MODULE 01

SYLLABUS: Overview of the Internet, Protocol layering. Application Layer: Application-Layer Paradigms, Client-server applications - World Wide Web and HTTP, FTP. Electronic Mail, DNS. Peer-to-peer paradigm - P2P Networks, Case study: BitTorrent.

TO WATCH VIDEO CLASSES OF THIS COURSE: TAP HERE OR SCAN QR CODE

INTRODUCTION

Overview of the Internet

The Internet is an internetwork, a network of networks, It connects billions of devices via medium.

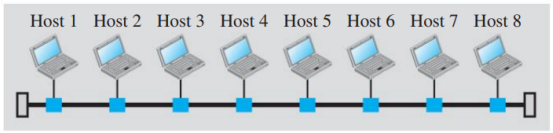
A network is the interconnection of a set of devices capable of communication.

- **Hosts (end systems):** Computers, laptops, phones, workstations, security systems.
- **Connecting devices:**
  - **Router:** connects a network to other networks.
  - **Switch:** connects devices together within a network.
  - **Modem (modulator-demodulator):** changes the form of data so that it suits the medium.

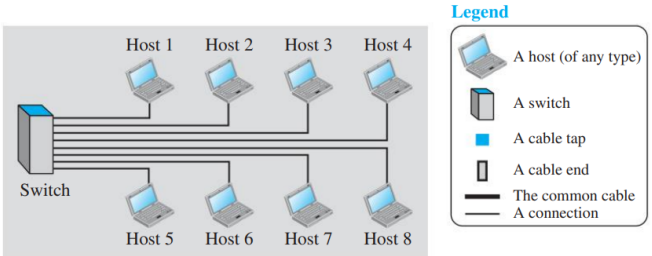
The devices are connected by transmission media, which may be wired (cable) or wireless (air). Networks are broadly of two categories: LAN and WAN.

Local Area Network (LAN)

- **Privately owned; connects hosts in one office, building or campus.**
- Each host has an **address** unique in the LAN.
- Every packet carries the **source address** and **destination address**.
- **Past:** all hosts shared a **common cable**. Every host received every packet, and only the destination kept it.
- **Today:** a **link-layer switch** reads the destination address and forwards the packet only to that host.
- A switch reduces traffic and lets **several pairs communicate at the same time**.



a. LAN with a common cable (past)



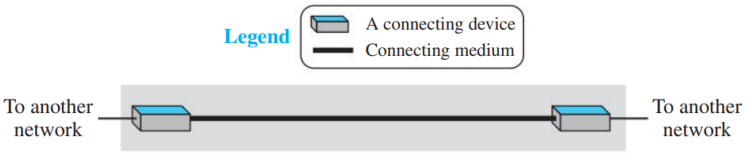
b. LAN with a switch (today)

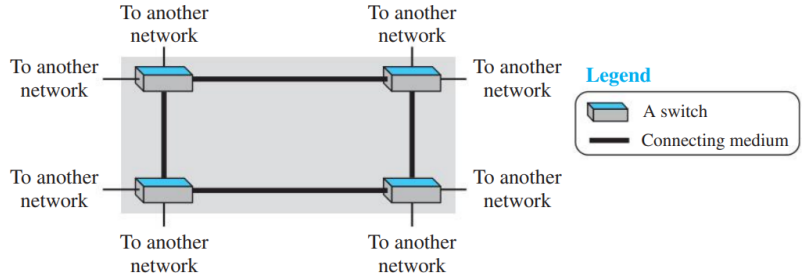
When LANs were used in isolation, they were designed only to share resources among their hosts. Today LANs are connected to each other and to WANs.

Wide Area Network (WAN)

- Covers a town, state, country or the world.
- Interconnects **connecting devices** (switches, routers, modems).
- Created and run by communication companies and leased to organisations.
- **Point-to-point WAN:** connects two devices through a cable or air.
- **Switched WAN:** has more than two ends. It is several point-to-point WANs joined by switches and forms the global **backbone**.

About Figure: Point-to-point WAN → : Two connecting devices joined by one transmission medium.





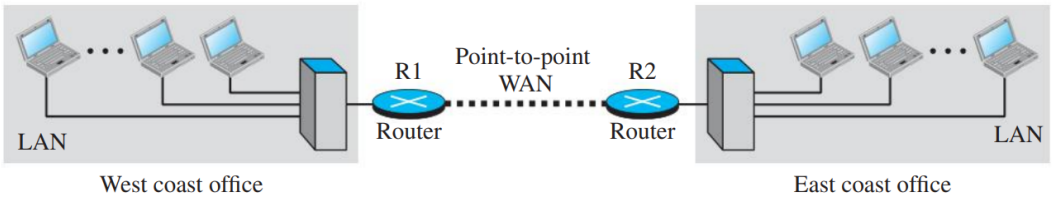
About Figure: A switched WAN ← : Several point-to-point links meet at switches, giving more than two ends.

Parameter	LAN	WAN
Span	Office, building, campus	Town to world
Interconnects	Hosts	Connecting devices
Ownership	Private	Communication companies (leased)
Types	Common cable, switched	Point-to-point, switched

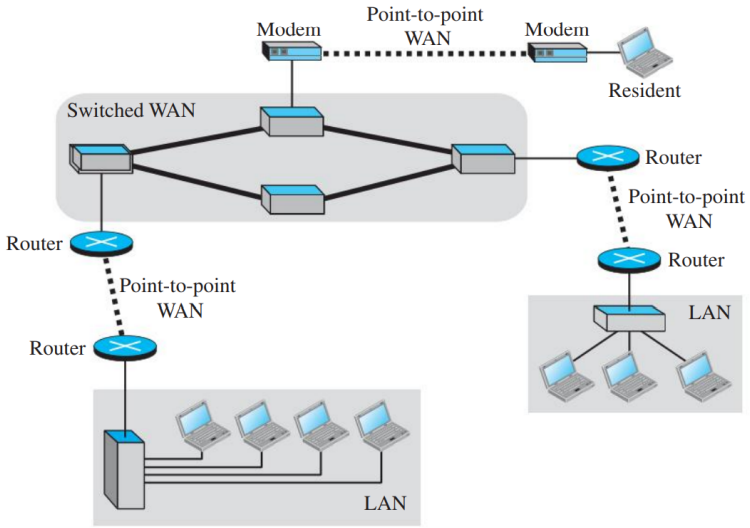
Internetwork

Two or more connected networks form an **internetwork (internet)**. Example: two office LANs are joined by a leased point-to-point WAN through routers R1 and R2. This is a **private internet**.

- Same-office message: the switch delivers it and the router blocks it.
- Message to the other office: R1 routes it to R2, which delivers it.



About Figure: Two LANs with routers R1 and R2, joined by a point-to-point WAN.



About Figure: Many LANs and WANs, including a switched WAN, joined by routers.

To connect n LANs directly in pairs by point-to-point WANs:

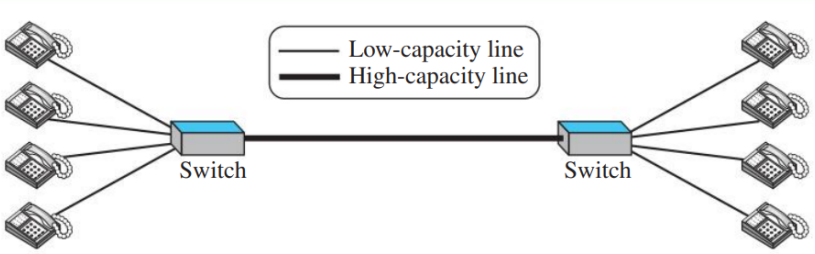
Number of point-to-point WANs, $W = \frac{n(n-1)}{2}$

Where W = number of WANs and n = number of LANs (for n = 5, W = 10).

