Distributed Systems - References

ACM Symposia on Operating Systems Principles '79, 81, .. the most recent was SOSP16 in 97).
The proceedings are special issues of ACM Operating Systems Revue.
The source for reports on many research projects.

Bacon J, "Concurrent Systems" Addison Wesley 1993(ED1), 1998(ED2)
ED1: Ch 5, 14, 20, 22, 23; ED2: Ch 5, 7, 15, 21, 22, 25, 26

Coulouris G, Dollimore J and Kindberg T
" Distributed Systems, concepts and design" Addison Wesley, ED2, 1994
For general background reading

Lamport L, "Time, clocks and the ordering of events in a Distributed System"
Comm ACM 21(7) 558 - 565 July 78

Mullender S (ed), "Distributed Systems"
Addison Wesley, ACM Press, ED2 1993
For general background reading

Tanenbaum A S "Distributed Operating Systems", Prentice Hall 1995
For general background reading

Distributed Systems research at Cambridge
Browse the web pages starting from SRG for information on current grants,
recent publications, RA posts etc.

For an overview of an early distributed systems research project see:
Bacon J M, Leslie I M and Needham R M
"Distributed Computing with a Processor Bank"
Lab TR 168 and Springer Verlag LNCS 433, 1990
Gives an overview of the CDCS Cambridge Distributed Computing System
which was our research environment in the Lab throughout the 80's

Distributed Systems:Study Guide

for Part2, Part 2G and Diploma Easter term 1999

Jean Bacon (jmb@cl.cam.ac.uk)

Desirable prerequisites:
Concurrent Systems or OS Foundations, OS Functions,
Digital Communications, Security

For background reading browse the references, but note that
books entirely devoted to DS will have too much detail
for an 8 lecture course.
The approach taken in the course can be found in parts of
Concurrent Systems ED2. I can put the text of these chapters
on the web - please let me know if that would be useful.

Ch 2 objects/platforms
Ch 5 DS fundamentals
Ch 7 sections on distributed filing systems (ED1 Ch 5)
Ch 15 distributed IPC, RPC (ED1 Ch 14)
Ch 21 distributed transactions (ED1 Ch 20)
Ch 22 distributed algorithms
Ch 26 Middleware

Past exam questions are probably the best revision exercises,
although the course has evolved over the years.
There are also exercises at the end of most chapters of CS
with solutions in the Library and on the web.

The notes start with the most important OHP foil -
fundamental properties of DS.
Keep these in mind for all the topics.
It's relatively easy to devise algorithms and protocols
if you abstract out failures and uncertainty over time.

The notes are numbered within the topics on the cover sheet.

I have included some information on some current research
projects in the Lab.
You can find out more by browsing the Lab web pages.

DISTRIBUTED SYSTEMS

FUNDAMENTAL PROPERTIES

1. concurrent execution of components

2. independent failure modes
   components and connections may fail independently

3. transmission delay

4. relativistic time

IMPLICATIONS

cannot know why there's no reply - failure or congestion

cannot use local timestamps to order the occurrence of events

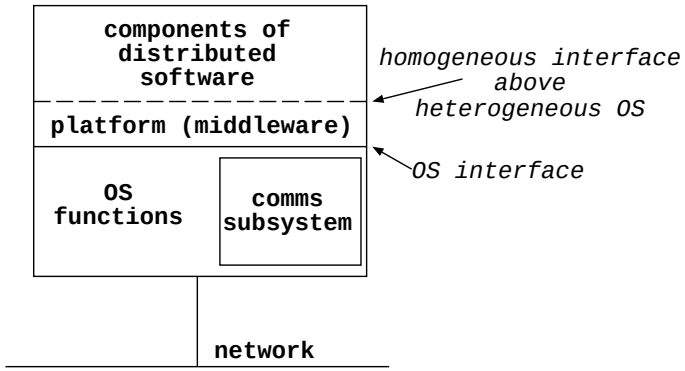
inconsistent state
- replicas of a single object
- related changes to different objects

DISTRIBUTED SOFTWARE SYSTEMS

* what is a "conventional" system?

* previous courses have covered general principles of operating systems and communications

* we now look explicitly at support for distributed software which may be:
  - directly by OS in a homogeneous cluster
  - by a platform (middleware) above heterogeneous OS.



