Lecture 14 - Distributed Systems (Sommerville Ch. 11)

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk

Objectives

- To explain the advantages and disadvantages of distributed systems architectures
- To describe different approaches to the development of client-server systems

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

System types

- Personal systems that are not distributed and that are designed to run on a personal computer or workstation.
- Embedded systems that run on a single processor or on an integrated group of processors.
- Distributed systems where the system software runs on a loosely integrated group of cooperating processors linked by a network.

Distributed system characteristics

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

Distributed system disadvantages

- Complexity
- Security
- Manageability
- Unpredictability

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 processes 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.

Issues in distributed system design

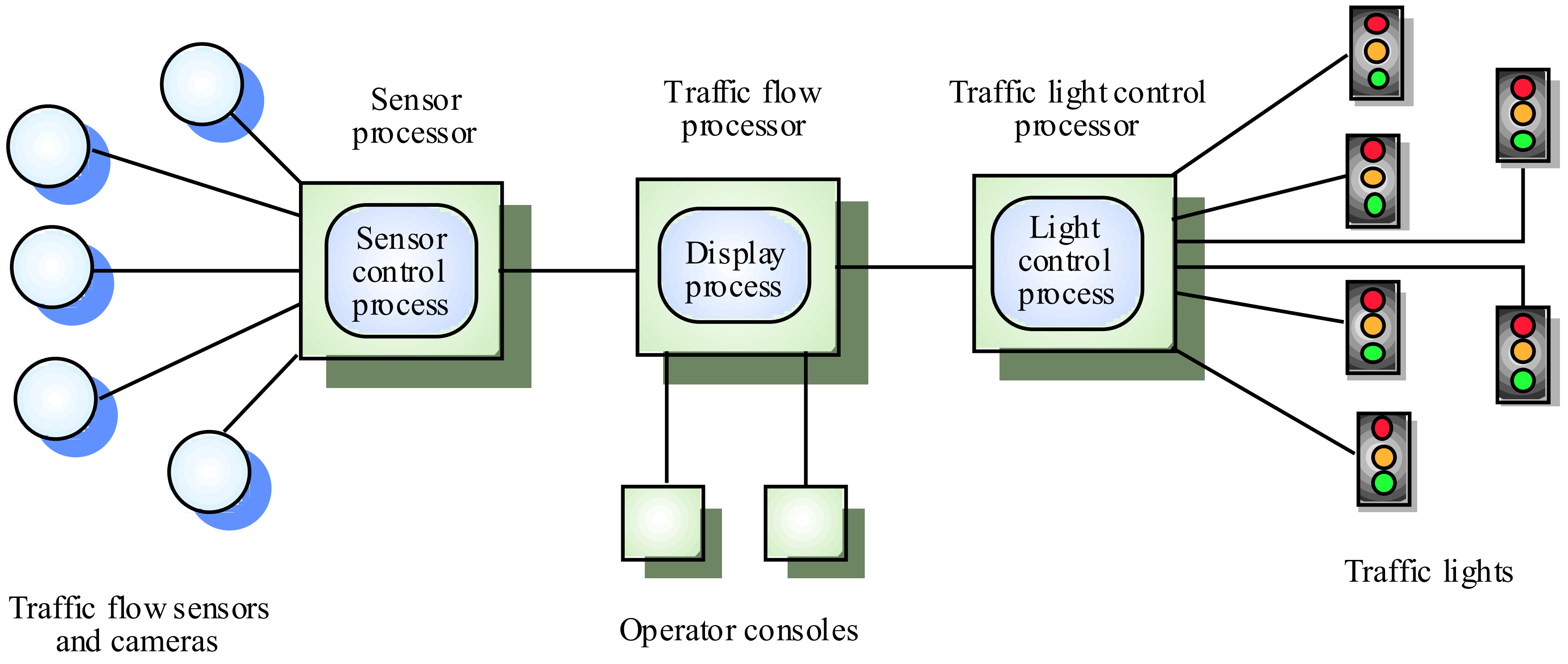
Distributed systems archiectures

- 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

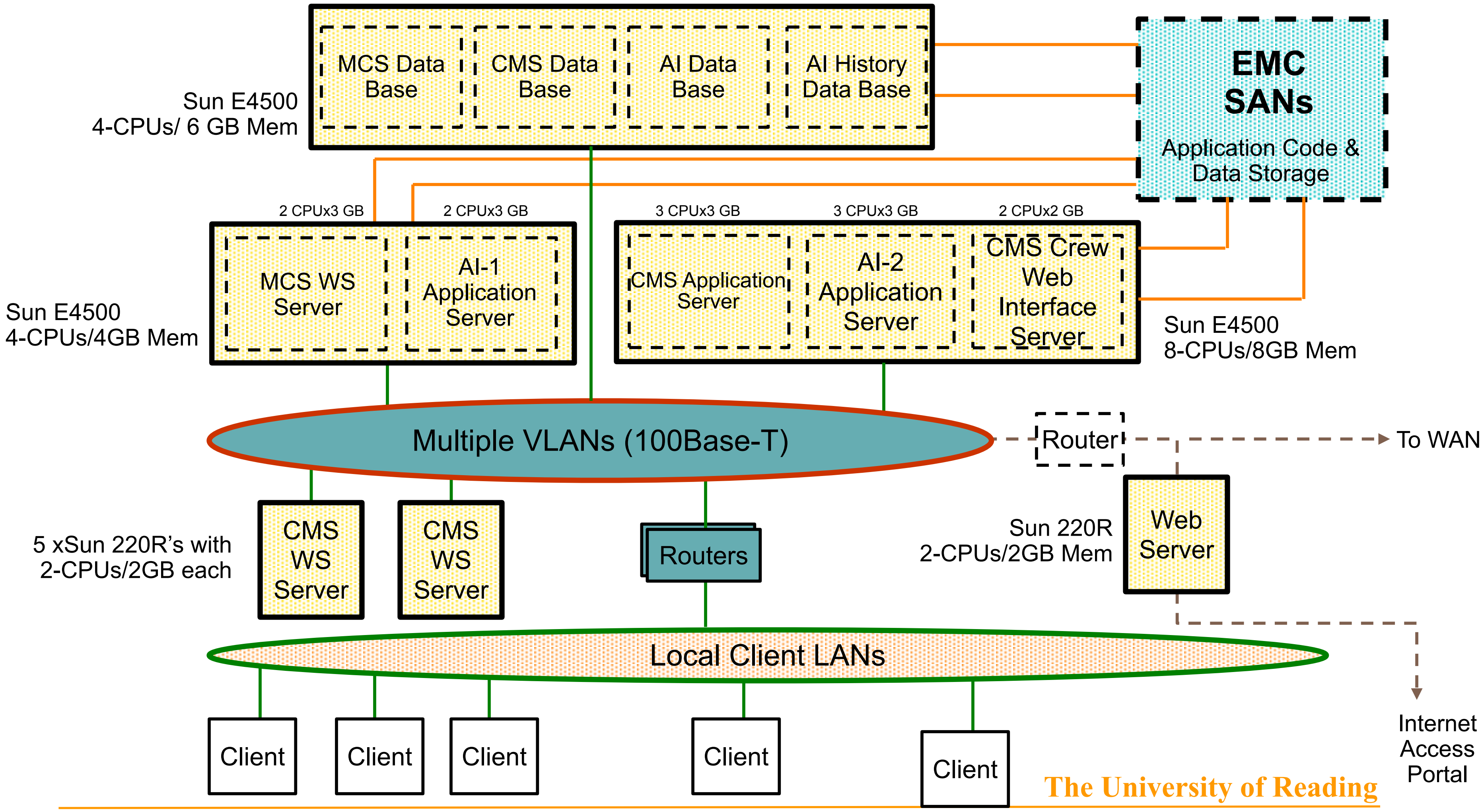
Multiprocessor architectures

- Simplest distributed system model
- System composed of multiple processes which may (but need not) 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 despatcher