Switching

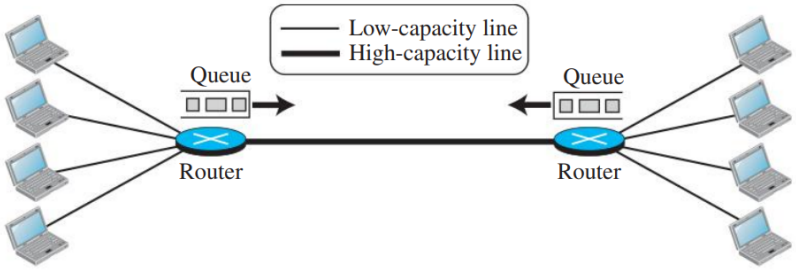
An internet is made of **links and switches**. A switch connects at least two links and forwards data between them. The two types are circuit-switched and packet-switched networks.

1. Circuit-Switched Network



- A dedicated **circuit** is always available between the two end systems.
- The switch only makes it active or inactive.
- Switches **forward only**; they cannot store.
- Example: telephone network.
- The shared high-capacity line is sized for all calls at once (4 calls = 4 times one voice line).
- With one call, only one-fourth of the line is used.
- So it is **efficient only at full capacity**.

2. Packet-Switched Network



- Data is sent as **packets**; each packet is independent.
- Routers **store and forward**.
- Each router has a **queue**. If the line is busy, packets wait and are forwarded in arrival order.
- More efficient than circuit switching, but packets **may be delayed**.
- The Internet mainly uses packet switching.

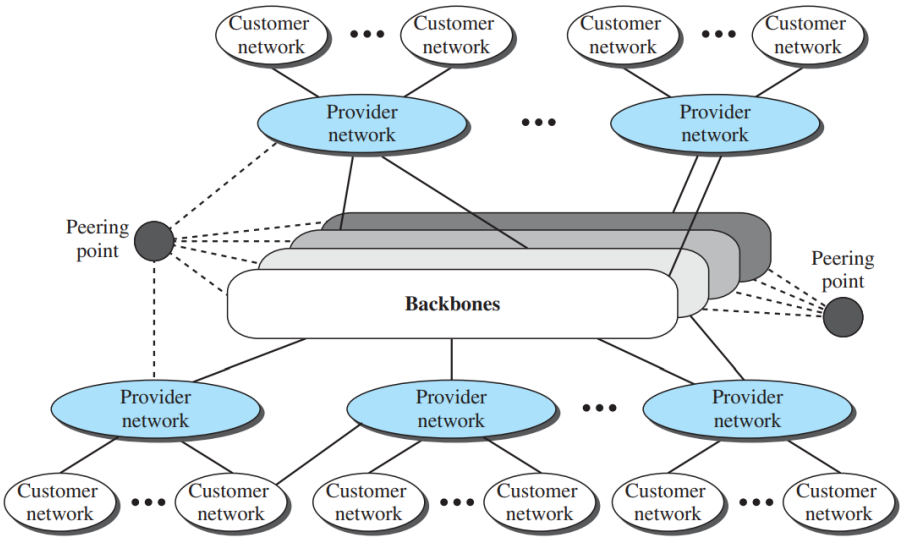
Parameter	Circuit-switched	Packet-switched
Path	Dedicated circuit	No dedicated path
Unit	Continuous communication	Packets
Switch function	Forward only	Store and forward
Efficiency	Only at full load	Higher
Delay	None once connected	Queuing delay possible
Example	Telephone network	Internet

The Internet

An **internet** (small *i*) is two or more connected networks. The **Internet** (capital *I*) is made of thousands of interconnected networks, at three levels:

1. **Backbones:** large networks owned by companies (Sprint, Verizon, AT&T, NTT). They connect at **peering points**.
2. **Provider networks:** smaller networks that pay backbones for service.
3. **Customer networks:** edge networks that use the service and pay providers.

Backbones are **international ISPs**. Provider networks are **national or regional ISPs**.



About Figure: Backbones joined at peering points sit at the top. Provider networks connect to them, and customer networks connect at the edge.

Accessing the Internet

A user connects to an ISP, normally through a point-to-point WAN.

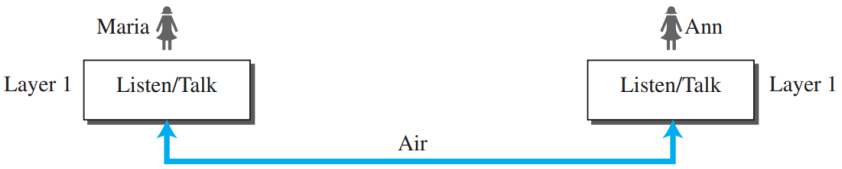
- **Telephone network:**
  - **Dial-up:** modem on the voice line; slow; the line cannot carry voice at the same time.
  - **DSL:** higher speed; voice and data together.
- **Cable network:** higher speed, shared with neighbours.
- **Wireless network:** wired/wireless mix or wireless WAN.
- **Direct connection:** a large organisation leases a high-speed WAN to a regional ISP.

Communication needs **hardware** (devices, media) and **software** (protocols). They are coordinated by protocol layering.

PROTOCOL LAYERING

A **protocol** is a set of rules followed by the sender, receiver and intermediate devices. For complex communication, the task is divided into **layers**, each with its own protocol. This is **protocol layering**.

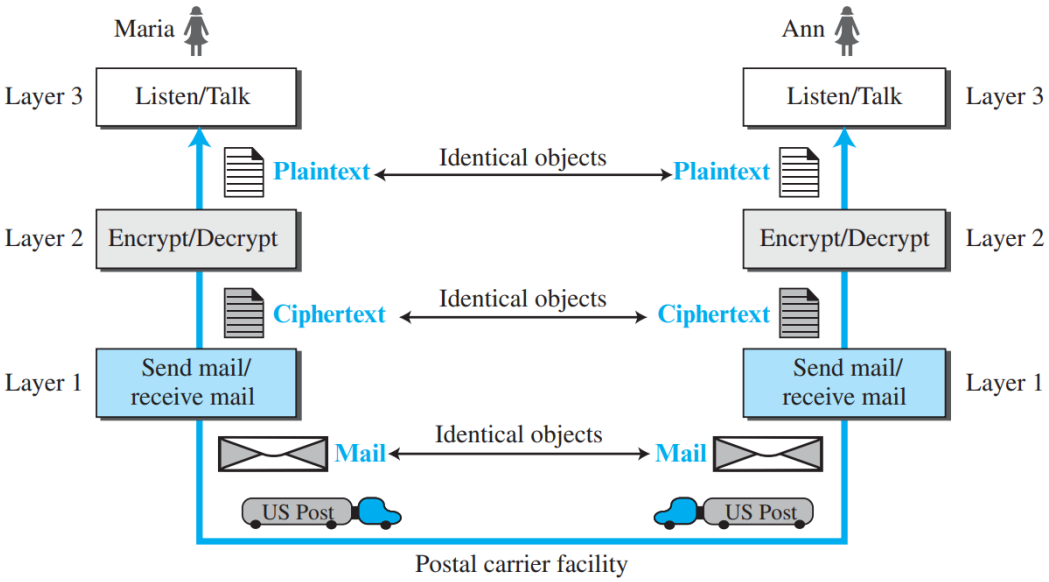
Need for Protocol Layering



Scenario 1 (one layer): Maria and Ann talk face to face. Even this needs rules: greet, use suitable vocabulary, do not interrupt, hold a dialogue, say goodbye.

Scenario 2 (three layers): Ann moves away. They send encrypted letters by post.

