1

System types

- Personal systems that are designed to run on a personal computer or workstation
- Distributed systems where the system software runs on a loosely integrated group of cooperating processors linked by a network

2

Distributed systems

- Virtually all large computer-based systems are now distributed systems
- Information processing is distributed over several computers rather than confined to a single machine
- Distributed software engineering is now very important

Distributed system characteristics

3

- Resource sharing
- Concurrency
- Scalability
- Fault tolerance
- Transparency

Distributed system disadvantages

4

- Complexity
- Security
- Manageability
- Unpredictability

Distributed Systems Architectures

- Architectural design for software that executes on more than one processor

Issues in distributed system design

Design issue	Description
Resource identification	The resources in a distributed system are spread across different computers and a naming scheme has to be devised so that users can discover and refer to the resources that they need. An example of such a naming scheme is the URL (Uniform Resource Locator) that is used to identify WWW pages. If a meaningful and universally understood identification scheme is not used then many of these resources will be inaccessible to system users.
Communications	The universal availability of the Internet and the efficient implementation of Internet TCP/IP communication protocols means that, for most distributed systems, these are the most effective way for the computers to communicate. However, where there are specific requirements for performance, reliability etc. alternative approaches to communications may be used.
Quality of service	The quality of service offered by a system reflects its performance, availability and reliability. It is affected by a number of factors such as the allocation of processes to processors in the system, the distribution of resources across the system, the network and the system hardware and the adaptability of the system.
Software architectures	The software architecture describes how the application functionality is distributed over a number of logical components and how these components are distributed across processors. Choosing the right architecture for an application is essential to achieve the desired quality of service.

7

Topics covered

- Multiprocessor architectures
- Client-server architectures
- Distributed object architectures
- CORBA

8

Distributed systems architectures

- Multiprocessor architectures
  - System composed of multiple processes that may execute on different processors
- Client-server architectures
  - Distributed services which are called on by clients. Servers that provide services are treated differently from clients that use services
- Distributed object architectures
  - No distinction between clients and servers. Any object on the system may provide and use services from other objects

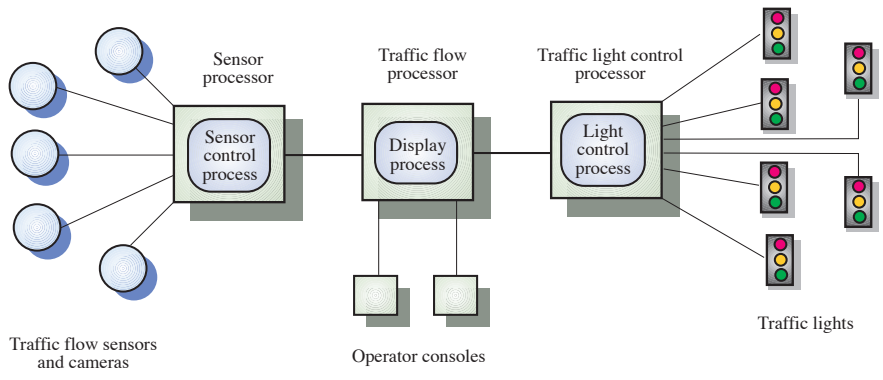
9

Multiprocessor architectures

- Simplest distributed system model
- System composed of multiple processes which may execute on different processors
- Architectural model of many large real-time systems
- Distribution of process to processor may be pre-ordered or may be under the control of a dispatcher

