The UNIVERSITY *of* EDINBURGH

SCHOOL *of* INFORMATICS

CS4/MSc

Distributed Systems

Björn Franke

bfranke@inf.ed.ac.uk

Room 2414

(Lecture 1: Introduction, 21st September 2006)

What is a distributed system?

A distributed system is one in which I can be prevented from getting on with my work because a computer I have never heard of has crashed....

A distributed system is broadly categorised as a collection or network of loosely coupled, autonomous computers with the following characteristics:

- The nodes are relatively loosely coupled.
- Each node is a self-contained autonomous computer with its own peripherals.
- The system can survive various categories of node and network failures.
- The nodes may execute logically separate computations, though these may be related to concurrent computations on other nodes.
- The system is asynchronous.

Overview of the course

Distributed Computation: characteristics of distributed systems, overview of related concepts (networks, operating systems, etc.)

Interprocess Communication: message passing, client/server communication model, remote procedure call (RPC), atomic transactions

Middleware: introduction to distributed object technologies such as Common Object Request Broker Architecture(CORBA), Java Remote Method Invocation (RMI) and Distributed Common Object Model (DCOM).

Distributed Coordination: physical and logical clocks, synchronisation, mutual exclusion, leader election, deadlock detection

Failure and Fault Tolerance: failure models, safety and liveness properties, distributed agreement in the presence of failures

Coherence: sharing of distributed data objects and memory, replication, techniques for managing replicated data

Emerging Technologies: mobile code and mobile agents, peer-to-peer systems and ubiquitous networking.

What's not in the course

Some topics could be included in detail but will only be touched upon because of overlap with other modules

- protocols and network design (Computer Communications)
- formal models of concurrent processes
  (Communication and Concurrency)
- encryption and security measures (Computer Security)
- distributed operating systems (Operating Systems)
- performance issues (Modelling and Simulation)

Distributed System Issues

Naming and Discovery — creation and management of names, finding the "right" resource

Replication — allowing some activities to take place more locally can improve performance but introduces issues of consistency

Concurrency — components proceed asynchronously but cooperation and coordination are necessary to achieve "complete" result

Failure and Recovery — components may fail independently, failures (and recoveries) may go undetected

Time — the lack of a global clock, local clocks may disagree and the problem is exacerbated by unreliable network communication.

Practicalities

Lectures There are two lectures per week, on Mondays and Thursdays at 11:10am. Lectures take place in the Swann Building, R. 715, & AB LT 3.

Notes Lecture notes will not be provided but copies of the (detailed) slides will be handed out for each lecture.

Text *Distributed Systems: Concepts and Design* by Coulouris, Dollimore and Kindberg, **4th** edition.

Practicals There will be two practicals and together they will contribute 25% to the course mark. Both will involve writing Java programs. The first will be based on CORBA. The second will focus on algorithms for distributed agreement.

There are no tutorials for the course but the lecturer will give assistance by appointment.

Basic notions

RESOURCE *something that can be usefully shared in a networked computer system* e.g. data, hardware, application,...

SERVICE part of a computer system which manages a collection of related resources and presents their functionality to users and applications; access is restricted to a well-defined set of *operations*.

SERVER a program or process on a networked computer which provides a service.

CLIENT a program or process generating a request to a server (*invocation*); requests are sent in messages from the client to the server and replies are sent in messages from the server to the client.

Note that *client* and *server* apply only to the roles played within a single invocation.

PEER PROCESSES processes that cooperate and communicate in a symmetrical manner to perform a task.

Challenges

A number of challenges are recognised in the design of distributed systems, and these have been tackled with varying degrees of success in existing systems. In some cases they place conflicting requirements on the system. The list below gives some indication of measures that are employed to meet each challenge:

- Heterogeneity—*standards and protocols; middleware; virtual machine;*
- Openness—*publication of services; notification of interfaces; independence from individual vendors;*
- Security—*firewalls; encryption*
- Scalability—*replication; caching; multiple servers;*
- Failure Handling—*failure tolerance; recover/roll-back; redundancy;*
- Concurrency—*concurrency control to ensure data consistency;*
- Transparency—*middleware; location transparent naming; anonymity*

ISO Reference Model for Open Distributed Processing

Aims to conceal the component-based structure of the system, and facilitate a perception of the system as a whole:

- Access Transparency
- Location Transparency
- Concurrency Transparency
- Replication Transparency
- Failure Transparency
- Mobility/Migration Transparency
- Performance Transparency
- Scaling Transparency

Access and location transparency together are considered to provide *network transparency.*