Layer	At sender (Maria)	At receiver (Ann)
3	Listens and creates **plaintext**	Reads the plaintext
2	**Encrypts** to **ciphertext**	**Decrypts** to plaintext
1	Puts the ciphertext in an envelope, adds addresses, mails it	Collects the letter and removes the ciphertext



About Figure: Maria and Ann each have three layers. Plaintext moves between layers 3 and 2, ciphertext between layers 2 and 1, and mail through the postal carrier.

Advantages of layering

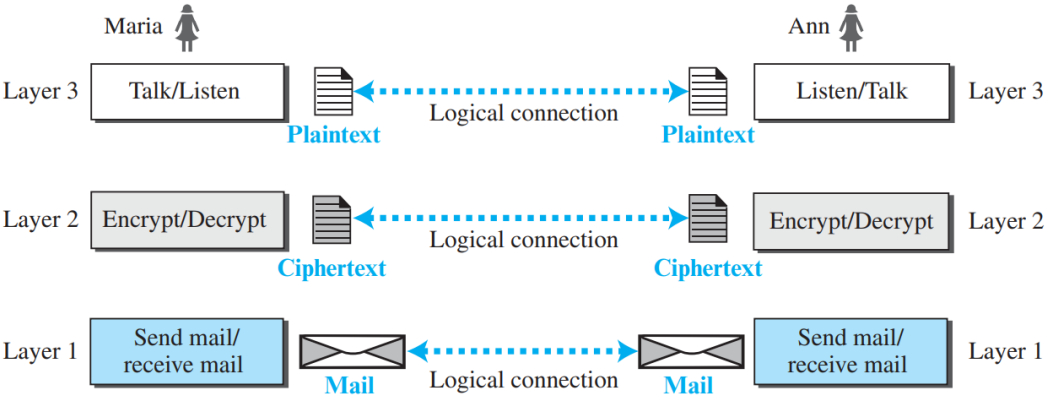
- **Modularity:** each layer is a black box. Only the layer that needs change is replaced.
- **Service separated from implementation:** a layer only needs the lower layer's service, not how it works.
- **Simpler intermediate systems:** they need only some layers, so they cost less.

Disadvantage: a single layer looks simpler, but any change forces the whole system to be replaced.

Principles of layering

1. **Bidirectional:** each layer performs two opposite tasks (talk/listen, encrypt/decrypt, send/receive).
2. **Identical objects:** the object under each layer is identical at both sites.

Logical connection: each layer imagines a direct connection to its **peer layer** at the other site. Real data goes down the layers, across the medium and up again.

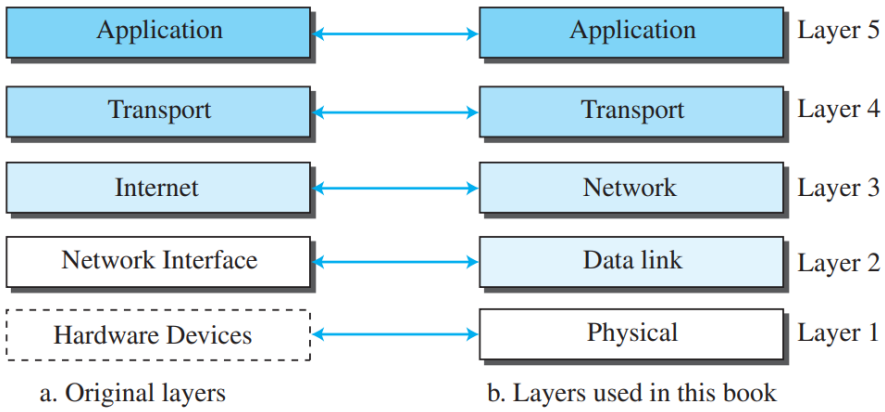


About Figure: Dotted lines join same-numbered layers of Maria and Ann. Only layer 1 uses a real path.

TCP/IP Protocol Suite

TCP/IP is a **protocol suite** (protocols organised in layers) used in the Internet.

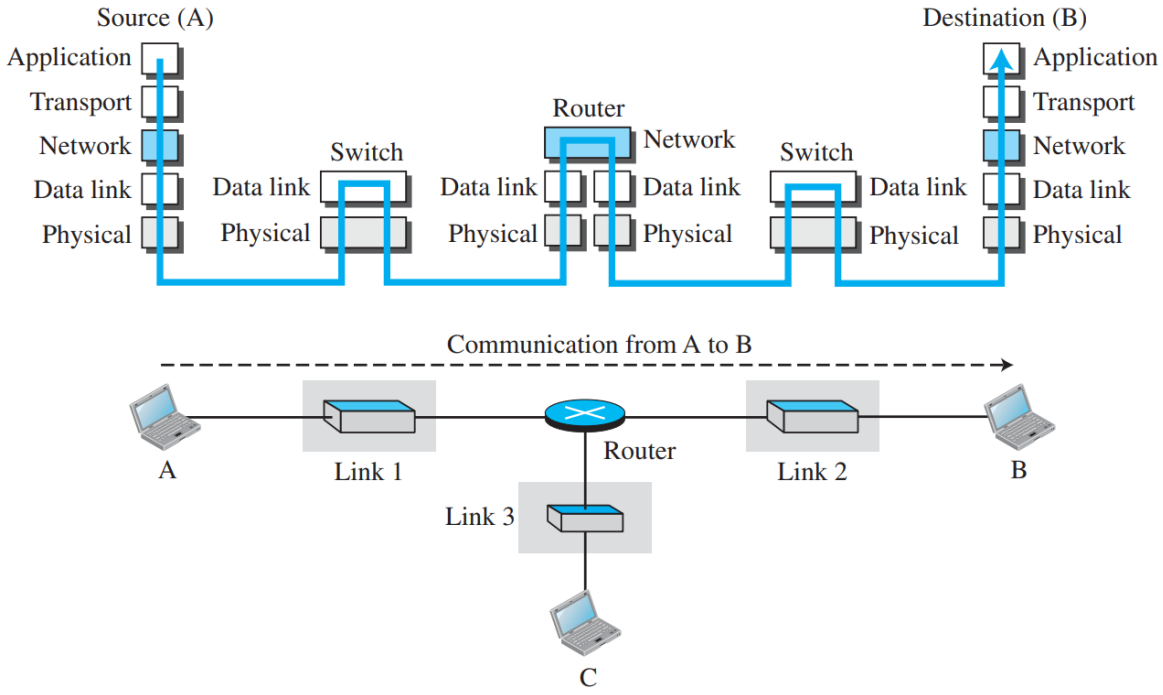
- It is **hierarchical**: each upper protocol uses the services of lower ones.
- It was defined with four layers.
- Today it is studied as **five layers**: application, transport, network, data link, physical.



About Figure: (a) The original four layers. (b) The five-layer model used in this book.

Layers in devices

Take three LANs, each with a link-layer switch, joined by one router. Host A talks to host B.