10

A multiprocessor traffic control system



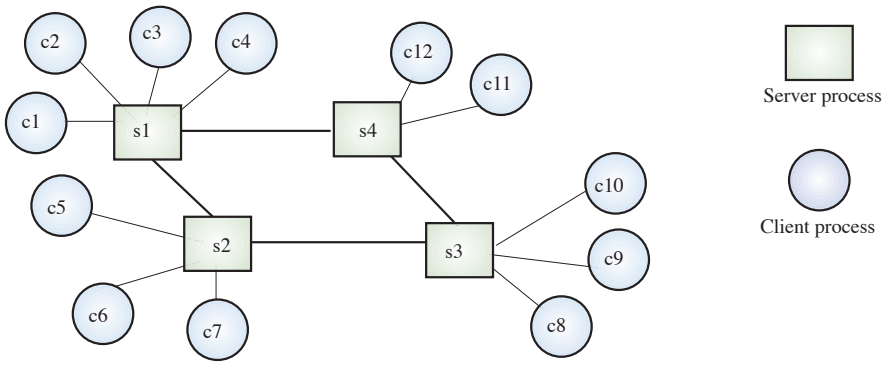
11

Client-server architectures

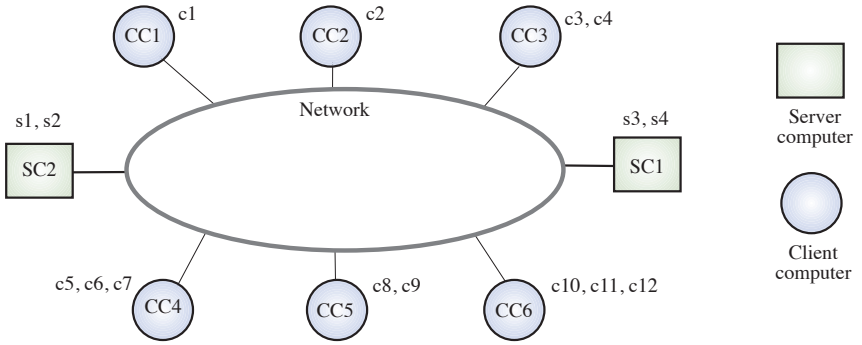
- The application is modelled as a set of services that are provided by servers and a set of clients that use these services
- Clients know of servers but servers need not know of clients
- Clients and servers are logical processes
- The mapping of processors to processes is not necessarily 1 : 1

12

A client-server system



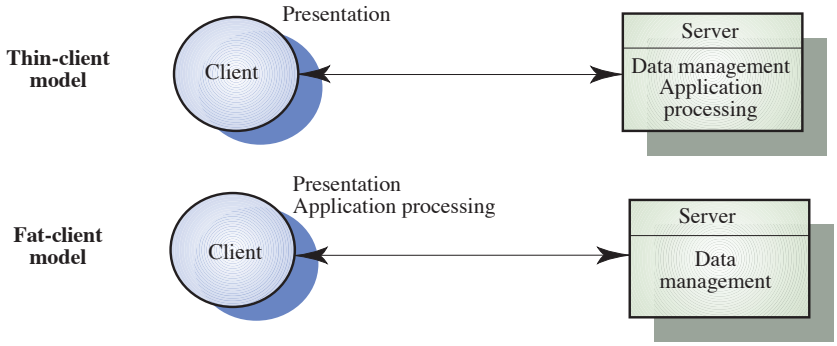
Computers in a C/S network



Thin and fat clients

- *Thin-client model*
  - In a thin-client model, all of the application processing and data management is carried out on the server. The client is simply responsible for running the presentation software.
- *Fat-client model*
  - In this model, the server is only responsible for data management. The software on the client implements the application logic and the interactions with the system user.

Thin and fat clients



Thin client model

- Used when legacy systems are migrated to client server architectures.
  - The legacy system acts as a server in its own right with a graphical interface implemented on a client
- A major disadvantage is that it places a heavy processing load on both the server and the network