A multiprocessor traffic control system



Example Client Server Architecture



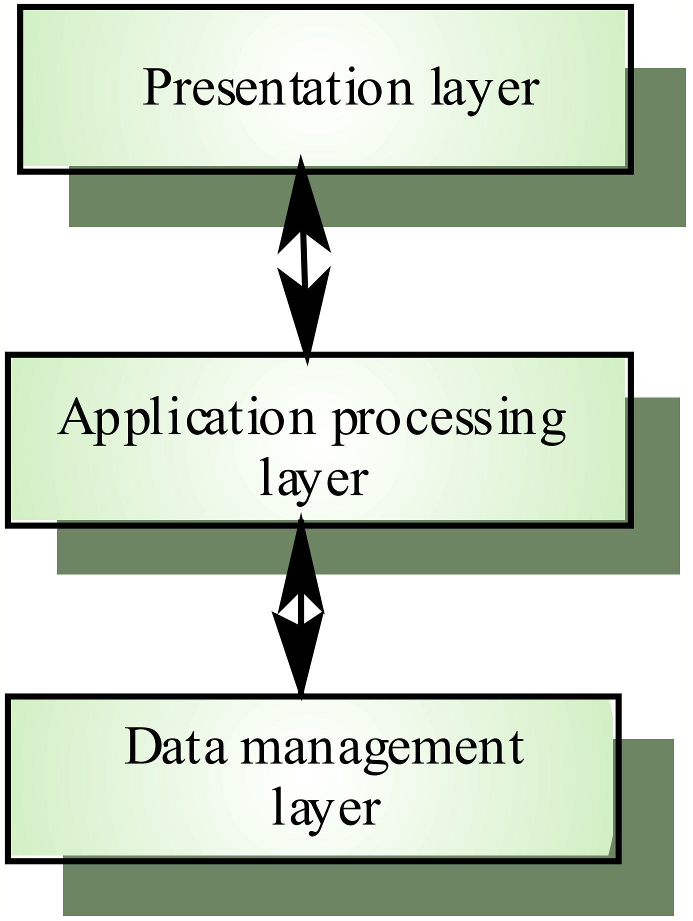
Client-server architectures

- 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

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

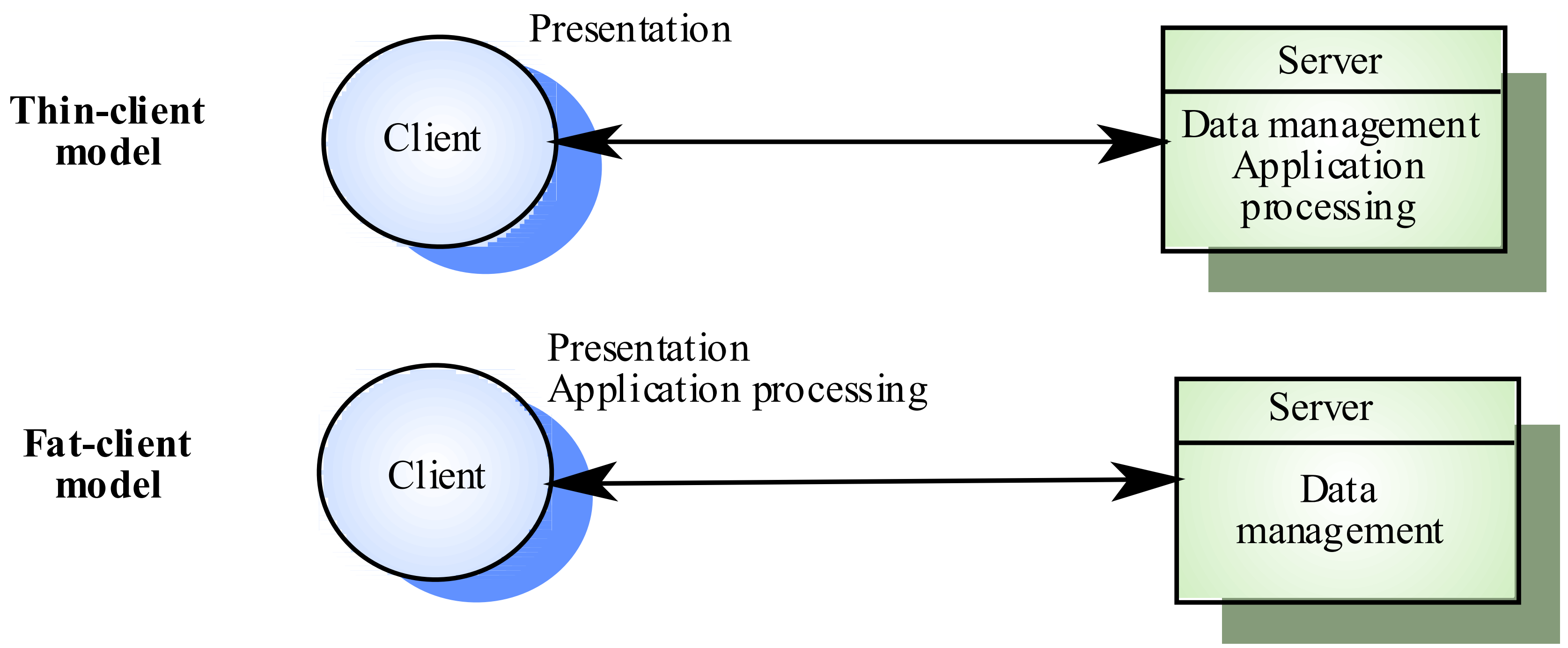
Application layers



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

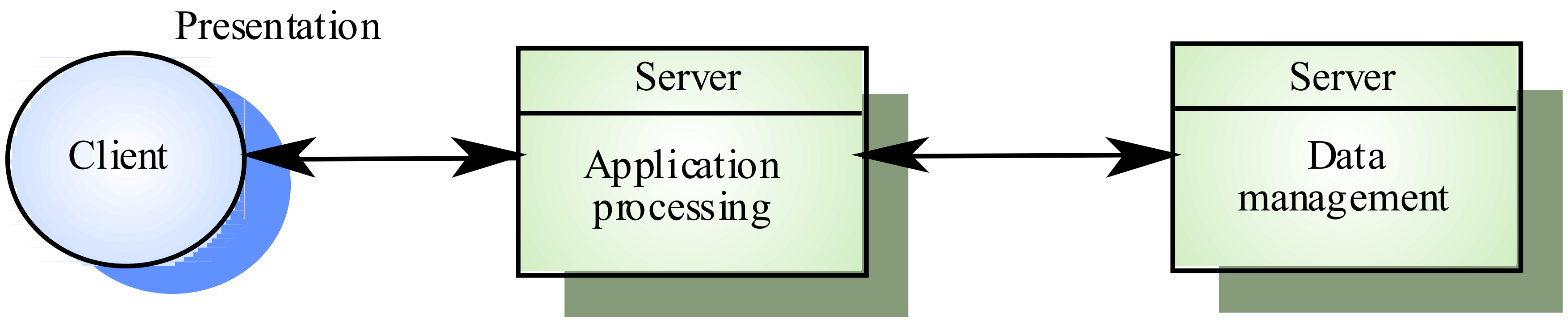
Fat client model

- 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

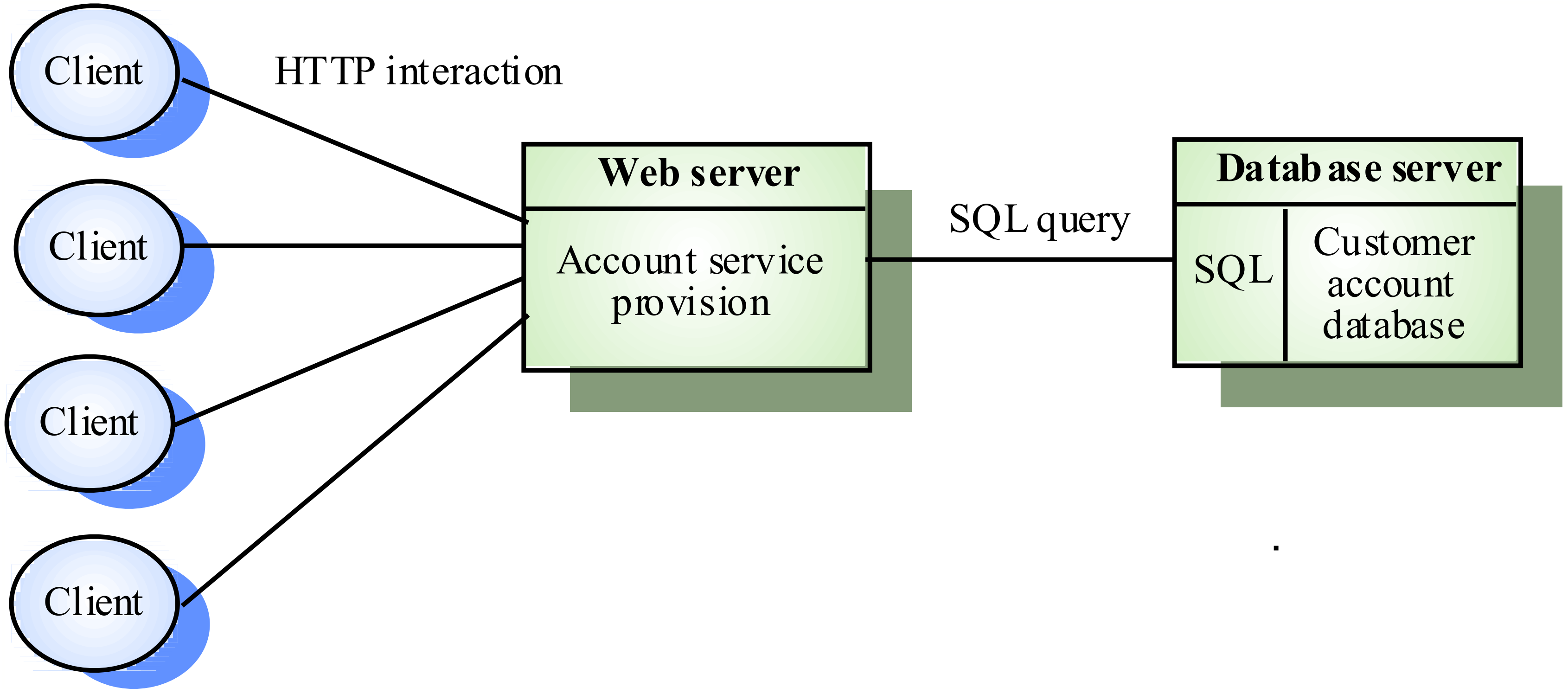
Three-tier architectures

- 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

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

Key points

- Almost all new large systems are distributed systems
- Distributed systems support resource sharing, openness, concurrency, scalability, fault tolerance and transparency
- Client-server architectures involve services being delivered by servers to programs operating on clients
- User interface software always runs on the client and data management on the server