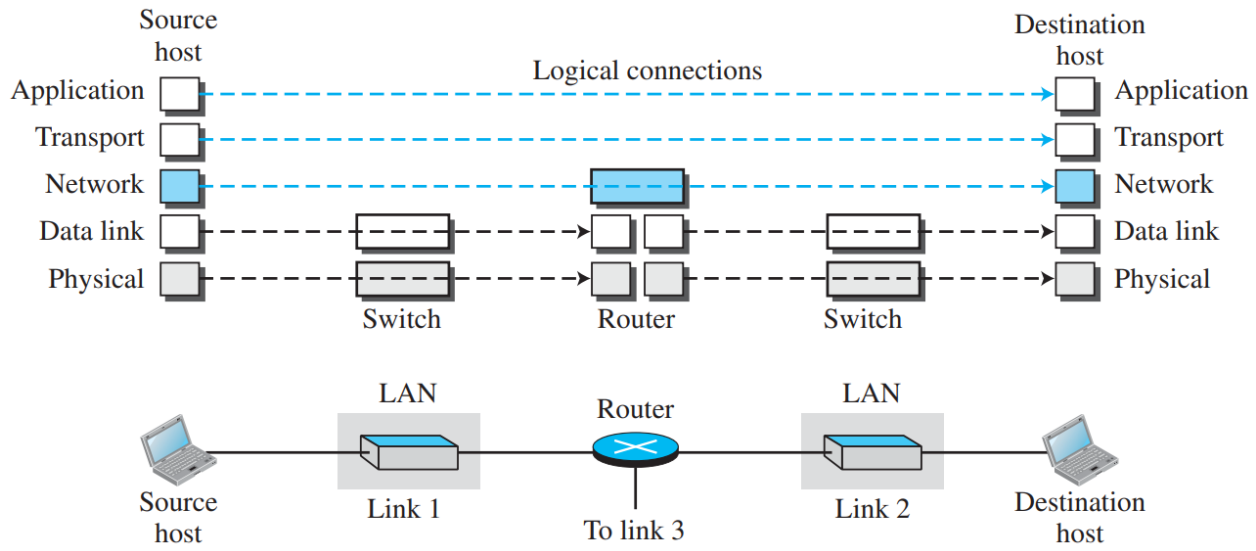
About Figure: The hosts run five layers, the router three, and each switch two.

Device	Layers	Reason
Source and destination hosts	All five	They create and consume the message
Router	Network, data link, physical	It only routes, so no transport or application layer
Link-layer switch	Data link, physical	All its connections are in one link

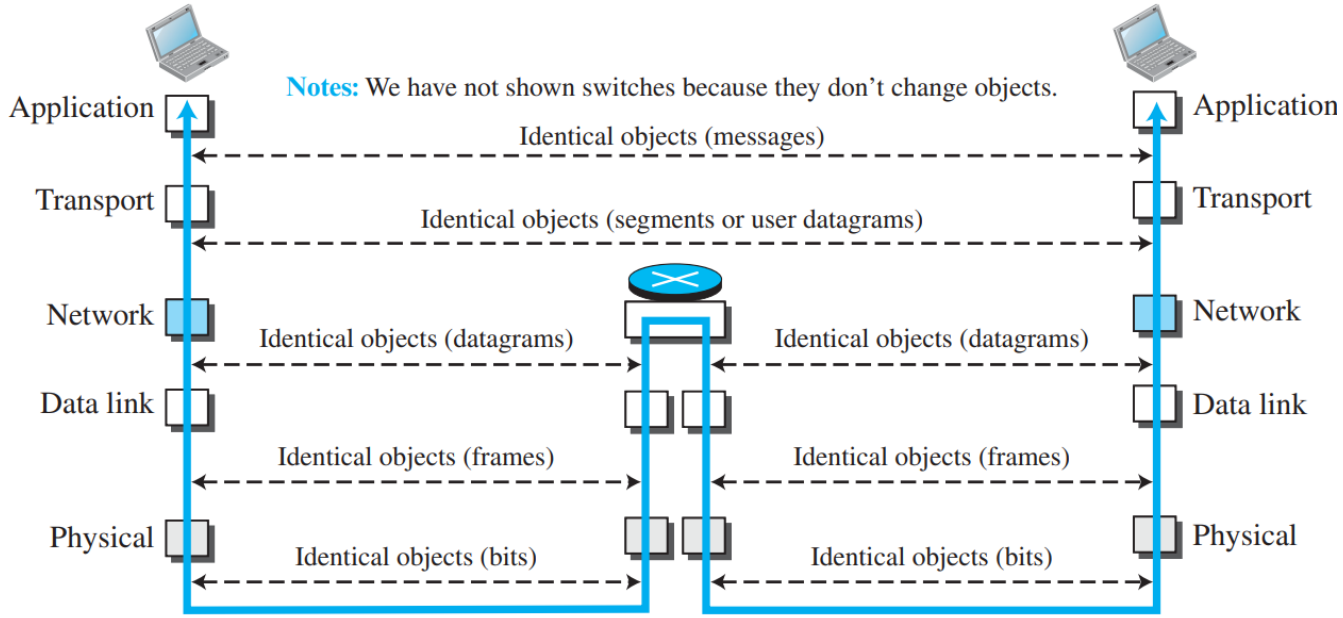
- A router has **one** network layer but **n pairs** of data-link and physical layers (n = links connected), since each link may use different protocols.
- A switch has one data-link and one physical layer.

Logical connections in TCP/IP

- Application, transport and network layers: **end-to-end** (domain is the internet).
- Data-link and physical layers: **hop-to-hop** (domain is the link).
- Top-three-layer data units are not changed by routers or switches.
- A router may **fragment** a datagram, so identical network-layer objects exist only between two hops.



About Figure: End-to-end connections for the top three layers; hop-to-hop connections for the bottom two.



About Figure: The identical object below each layer: message, segment/user datagram, datagram, frame and bit.

Functions of the TCP/IP Layers

Layer	Data unit	Scope
Application	Message	End-to-end
Transport	Segment (TCP) / user datagram (UDP)	End-to-end
Network	Datagram	Host-to-host
Data link	Frame	Hop-to-hop
Physical	Bit	Hop-to-hop

1. Application layer

- Gives **process-to-process communication**; a process sends a request and receives a response.
- Exchanged unit: **message**.
- Has many predefined protocols; users can also create their own.

Protocol	Use
HTTP	Accessing the World Wide Web
SMTP	Electronic mail
FTP	File transfer between hosts
TELNET, SSH	Remote access
SNMP	Network management
DNS	Finding the network-layer address from a name
IGMP	Collecting group membership

2. Transport layer

- Delivers the message from an application program at the source to the **corresponding program at the destination**.
- Kept separate from the application layer for **separation of tasks**, and because there are several transport protocols to choose from.

Protocol	Nature	Features
TCP	Connection-oriented	Logical connection first; byte stream; **flow, error and congestion control**
UDP	Connectionless	Independent datagrams; no flow, error or congestion control; small overhead; suits short messages
SCTP	Newer	Designed for multimedia applications

3. Network layer

- Provides **host-to-host communication** and **routing** through routers.
- Kept separate because routers do not need the application and transport layers.
- **IP** defines the datagram format, address format and forwarding of datagrams hop by hop.
- IP is **connectionless**, with no flow, error or congestion control.
- **Routing protocols** (unicast and multicast) build **forwarding tables**; they do not route packets themselves.

Auxiliary protocol	Function
ICMP	Reports problems while routing
IGMP	Helps IP with multicasting
DHCP	Gives a host its network-layer address
ARP	Finds the link-layer address from the network-layer address

4. Data-link layer

- Moves the datagram across **one link** (wired or wireless LAN or WAN).
- TCP/IP defines **no specific protocol** here; it supports any protocol that can carry a datagram.
- Encapsulates the datagram in a **frame**.
- Services vary: some protocols do error detection and correction, some only detection.