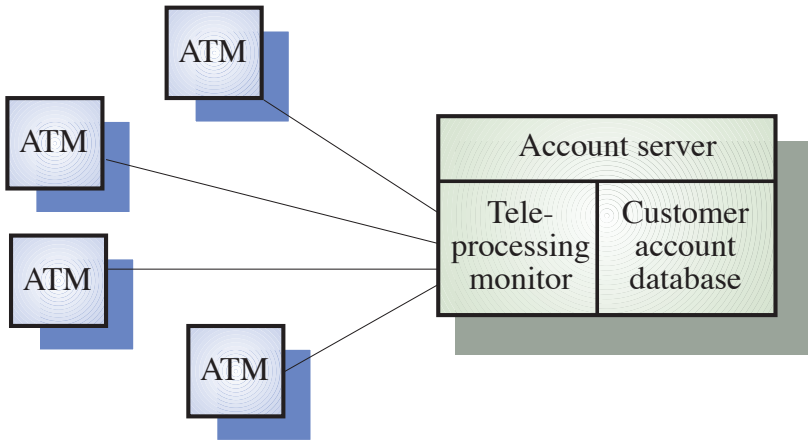
17

Fat client model

- More processing is delegated to the client as the application processing is locally executed
- Most suitable for new C/S systems where the capabilities of the client system are known in advance
- More complex than a thin client model especially for management. New versions of the application have to be installed on all clients

18

A client-server ATM system



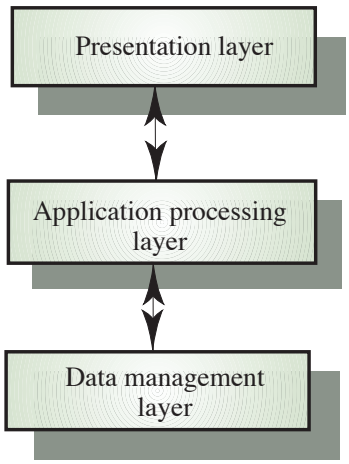
19

Layered application architecture

- Presentation layer
  - Concerned with presenting the results of a computation to system users and with collecting user inputs
- Application processing layer
  - Concerned with providing application specific functionality e.g., in a banking system, banking functions such as open account, close account, etc.
- Data management layer
  - Concerned with managing the system databases

20

Application layers



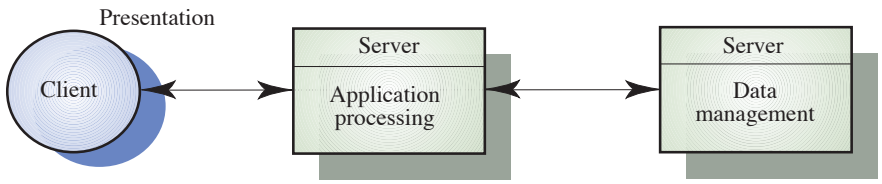
21

Three-tier architectures

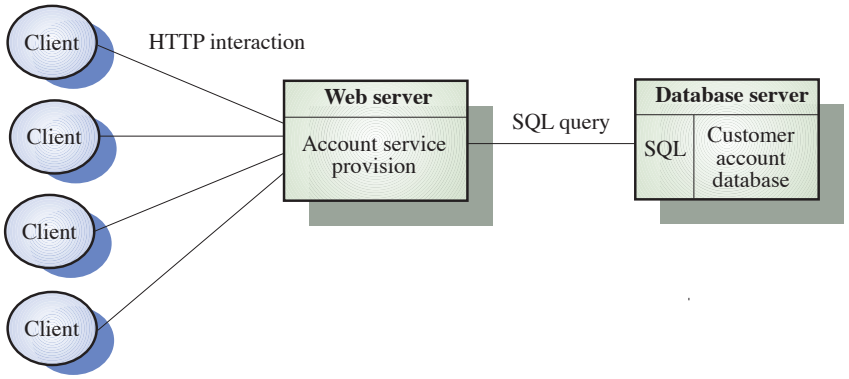
- In a three-tier architecture, each of the application architecture layers may execute on a separate processor
- Allows for better performance than a thin-client approach and is simpler to manage than a fat-client approach
- A more scalable architecture - as demands increase, extra servers can be added