Architectural Models

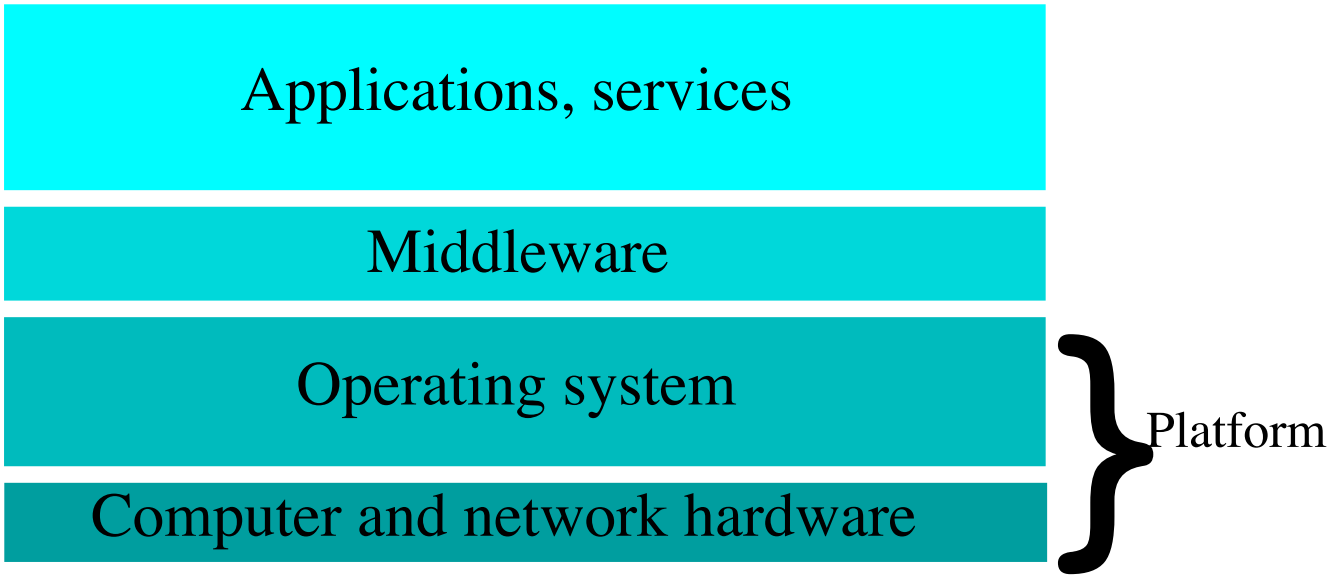
Architectural models are used to show the possible ways in which components of a distributed system may interact. They are useful for highlighting:

- the placement of components across a network of computers, providing useful patterns for the distribution of data and workload.
- the inter-relationships between the components, outlining their functional roles and the patterns of communication between them.

In the following slides we examine several such models.

Service Layers in Distributed Systems

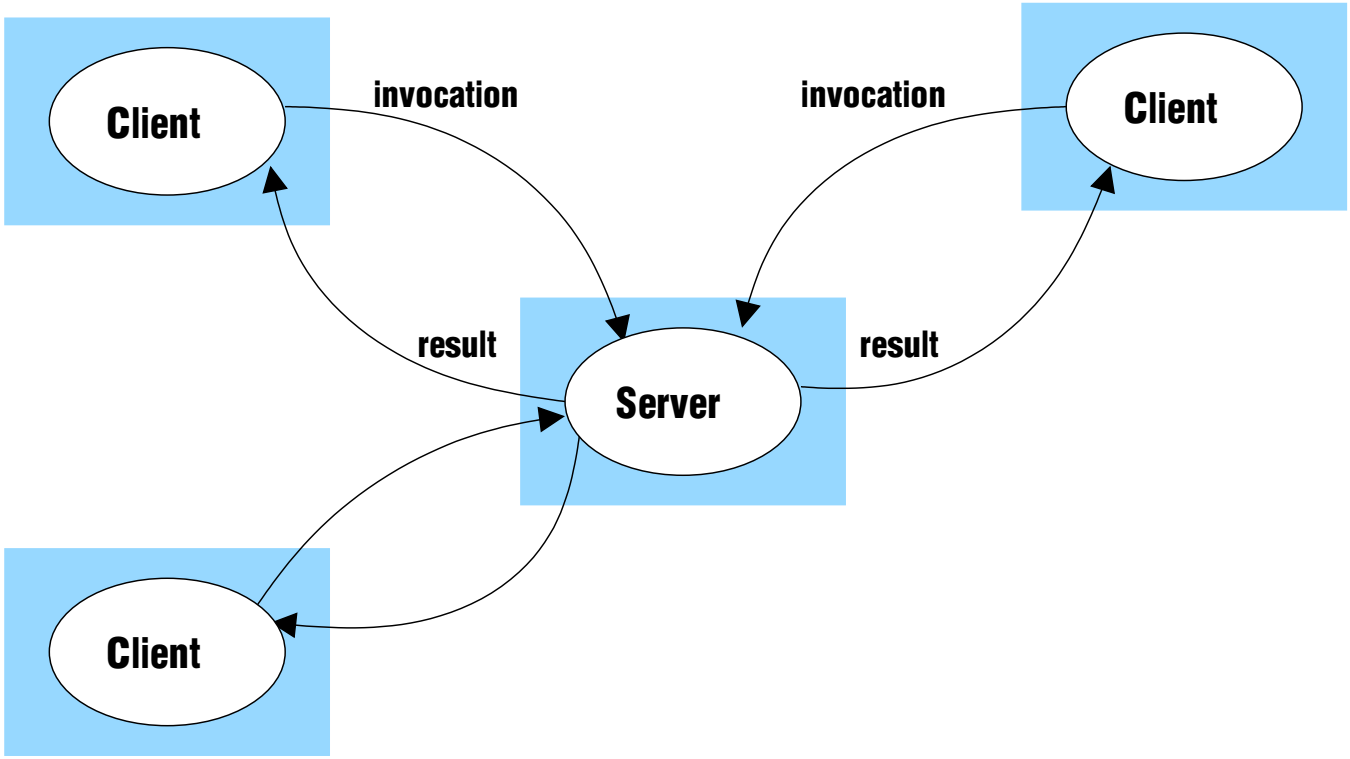
In the layered view of a system each layer offers services to the level above and builds its own service on the services of the layer below.