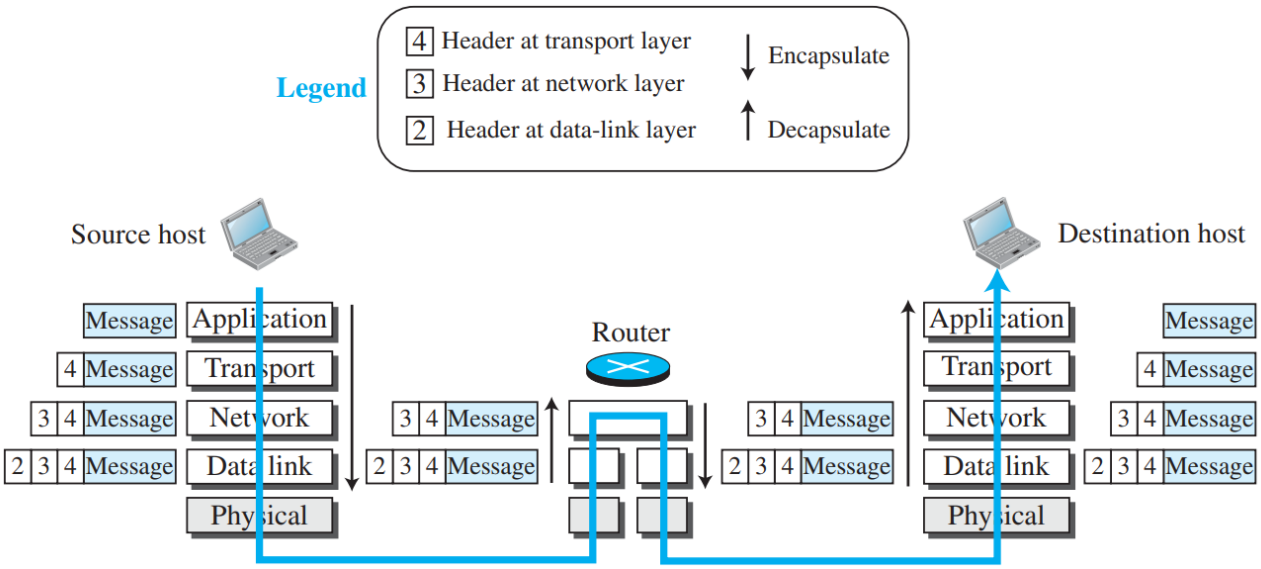
5. Physical layer

- Carries individual **bits** of a frame across the link.
- Below it lies the **transmission medium**, which carries **electrical or optical signals**, not bits.
- Physical-layer protocols convert bits to signals.

MODEL QUESTION (5 Marks): Explain the layered architecture of the TCP/IP protocol suite. State the function of each layer and the name of the data unit handled at each layer.

Encapsulation and Decapsulation

Encapsulation: each layer adds a header to the payload from the layer above, moving down.
Decapsulation: each layer removes its header, moving up.



About Figure: The source adds transport, network and data-link headers. The router swaps the data-link header. The destination removes all headers.

<u>At the source host (encapsulation only)</u>

1. **Application:** data is a **message**.
2. **Transport:** adds a header with **source and destination port numbers** and flow/error/congestion control information. Result: **segment** or **user datagram**.
3. **Network:** adds a header with **source and destination addresses**, header error checking and fragmentation information. Result: **datagram**.
4. **Data link:** adds a header with the **link-layer addresses** of the host and next hop. Result: **frame**.

<u>At the router (decapsulation, then encapsulation)</u>

1. The data-link layer removes the frame header.
2. The network layer reads the addresses and uses its **forwarding table** to find the next hop. It changes the datagram only to **fragment** it.
3. The next link's data-link layer adds a new frame header.

<u>At the destination host (decapsulation only)</u>

- Each layer removes its header and passes the payload up to the application.
- Decapsulation includes **error checking**.
- Link-layer switches do not encapsulate or decapsulate.

Overhead and efficiency. Every header is overhead, so only part of the frame is application data.

Frame size, $F = D + H_t + H_n + H_d$
Where F = frame size, D = application data, H_t = transport header, H_n = network header, H_d = data-link header and trailer (all in bytes).

Overhead efficiency, $\eta = \frac{D}{F} \times 100\,\%$
Where eta = percentage of the frame that is application data.

MODEL QUESTION (3 Marks): An application on a host generates 1460 bytes of data. The transport layer uses UDP with an 8-byte header, the network layer adds a 20-byte IP header, and the data link layer adds an Ethernet header and trailer totalling 18 bytes. Calculate the size of the frame transmitted over the network and the percentage of the frame that is actual application data (overhead efficiency).

Solution

Given: D = 1460 bytes, H_t = 8 bytes, H_n = 20 bytes, H_d = 18 bytes.
Required: frame size F and efficiency eta.

$$F = D + H_t + H_n + H_d$$

F = 1460 + 8 + 20 + 18 = 1506 bytes.

$$\eta = \frac{D}{F} \times 100\%$$

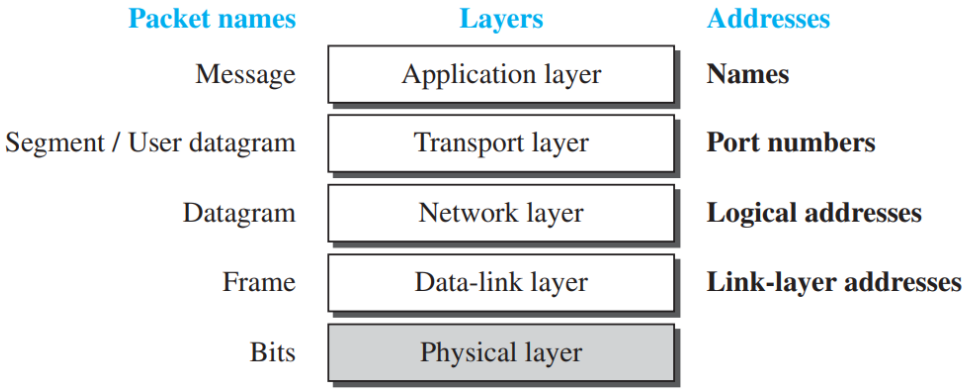
$$\eta = \frac{1460}{1506} \times 100\% = 96.95\%.$$

Frame size = **1506 bytes**; application data = **96.95 %** of the frame (overhead 46 bytes, 3.05 %).

MODEL QUESTION (4 Marks): Trace the journey of a message from an application on host A to the application on host B, showing encapsulation at the sender and decapsulation at the receiver. Draw a neat diagram showing the logical connection between peer layers.

Addressing

Four pairs of addresses are used. The **physical layer needs none** because its unit is a bit.

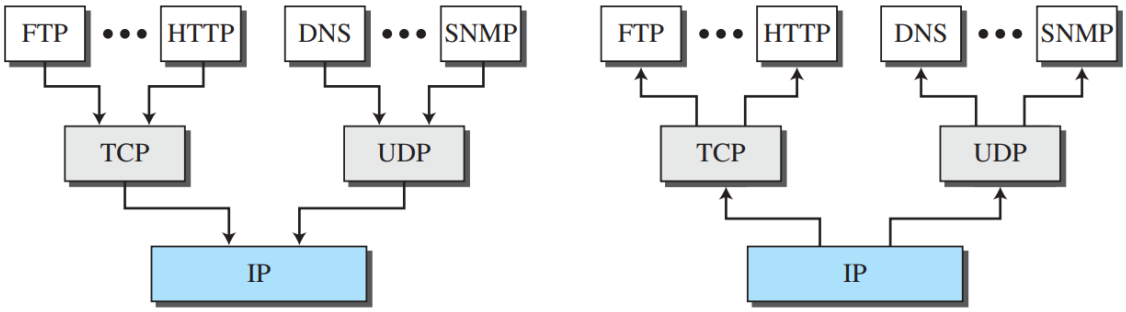


About Figure: Each layer paired with its packet name and address type; no address for the physical layer.

