encrypt contents and sequence numbers etc
  - buys integrity and confidentiality, but not access control
  - therefore user sends credit card numbers or passwords to server, which isn't ideal
- User acquires digital identity, i.e., a credential
  - send session key signed with user's secret key, encrypted with server's public key
  - gives mutual authentication and no need to send credit card numbers or passwords

Non-repudiation

- In complex exchanges sign each message and include signed digest of previous messages
  - forces all parties to stay in the protocol and not miss out steps
  - archive with a trusted third party to resolve disputes
- Include sequence numbers and/or time stamps in messages to avoid replay attacks
  - how do you get synchronised global time?
  - sequence numbers and timestamps enable known plaintext / differential cryptanalysis
- Designing security protocols is hard!
- *But most security attacks are on operational procedures rather than cryptosystems or protocols per se*

Security and Transparency

- Protocols are application-specific
- Hard to do non-repudiation as a transparency, except at message level
- Can do:
  - authentication and access control
  - confidentility
  - integrity
- Some RPCs (e.g. OSF DCE) build this in
  - easy to add to Java RMI

Smartcards

- Keys are too complex to memorise
- Keeping keys on your disc isn't safe
  - Encrypt them with a pass phrase?
- Use a smartcard
  - credit card sized computer with a simple CPU and memory
  - convenient, esy to undestand
  - stores keys and does crypto algorithms
  - self-destructs if tampered with (but the bad guys are geting cleverer.....)
  - card reader device requires a PIN or biometric data to be entered

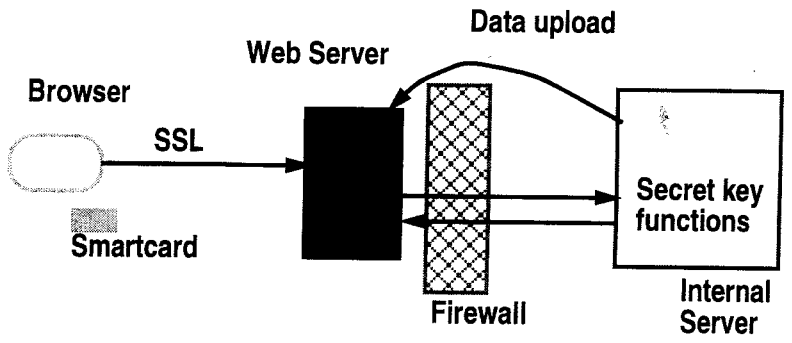
Electronic Commerce

- Using the Internet to buy and sell
  - 24M Internet users worldwide, and growing
- Customer-to-business
  - alternative to mail order / telephone order
  - n.b. TV shopping not that successful
- Business-to-business
  - low cost Electronic Data Interchange
- The big issues
  - Cryptographic policy
  - Risk management - financial, contractual
  - Cross-border transactions

The *Secure Electronic Transactions* Protocol

- Proposed by VISA and Mastercard, based on public key crypto
- Bank gives smartcard to
  - card holder - as an electronic payment card
  - merchant - as an authenticator to allow access to payment *gateways*
- Payment transaction data consists of
  - transaction id, order information (what), payment instruction (how much)
  - separately encrypted so bank can't see order details
  - dual signed - OI and PI each conatin digest of the other to stop "cut and paste" attacks
- Steps
  - get authorisation (client->merchant->payment gateway)
  - make purchase (client->merchant)
  - get payment (merchant->payment gateway)

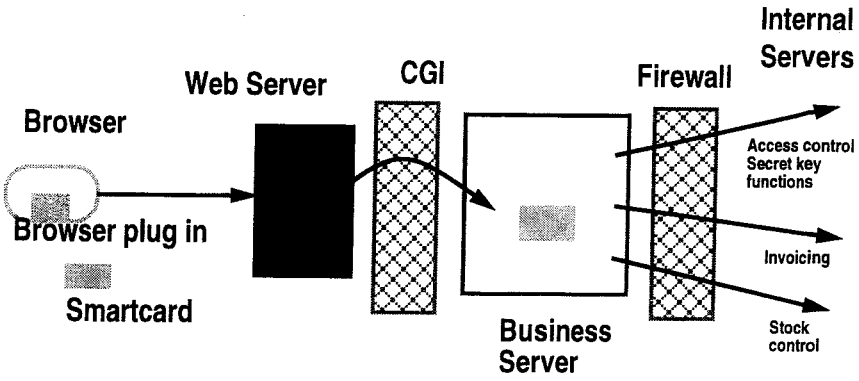
HP WCSO Value-Added Reseller Support



Security Issues

- Symmetric key distribution since
  - user population is small
  - interaction is many to one
  - access is always "on-line"
  - no need to wait for widespread public key infrastructure
- Web server is sacrificial
- SSL for confidentiality
- Sign web pages before upload to prevent corruption of web server

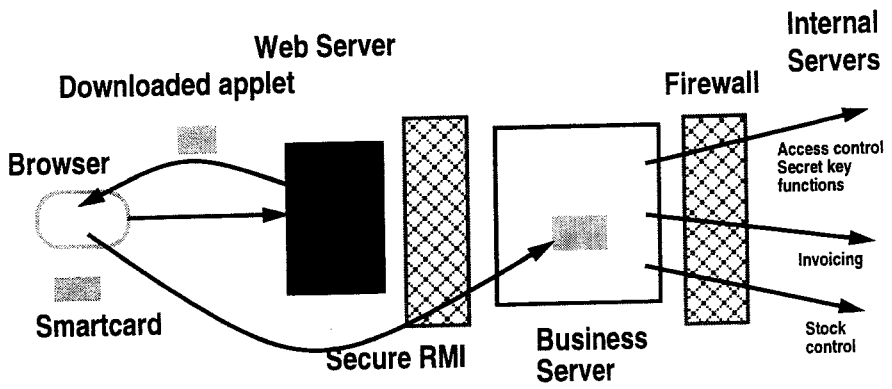
HP WCSO Repair Agent Support



Security Issues

- HP issues browser plug-in with smartcard to repair shop
- plug-in encrypts form contents (via smartcard crypto engine and keys)
  - have to trust security of customer's operating system
- Web server simply acts as a message switch
- Business server decrypts messages and processes *transactions*
- Business server insulated in "Amber zone" between Internet ("Red Zone") and Intranet ("Green Zone")
  - full access control is applied in amber zone

Alternative with Java



Security Issues

- Need to sign applet (and user has to trust us)
- no CGI overhead - simple end-to-end channel
- Interface can be customised to individual user's needs at runtime
- Have to trust security of customer's operating system
  - INTEL/ Microsoft: hardware loader which only accepts signed modules