The operating system and the hardware together are often referred to as the *platform*. The *middleware* layer is intended to hide heterogeneity and provide a useful abstraction for application programmers.

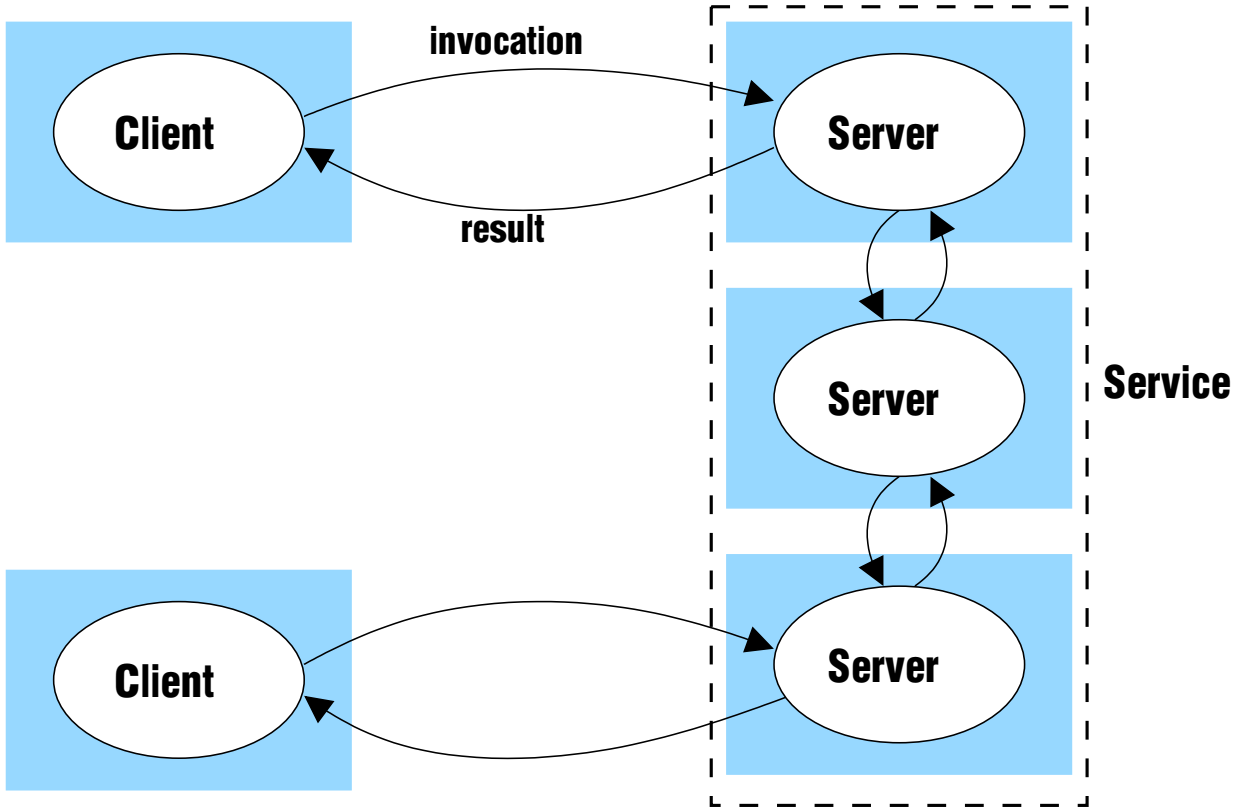
Client-Server Model (Simple Case)

In the simplest case clients invoke single servers which satisfy their requests individually:



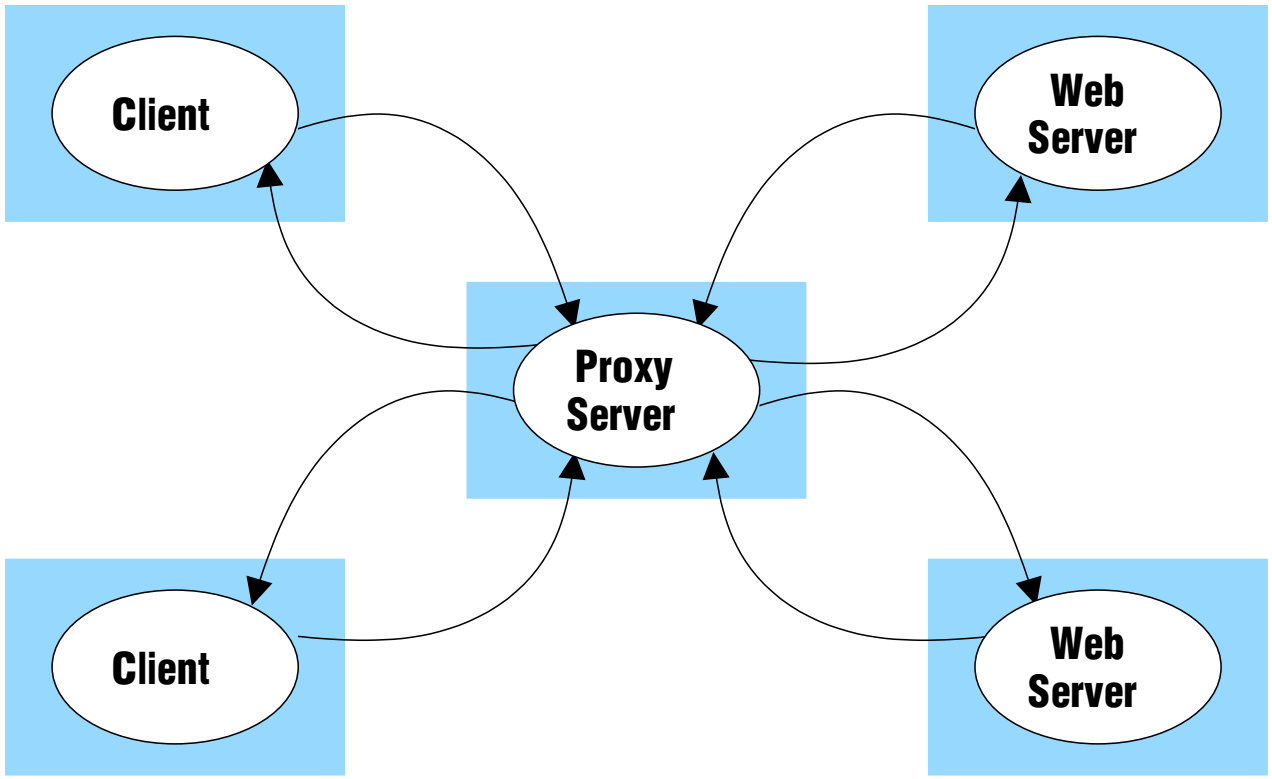
Alternative Client-Server Model

A more general case allows service to be provided by multiple servers: servers on distinct hosts interact as appropriate to provide service.