Layer	Address	Scope
Application	**Names** (someorg.com, e-mail address)	Identifies the service
Transport	**Port numbers**	Local; identifies the program on a host
Network	**Logical (IP) address**	**Global**; identifies a device's connection to the Internet
Data link	**Link-layer (MAC) address**	**Local**; identifies a host or router in a LAN or WAN

Multiplexing and Demultiplexing

- **Multiplexing (source):** a protocol encapsulates packets from several higher-layer protocols, one at a time.
- **Demultiplexing (destination):** a protocol delivers decapsulated packets to several higher-layer protocols, one at a time.
- A **header field** identifies the protocol of the encapsulated packet.



a. Multiplexing at source

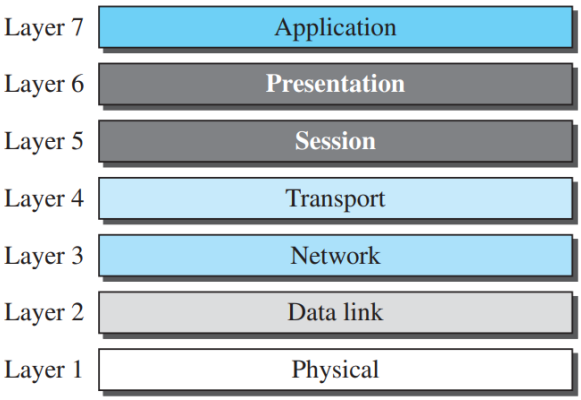
b. Demultiplexing at destination

About Figure: (a) Application protocols feed TCP or UDP, which feed IP, at the source. (b) IP delivers to TCP or UDP, which deliver to the right application protocol, at the destination.

Layer	Example
Transport	TCP or UDP accepts messages from HTTP, FTP, DNS, SNMP
Network	IP accepts segments from TCP, datagrams from UDP, packets from ICMP and IGMP
Data link	A frame carries payload from IP or ARP

OSI Model

- **ISO** (International Organization for Standardization, 1947) is the organisation. **OSI** is its model.
- OSI (Open Systems Interconnection) was introduced in the late 1970s.
- An **open system** lets any two different systems communicate regardless of architecture.
- OSI is **not a protocol**. It is a model for designing a flexible, robust and interoperable network.
- It has **seven layers**.

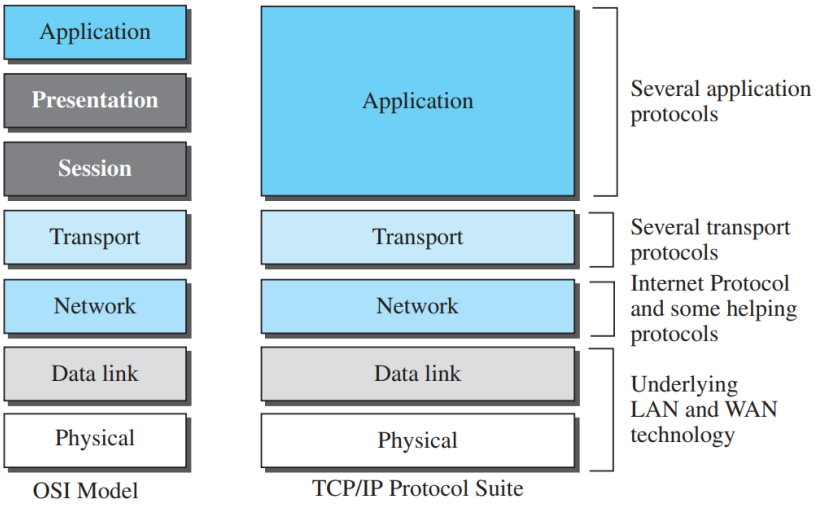


About Figure: Seven layers from top to bottom: application, presentation, session, transport, network, data link, physical.

OSI layer	Main function
Application	Network services to the user
Presentation	Translation, encryption, compression
Session	Dialogue control and synchronisation
Transport	End-to-end delivery, flow and error control
Network	Logical addressing and routing
Data link	Framing, physical addressing, error detection
Physical	Transmission of raw bits

OSI versus TCP/IP

- TCP/IP has no separate **session** and **presentation** layers.
- The TCP/IP application layer equals OSI application + presentation + session.



About Figure: The seven OSI layers beside the five TCP/IP layers. Three OSI top layers map to one TCP/IP application layer; the other four match one to one.

Why the two layers were left out:

1. TCP/IP has several transport protocols, and some session functions exist in them.
2. The application layer is many programs, so each can include any session or presentation function it needs.

Parameter	OSI	TCP/IP
Layers	Seven	Five (originally four)
Session, presentation	Separate	Merged into application
Nature	Reference model	Actual protocol suite
Introduced	Late 1970s, after TCP/IP was in place	Developed first

Why OSI was not successful

1. It was completed when TCP/IP was fully in place; changing would cost too much.
2. Some layers were never fully defined (session, presentation protocols and software).
3. Its implementations did not perform well enough for the Internet to switch.

APPLICATION-LAYER PARADIGMS

The **application layer** provides services to the user. The Internet was built so that this layer can offer services. The other four layers only support it.

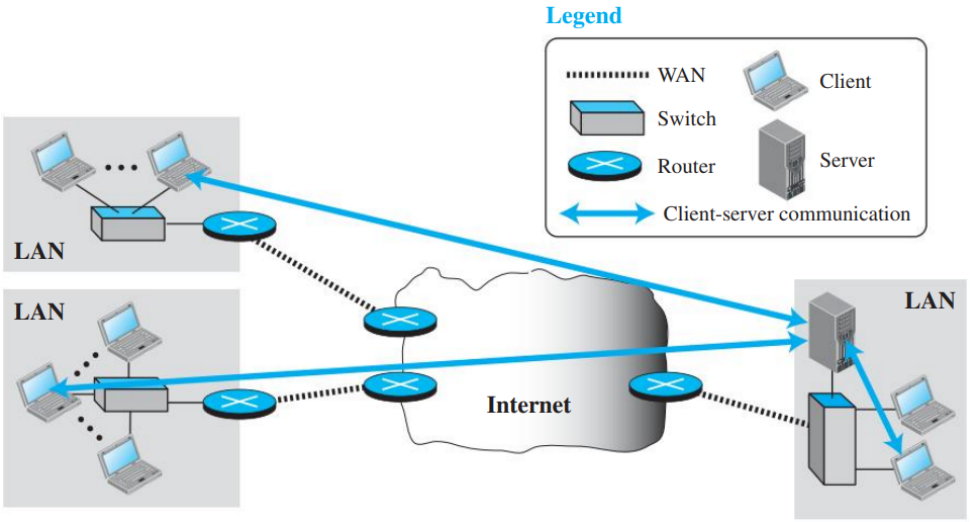
Providing Services

- Older networks (telephone, postal) were designed for **one service** each.
- TCP/IP is **layered**, so protocols can be added, removed or replaced.
- The application layer is the **highest layer**. Its protocols give service to no other protocol and only **receive service from the transport layer**.
- So new application protocols can be added easily, as long as they use a transport-layer protocol.

Parameter	Standard protocols	Nonstandard protocols
Defined by	Internet authority (standardised, documented)	A programmer or company
Structure	A pair of programs (client and server)	Two programs using transport-layer services
Approval	Standardisation process	**Not needed** if used privately
Examples	HTTP, FTP, SMTP, DNS	A company's own protocol linking its offices

1. Client-Server Paradigm

- The service provider is a **server process**. It runs **continuously**, waiting for requests.
- A **client process** starts only when it needs service.
- Their roles are different: a client cannot run as a server, and a server cannot run as a client.
- Two programs must be written for each service.
- Analogy: a telephone directory centre (server) and a subscriber (client).



About Figure: Three clients on different LANs communicate with one server through the Internet.