DISTRIBUTED SYSTEMS POTENTIAL
(from about 1985)

* **cost effectiveness**
  many small machines cf. few large ones

* **high performance**
  real concurrency cf. timesharing - better response
  distributed algorithms to exploit concurrency

* **availability**
  if one processor fails the others can
  continue.....MAYBE

* **extensibility**
  - incremental growth rather than replacement
  - design of a range of configurations

* **simplicity due to physical modularity**
  - may have dedicated processors in some
    applications e.g. process control

DISTRIBUTED SYSTEMS, EXAMPLES

* **process control**
  manufacturing,

* **automation of commerce**
  stock control,invoicing, orders, sales

* **utilities**
  billing, customer interaction,
  fault reporting and repair

* **public bureaucracies**
  health service, tax, social services

* **financial systems**
  global transaction processing

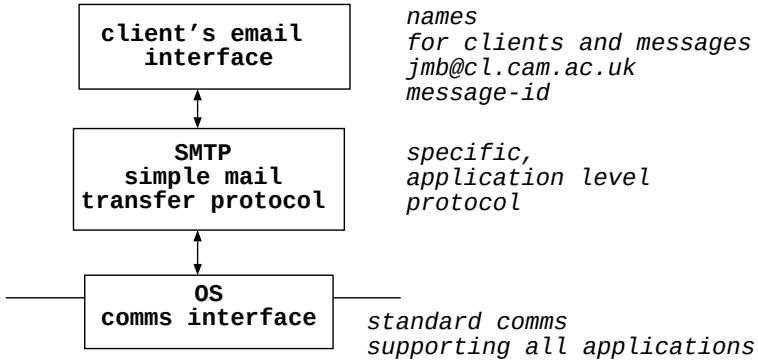
* **office systems**
  document storage, provision of applications,
  groupware,.....

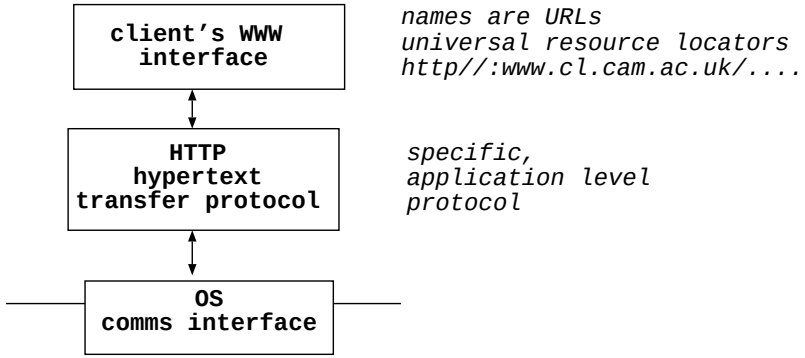
* **distributed database systems**

* **mobile users and computers**

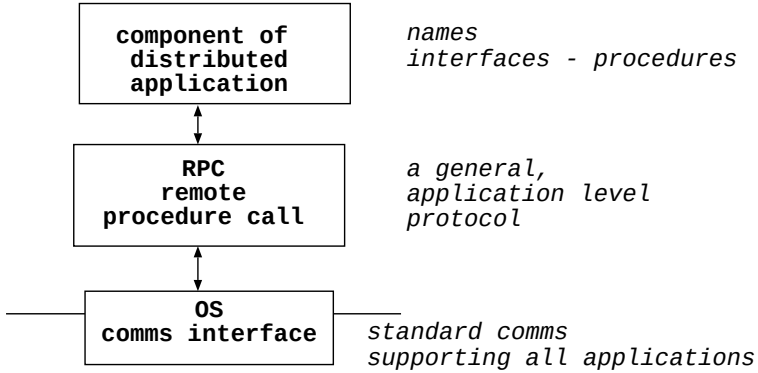
SPECIFIC DISTRIBUTED APPLICATIONS

e.g. email, news, ftp, WWW





GENERIC SUPPORT for DISTRIBUTED APPLICATIONS



RPC is an example of a general, application-level protocol above which distributed applications may be developed.

A middleware platform will contain communications support which may be RPC and/or message passing.

Examples are ISO-ODP, OSF-DCE, OMG-CORBA, DCOM, Java-RMI

As we are beginning to see from the above, NAMES must be defined system-wide and platforms must support name-to-location mapping.

OPEN & PROPRIETARY SOFTWARE

* **Open software**

  evolution is controlled by standards bodies such as ISO or by consortia such as OMG (Object Management Group).

  RFPs (requests for proposals) are issued and draft specifications are published with RFCs (requests for comments)

  a "standard" may be published without implementations

* **proprietary software**

  can be changed at the whim of the owner

  consistency of functionality across versions is not guaranteed

RELATED ISSUES

* **single-language/multi-language?**

  can components be written in different languages and interoperate?