22

A 3-tier C/S architecture



An internet banking system



Use of C/S architectures

Architecture	Applications
Two-tier C/S architecture with thin clients	Legacy system applications where separating application processing and data management is impractical Computationally-intensive applications such as compilers with little or no data management Data-intensive applications (browsing and querying) with little or no application processing.
Two-tier C/S architecture with fat clients	Applications where application processing is provided by COTS (e.g. Microsoft Excel) on the client Applications where computationally-intensive processing of data (e.g. data visualisation) is required. Applications with relatively stable end-user functionality used in an environment with well-established system management
Three-tier or multi-tier C/S architecture	Large scale applications with hundreds or thousands of clients Applications where both the data and the application are volatile. Applications where data from multiple sources are integrated

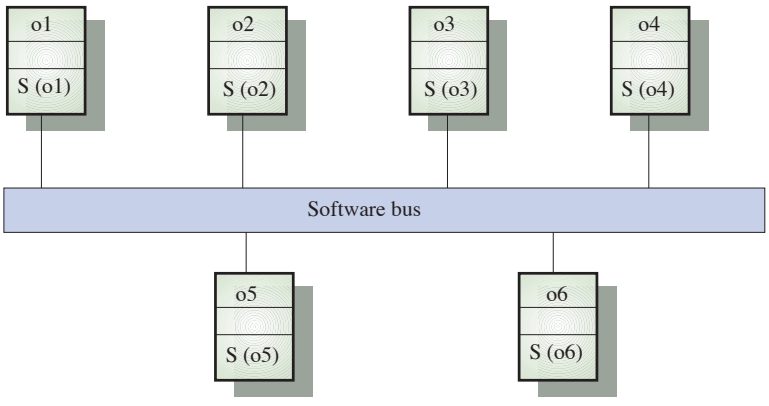
25

Distributed object architectures

- There is no distinction in a distributed object architectures between clients and servers
- Each distributable entity is an object that provides services to other objects and receives services from other objects
- Object communication is through a middleware system called an object request broker (software bus)
- However, more complex to design than C/S systems

26

Distributed object architecture



Advantages of distributed object architecture

27

- It allows the system designer to delay decisions on where and how services should be provided
- It is a very flexible and scaleable system architecture that allows new resources to be added to it as required
- It is possible to reconfigure the system dynamically with objects migrating across the network as required

Uses of distributed object architecture

28

- As a logical model that allows you to structure and organize the system. In this case, you think about how to provide application functionality solely in terms of services and combinations of services
- As a flexible approach to the implementation of client-server systems. The logical model of the system is a client-server model but both clients and servers are realized as distributed objects communicating through a software bus

29

Middleware

- Software that manages and supports the different components of a distributed system. In essence, it sits in the *middle* of the system
- Middleware is usually off-the-shelf rather than specially written software

30

CORBA

- Common Object Request Broker Architecture (CORBA) is an international standard for an Object Request Broker - middleware to manage communications between distributed objects
- Several implementation of CORBA are available
- Distributed Component Object Model (DCOM) is an alternative approach by Microsoft to object request brokers
- CORBA has been defined by the Object Management Group (OMG)

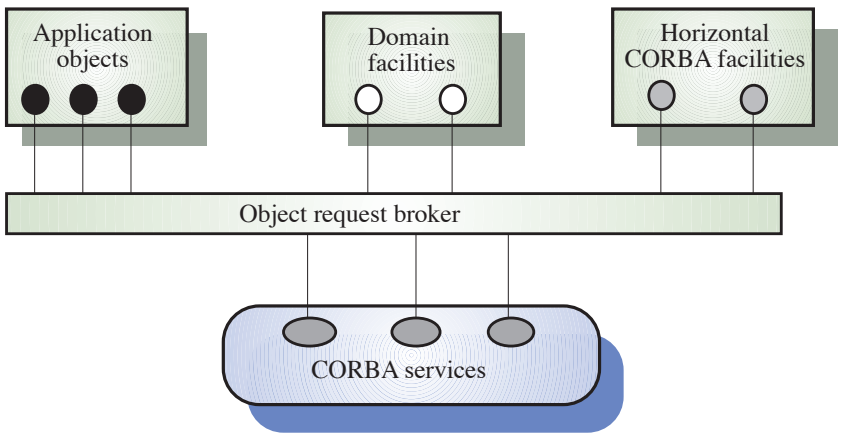
31

Application structure

- Application objects
- Standard objects, defined by the OMG, for a specific domain e.g. insurance
- Fundamental CORBA services such as directories and security management
- Horizontal (i.e. cutting across applications) facilities such as user interface facilities