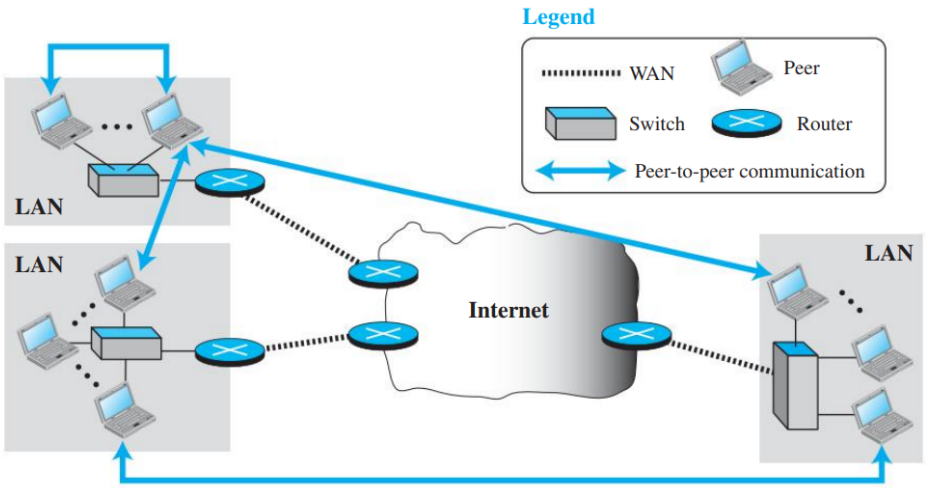
Problems

- The load is concentrated on the server, so it must be a **powerful computer**. Even then it may be overwhelmed by many clients.
- Someone must bear the server's cost, so the service must bring some income.

Examples: WWW and HTTP, FTP, SSH, e-mail.

2. Peer-to-Peer (P2P) Paradigm

- **No server runs all the time.** Responsibility is **shared between peers**.
- A computer can provide service at one time and receive service at another, or **do both together**.
- Suited to Internet telephony and file sharing.



About Figure: Peers on different LANs communicate directly with each other through the Internet, with no central server.
Advantages: easily **scalable** and **cost-effective** (no always-on costly server).

Challenges

- **Security:** harder to secure communication between distributed peers.
- **Applicability:** not every application can use it (for example the Web).

Examples: BitTorrent, Skype, IPTV, Internet telephony.

3. Mixed Paradigm

An application can combine both. A light client-server exchange finds the address of a peer. The actual service is then taken from that peer by P2P.

Parameter	Client-server	Peer-to-peer
Roles	Fixed: client requests, server provides	Each peer both provides and receives
Server process	Must run continuously	No always-running server
Load	Concentrated on the server	Shared among peers
Scalability and cost	Needs a powerful costly server	Easily scalable, cost-effective
Security	Easier to secure	Harder to secure
Applicability	Suits all services	Not all applications
Example	WWW/HTTP, FTP, e-mail	BitTorrent, Skype

MODEL QUESTION (3 Marks): Differentiate between the client-server paradigm and the peer-to-peer (P2P) paradigm. Give one example of each.

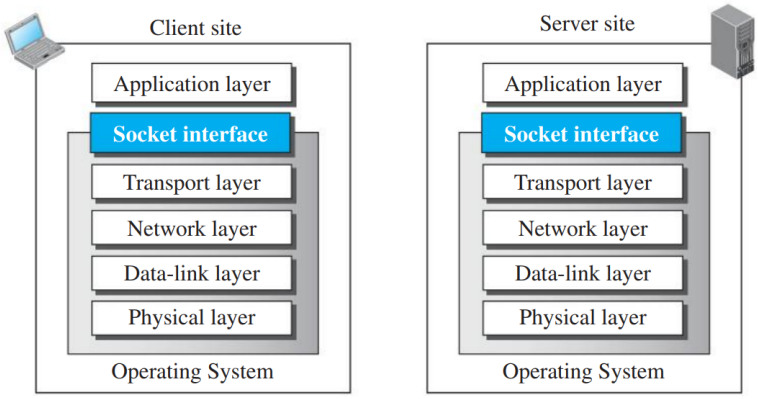
Now a Little bit detailed about client-server:

Client-Server Paradigm

- A **client** is a running program that **starts** communication by sending a request.
- A **server** waits for requests, handles each, and sends back the result.
- The server must be running first. Its lifetime is **infinite**.
- A client has a **finite** lifetime: it sends some requests, gets responses and stops.

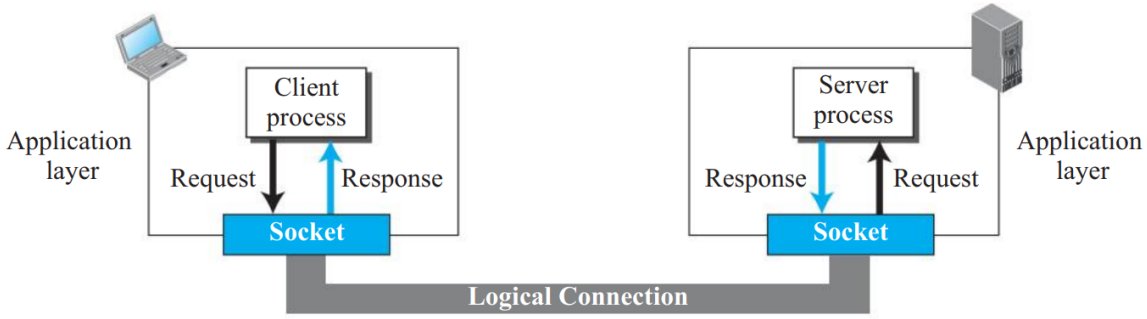
Application Programming Interface and Sockets

- A process needs instructions to tell the lower four layers (inside the operating system) to open a connection, send, receive and close.
- A set of such instructions is an **Application Programming Interface (API)**.
- Common APIs: **socket interface**, **Transport Layer Interface (TLI)** and **STREAM**.
- The **socket interface** (UC Berkeley, early 1980s, UNIX) is the most common.



About Figure: The socket interface sits between the application layer and the operating system, which holds the transport, network, data-link and physical layers.

- Programming languages already read and write other **sources and sinks** (keyboard, monitor, file).
- Sockets add network communication as one more source and sink, using the same read and write instructions.
- A **socket** is not a physical entity. It is an **abstraction**, a data structure created and used by the application.
- For the application, communication is between **two sockets** at the two ends.



About Figure: A client process and a server process each use a socket. The request and response pass between the sockets over the logical connection.

Socket Addresses

Two-way communication needs a **local (sender)** and a **remote (receiver)** address. So a pair of socket addresses is needed.

Socket address = IP address (32 bits) + port number (16 bits)

- The **IP address** identifies the computer.
- The **port number** identifies the process on that computer.



About Figure: A socket address has a 32-bit IP address and a 16-bit port number.
Analogy: a telephone number (IP address) plus an extension (port number).

Finding socket addresses

Site	Address	How it is obtained
Server	Local	Given by the OS (IP address). Standard servers have **well-known ports** (HTTP = 80). Others get a port chosen by the designer
Server	Remote	Taken from the **client's request packet**. It changes with each client
Client	Local	OS gives the IP address. The port is a temporary **ephemeral port number**, unused by any other running client
Client	Remote	Port is well known. The IP address is found from the server name using **DNS** (name to IP address, like a telephone directory). When testing own programs, the programmer supplies both

Using the Services of the Transport Layer

Processes use the transport layer for communication. The choice of protocol affects the application's capability.