* **open interoperability**

  - is desirable across middleware platforms as well as programming languages

CLUSTER COMPUTING

* **small-scale, homogeneous, single-language**

  used to support parallel algorithms
  (e.g. nuclear physics)

  focus on resource-sharing, load balancing

  may be based on a distributed shared-memory (DSM) abstraction

ASPECTS OF DISTRIBUTED SYSTEM DESIGN

MODEL of distributed computation

what are the named, communicating objects?
objects,
clients and servers,
components

ARCHITECTURE - issues

naming
location of named objects
security of communication
protection (access control)
reliability
heterogeneous/homogeneous

placement of functionality, coarse-grained

ENGINEERING issues

transparency of distribution
- location transparency
- failure transparency
- migration transparency

(transparency =
can't be detected when running
needn't be programmed for explicity)
(may not be achievable in practice)
(which system level should provide
which transparency?)

placement of functionality, fine-grained

use of replication - achieving consistency

DISTRIBUTED SYSTEMS ARCHITECTURAL EVOLUTION

PROFESSIONAL COMPUTING

Distribution of function between clients and servers:

- terminals + multiaccess systems
- terminals + pool of processors + servers (e.g. Cambridge)
- diskless workstations + servers (e.g. Stanford)
- workstations + servers (e.g. Xerox PARC)

→ cluster-computing

→ large-scale, open interoperability

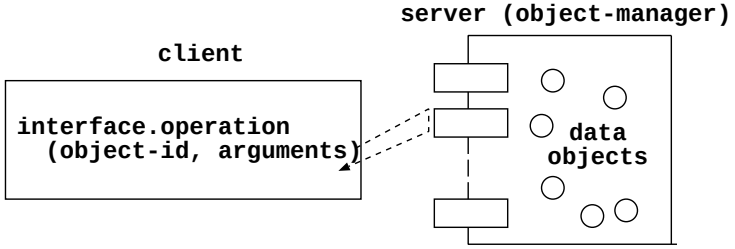
→ mobile users and computers

PERSONAL & UBIQUITOUS COMPUTING

- standalone PCs
- networked PCs
- network computer
  diskless + storage servers + processor-pool
- laptops, palmtops, mobile devices

Client-Server model

naming: server interfaces and their operations



* everyone starts with the ability to invoke the name service (may be called interface trader)
* servers export their interfaces to the name service which supports name-to-location mapping for clients
* clients import interfaces from the name service
* clients invoke operations at servers

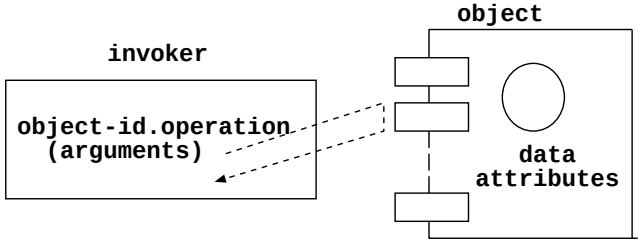
---

* a service may be provided by a number of servers
* a server may be an object manager
  - the objects may be persistent
* a server may be a client of other servers

  e.g. client -> TPS -> File-service

Object model

NAMING: objects are named application/system-wide



* typed objects may be registered with the platform which returns a globally valid id

* object id's may be used as first-class values

* on invocation, the platform provides name-to-location mapping

---

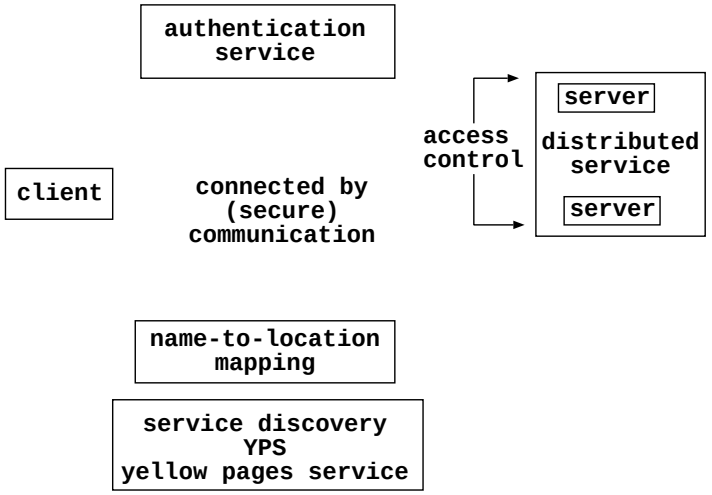
* the platform may support only remote invocation
  e.g. OMG CORBA

* the system may map objects into the invoker's address-space if a DSM (distributed shared memory) model is assumed.

  Appropriate only for homogeneous clusters of computers/OSs and a single-language system.

  e.g. Eden, Emerald, Somiw-SOS, Cool, Clouds, Comandos

DISTRIBUTED SYSTEMS ARCHITECTURAL COMPONENTS



authentication - client to server
server to client

access control - applied at server

naming - defined by system model

name-to-location binding
may be provided at service or platform level