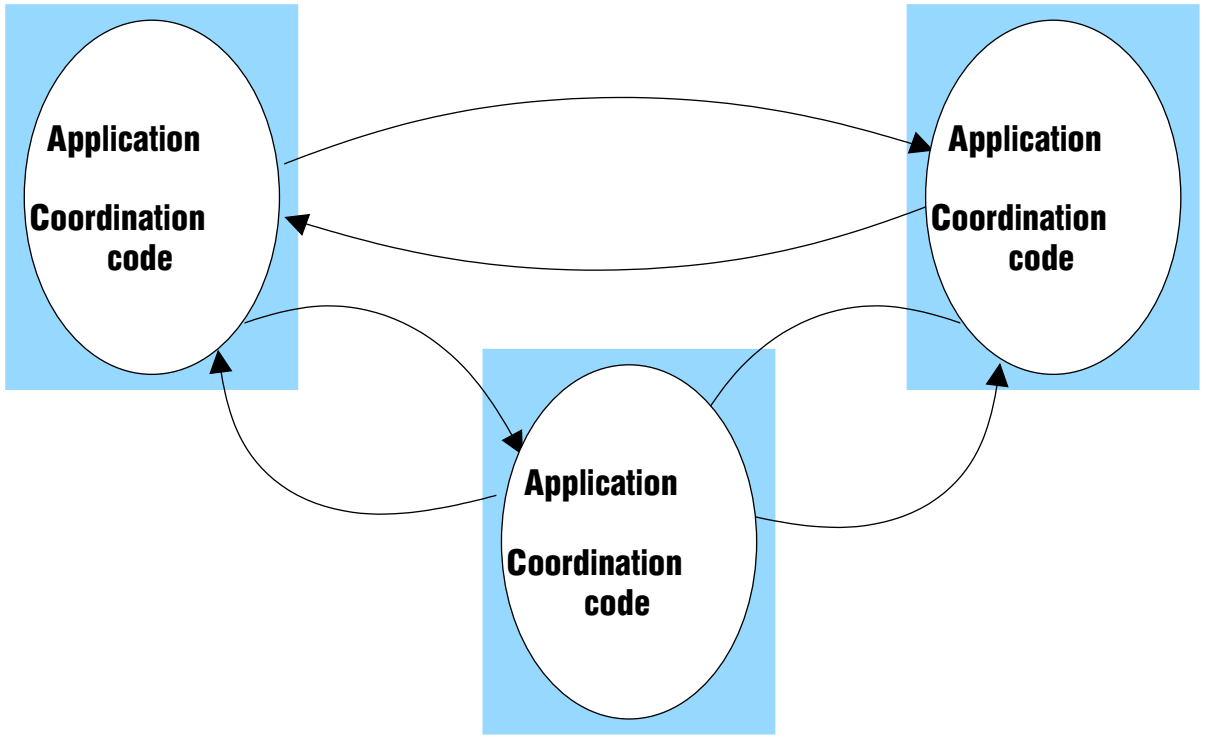
Proxy Servers and Caches

A *cache* is used to avoid performance bottlenecks and reduce network traffic: data resources are moved closer to the clients. A proxy server may host a cache.



Peer Processes

All processes play similar roles and result is achieved by cooperation.

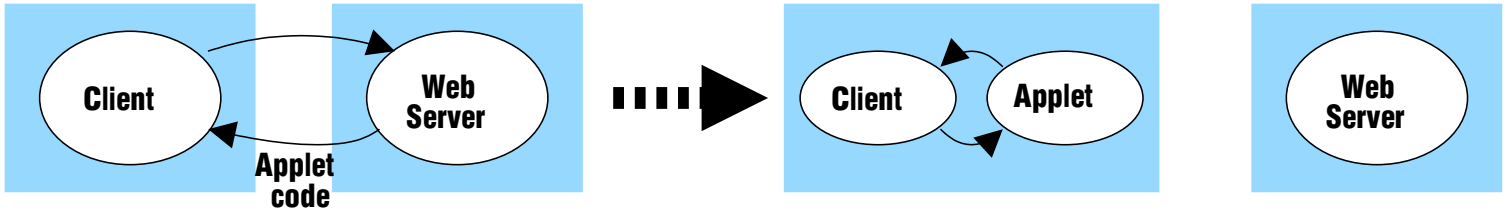


Pattern of communication will depend on the application requirements.

Impact of Mobility on Client-Server Model

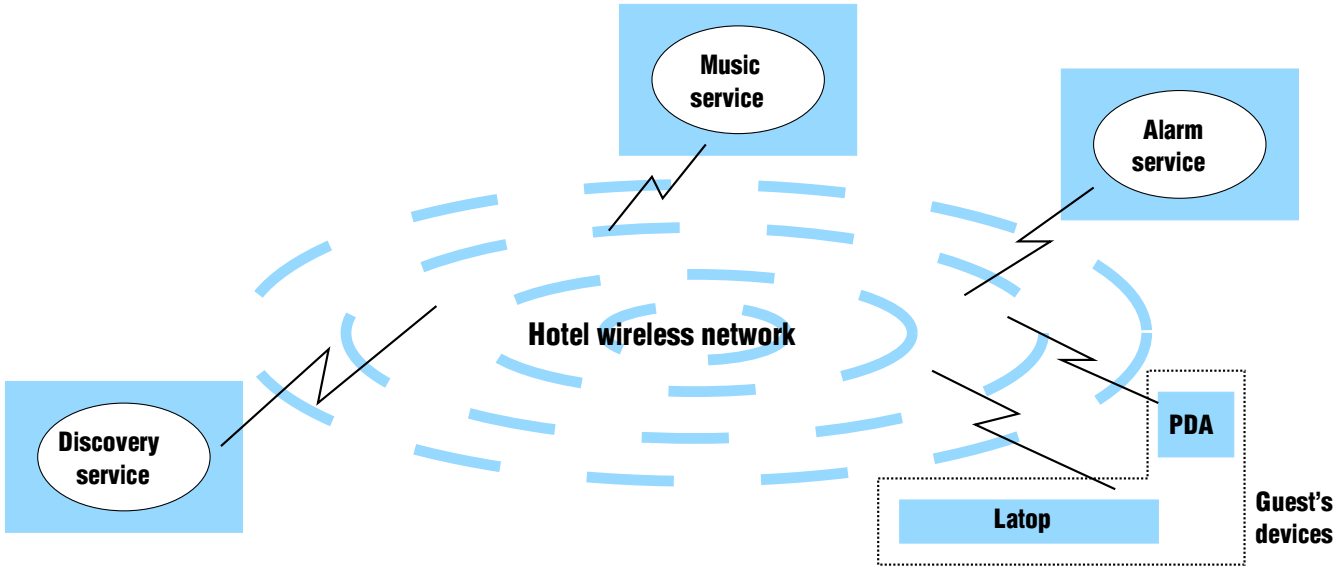
Mobility, both of code and devices, has altered the way that people think about client-server systems. There have been two major impacts:

- When mobile code is used, the code may be moved from one process to another, in effect allowing one process (e.g. a server) to delegate a task to another process (e.g. the client). For example, an applet is a piece of code which a client downloads from the server and runs locally. This can be used to reduce access delays and minimize communication costs.



Impact of Mobility (2)

- Mobility of devices mean that the configuration of a distributed system may be dynamic during the running of the system. Such systems are now being designed to allow computers and other mobile devices to be seamlessly added and removed (*spontaneous networking*). In these cases they need some means to *discover* the available services and *register* their own services.



Communication Subsystem

The hardware and software within a distributed system which provides the communication facilities is known as the *communication subsystem*. It consists of:

Transmission media: providing the physical connectivity, e.g. wire, cable, fibre and wireless channels;

Hardware devices: providing the linkage, e.g. routers, bridges, hubs, repeaters, network interfaces and gateways;

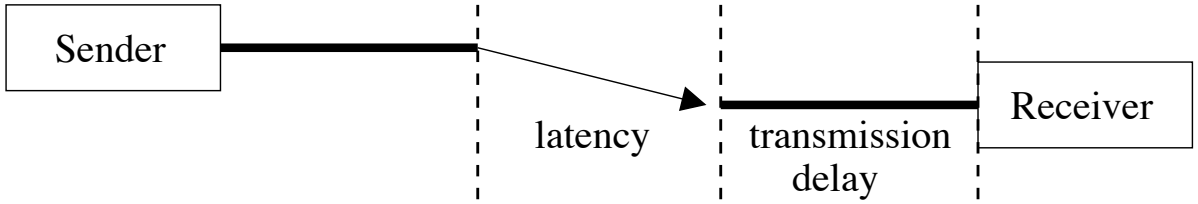
Software components: managing the communication, e.g. protocol stacks, communication handlers and drivers.

Impact on Distributed Computing

The impact of the communication on a distributed computation will be one of delay — the delay introduced by the message passing. The delay experienced by each individual message can be broken down into two factors: *latency* and *transmission delay*.

Latency is the time which is necessary to set up the communication, i.e. it is the delay incurred from the time the message is sent until it starts to arrive at the destination.

The transmission delay, determined by the length of the message and the *data transfer rate*, is the time it takes to transfer the data of the message.



message transmission time = latency + length/data transfer rate

Types of Network

	Range	*Bandwidth(Mbps)*	*Latency(ms)*	*Reliability*
LAN	1-2 km	10–10000	1–10	high
Ethernets, token rings over twisted copper wire, coaxial cable or optical fibre; usually serves a single organisation; no switching, messages are broadcast.				
WAN	worldwide	0.010–600	100–500	high
dedicated computers act as routers and are the source of delay; transmission speeds approach the speed of light				
MAN	2-50 km	1–150	10	high
Ethernets or ATM networks over high-bandwidth copper and fibre optic cabling				
Wireless LAN	0.15-1.5 km	2–108	5–20	low
e.g. infra-red links for mobile devices; connection limited to the immediate vicinity				
Wireless WAN	worldwide	0.010–2	100–500	low
e.g. mobile phone technologies GSM and CDPD				

Internetworking

Internetworks are networks built from subnetworks, which may themselves consist of subnetworks.... The Internet is the prime example.

When messages are passed across an internetwork, each constituent network may use a different transmission medium with differing switching requirements and reliability characteristics. However, it should appear as an intergrated network to the application programmer.

In most cases the problems associated with internetworking are solved by using IP addresses and the IP protocol.

Additionally specialised routers, known as *Internet Routers* are placed at points where subnetworks are connected. These machines will have distinct identities (IP addresses) within each subnetwork.

Bridges may also be used to link networks of different types, and the functionality of the bridge and the router may be combined in a single host. Similarly, routers often also serve as firewalls.