32

CORBA application structure



33

CORBA standards

- An object model for application objects
  - A CORBA object is an encapsulation of state with a well-defined, language-neutral interface defined in an IDL (interface definition language)
- An object request broker that manages requests for object services
- A set of general object services of use to many distributed applications

34

CORBA objects

- CORBA objects are comparable, in principle, to objects in C++ and Java
- They MUST have a separate interface definition that is expressed using a common language (IDL) similar to C++
- There is a mapping from this IDL to programming languages (C++, Java, etc.)
- Therefore, objects written in different languages can communicate with each other

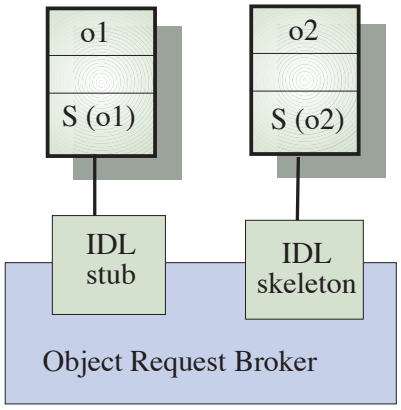
35

Object request broker (ORB)

- The ORB handles object communications. It knows of all objects in the system and their interfaces
- Using an ORB, the calling object binds an IDL stub that defines the interface of the called object
- Calling this stub results in calls to the ORB which then calls the required object through a published IDL skeleton (links the interface to the service implementation)

36

ORB-based object communications



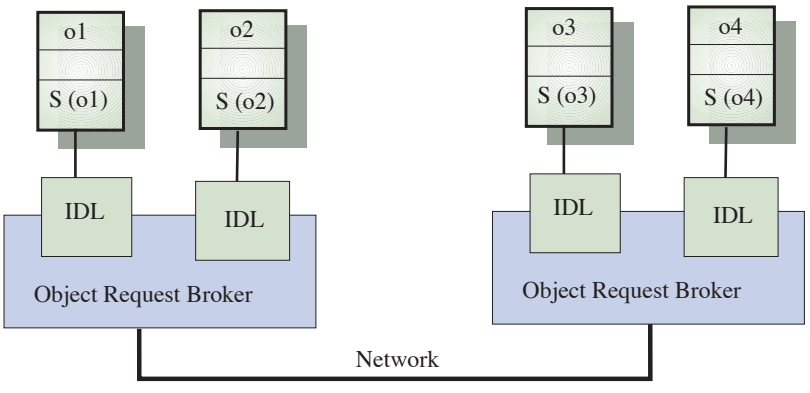
37

Inter-ORB communications

- ORBs handle communications between objects executing on the same machine
- Several ORBS may be available and each computer in a distributed system will have its own ORB
- Inter-ORB communications are used for distributed object calls

38

Inter-ORB communications



39

CORBA services

- Naming and trading services
  - These allow objects to discover and refer to other objects on the network
- Notification services
  - These allow objects to notify other objects that an event has occurred
- Transaction services
  - These support atomic transactions and rollback on failure

40

Additional Resources on CORBA

- http://www.corba.org/
- CORBA Success Stories
  http://www.corba.org/success.htm
  - The Weather Channel (TWC) used CORBA and Linux to develop a system that provides reliability, doesn't require a high level of operational support and offers the ability to have detailed data logging. They were able to meet these needs and slash their software development time from months to weeks.
  - http://www.corba.org/industries/publish/twc.htm