Internet Protocols

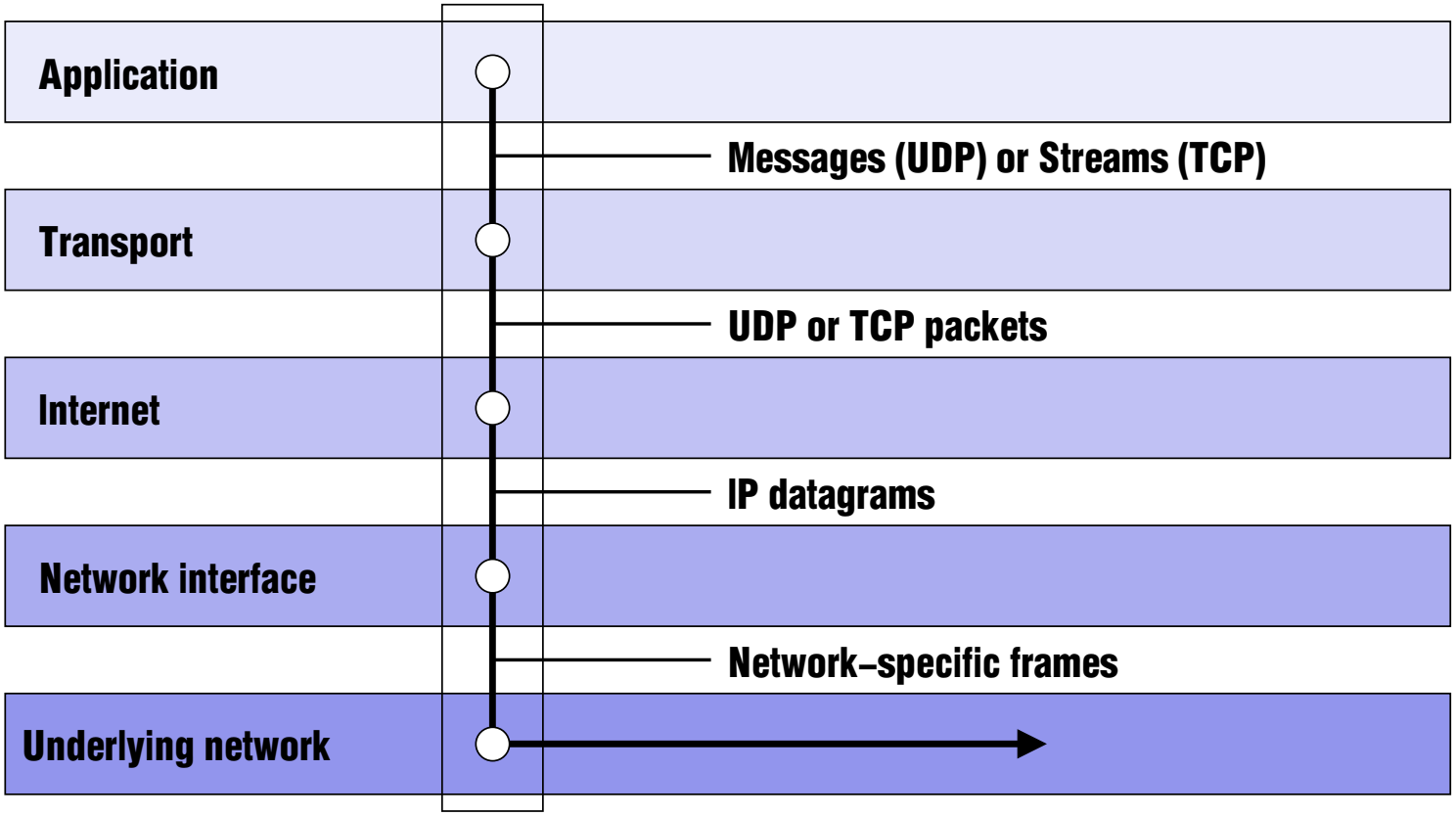
The Internet protocol stack works similarly to the ISO Open Systems Interconnections seven-layer reference model but is less rigidly defined, i.e. there is much more blurring between the layers. For example, the application layer protocol HTTP may be transported directly over TCP or the intermediate Secure Socket Layer (SSL) protocol may be introduced when needed.

IP datagrams form the basic form of communication over the Internet, but how these are actually sent over the network is not specified since this will depend on the technology of the underlying data links. For a single datagram this may change several times during transmission.

TCP and UDP provide two alternative transport layer protocols.

- TCP is designed to provide reliable, connection-oriented transmission and is quite sophisticated.
- UDP is much more basic, barely offering more than access to IP. The advantage of this is that it introduces very little overhead.

TCP/IP Layers



IP addresses

- This is a **universal** addressing scheme: any host on the Internet can send a message to any other.
- 32-bit addresses, currently being replaced by 128 bit addresses. Under both schemes the address identifies both the host and the subnetwork to which it belongs.
- One subclass of addresses is reserved for multicasting but this is not supported by all routers.
- The network portion of addresses are allocated centrally by the Internet Network Information Center; host identifiers are allocated within each network.
- Dynamic IP addresses are commonly used for hosts which do not need a constant immutable presence on the Internet, for example allocated by a DHCP (Dynamic Host Configuration Server) server to a "visiting" device. These are allocated on a **leased** basis, which conserves addresses and adds security.

Domain Names

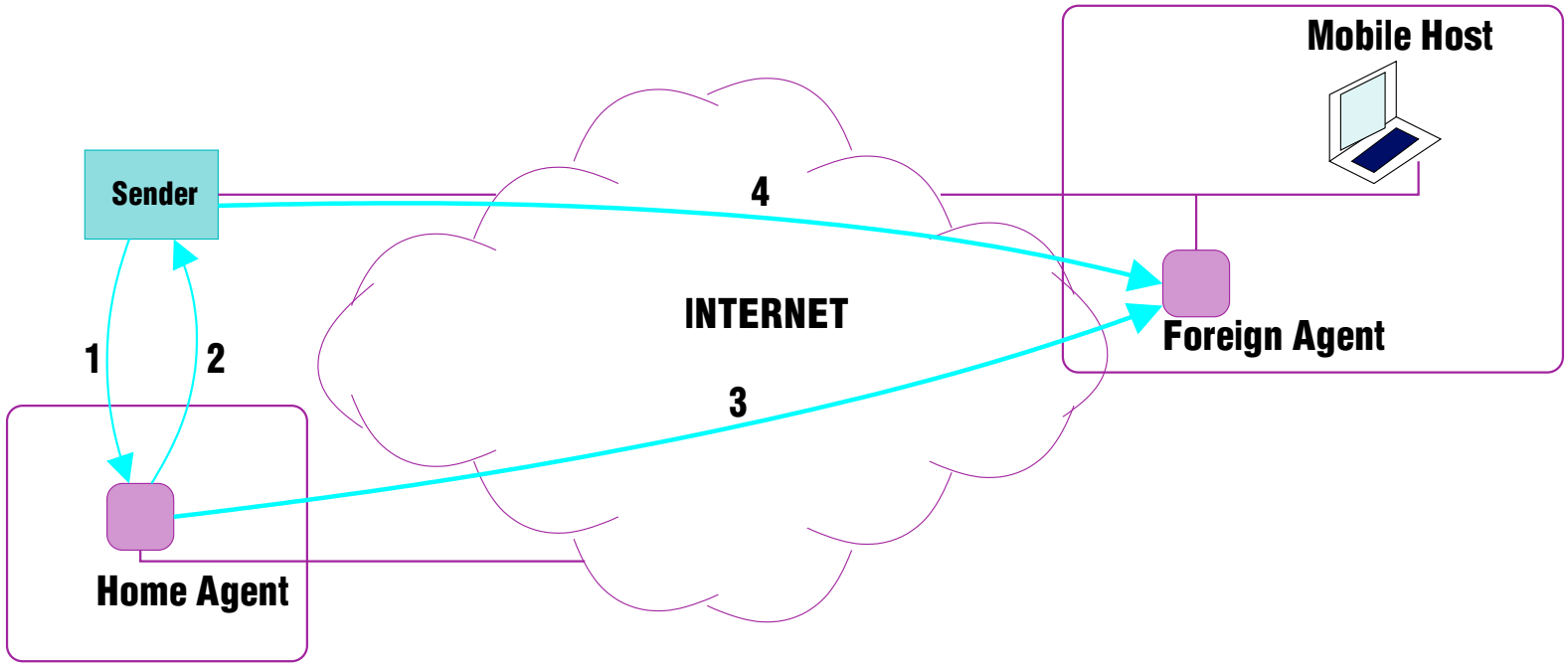
- Because IP addresses are not very memorable for human users, the Internet also supports a scheme of symbolic names for hosts and networks, e.g. `clootie.inf.ed.ac.uk`.
- This **domain name** is represented in a hierarchical fashion designed to reflect the organisational hierarchy and independent of the physical arrangement of the Internet (*location transparency*).
- In order for communication to take place a domain name must be translated into an IP address and this is carried out using the **Domain Name Service (DNS)**.
- There are at least two DNS servers in each domain.
- Each server holds a partial map of the domain name tree below their domain. This must consist of at least all the domain and host names within their own domain; often it will contain more.
- Name resolution is carried out recursively from right to left, issuing request to DNS servers in relevant domains as necessary. Results are cached for future use.

MobileIP (1)

MobileIP allows IP communication to continue transparently with respect to the sever's current location.

- The mobile host is allocated a permanent IP address, corresponding to its "home" domain.
- When it is roaming, a *home agent* runs on a fixed machine in the home domain. Correspondingly a *foreign agent* running on a fixed machine at the temporary domain.
- The home agent keeps track of the current IP address of the mobile host and acts as a proxy during periods of disconnection.
- When the mobile machine is registered with the foreign agent, the foreign agent contacts the home agent, notifying it of the new temporary IP address.
- Requests for the server are captured by home agent and re-routed, embedded in MobileIP packets, to the foreign agent.

MobileIP (2)



1 Sender sends first IP packet addressed to the mobile host
2 Home agent returns the address of the foreign agent to the sender
3 Home agent forwards the first IP packet to the foreign agent
4 Subsequent IP packets sent to the foreign agent directly.