Parameter	UDP	TCP	SCTP
Service	Connectionless, unreliable, datagram	Connection-oriented, reliable, byte-stream	Connection-oriented, reliable, message-oriented
Connection	None; each datagram independent	**Handshaking** first (sets packet size, buffer size)	Connection-oriented; **multistream** over multiple network-layer connections
Reliability	No resending of lost or corrupted datagrams	Bytes numbered; lost bytes resent; flow and congestion control	Reliable
Message boundaries	**Preserved** (message-oriented)	**Not preserved**	**Preserved**
Analogy	Postal service	Telephone call	Combination of both
Suited to	Small messages where speed and simplicity matter (management, multimedia)	Long messages needing reliability	Applications needing reliability and staying connected despite a failure

WORLD WIDE WEB AND HTTP

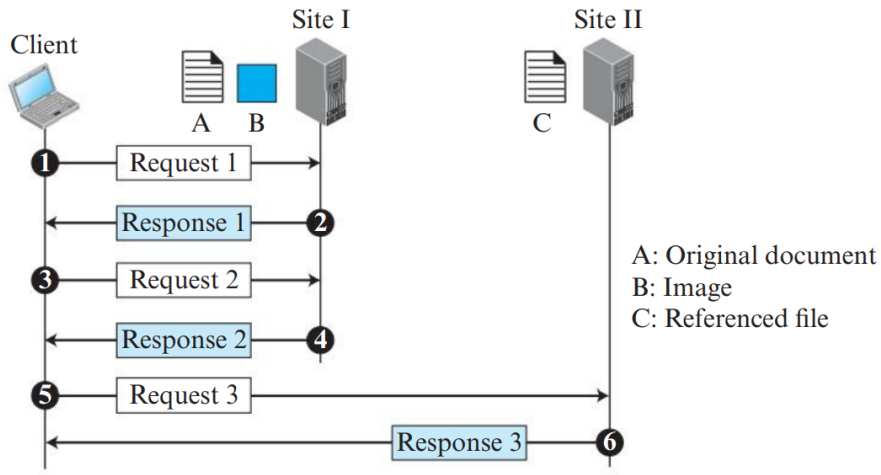
World Wide Web (WWW)

- Proposed by **Tim Berners-Lee in 1989 at CERN**.
- The Web is a **repository of information**. Its documents (**web pages**) are **distributed** worldwide and **linked** together.
- **Distribution** allows growth without overloading a few servers.
- **Linking** is by **hypertext**. The term is now **hypermedia**, since a page can be text, image, audio or video.

Architecture

- The WWW is a **distributed client-server service**. A client uses a **browser** to get service from a server.
- The service is spread over **sites**. Each site holds web pages.
- A **simple** page has no links. A **composite** page has links to other pages.
- Each page is a file with its own name and address.

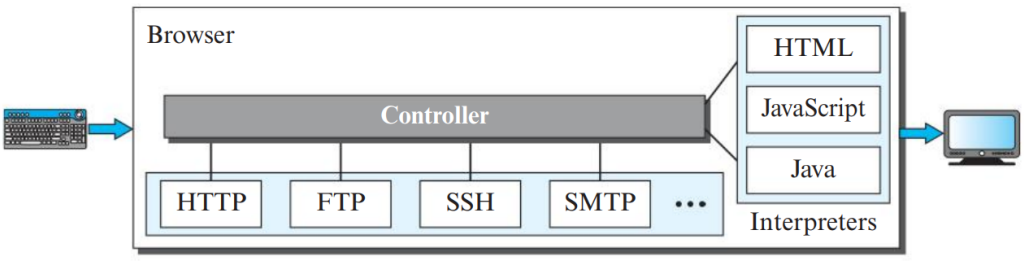
Example: a document (file A) and an image (file B) are on site I. A referenced text file (file C) is on site II. Viewing everything needs **three request/response transactions**. Files A, B and C are independent pages.



About Figure: A client makes three request/response exchanges: for the main document, the image and the referenced file on another site.

Web client (browser)

- **Controller:** takes keyboard or mouse input, uses a client protocol to get the document, then uses an interpreter to display it.
- **Client protocols:** HTTP, FTP, SSH, SMTP.
- **Interpreters:** HTML, Java, JavaScript.
- Examples: Internet Explorer, Netscape Navigator, Firefox.



About Figure: A browser has a controller connected to client protocols (HTTP, FTP, SSH, SMTP) and interpreters (HTML, Java, JavaScript).

Web server

- Stores the pages and sends the requested one.
- Keeps requested files in a **cache** in memory for speed.
- Uses **multithreading or multiprocessing** to serve several requests at once.
- Examples: Apache, Microsoft Internet Information Server.

Uniform Resource Locator (URL)

A web page is identified by four items:

- **Protocol:** program used to fetch the page (usually HTTP, or FTP).
- **Host:** IP address or domain name of the server.
- **Port:** 16-bit number, predefined (HTTP = **80**). Written only if different.
- **Path:** location and name of the file (in UNIX, directories separated by slashes).

protocol://host/path
protocol://host:port/path

Example: http://www.mhhe.com/compsci/forouzan/ (host = www.mhhe.com, path = compsci/forouzan/).

Web documents

Type	Meaning	Details
Static	Fixed content, created and stored at the server	Client gets only a copy. Written in HTML, XML, XSL, XHTML
Dynamic	Created by the server **when requested**	Server runs a program or script. Content varies per request (for example date and time). Earlier CGI; now JSP, ASP, ColdFusion
Active	Program or script that **runs at the client**	Server sends a copy of the program or script. Done with **Java applets** (bytecode) or **JavaScript**

HyperText Transfer Protocol (HTTP)

- HTTP defines how client-server programs fetch web pages.
- The client sends a **request**. The server returns a **response**.
- The server uses **port 80**. The client uses an ephemeral port.
- HTTP uses **TCP** (connection-oriented, reliable). TCP handles errors and lost messages.

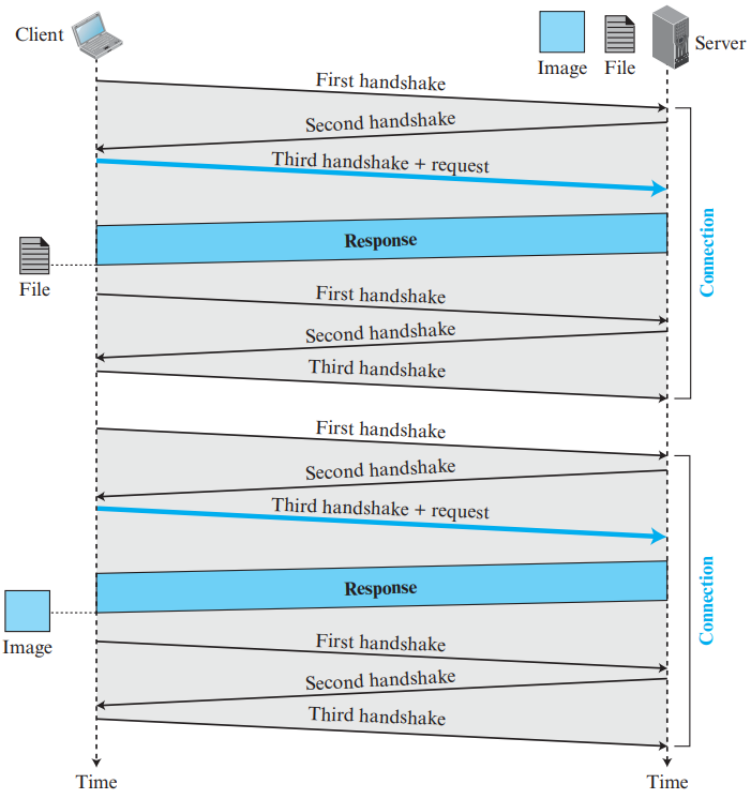
Nonpersistent and persistent connections

Parameter	Nonpersistent	Persistent
Rule	**One TCP connection per** request/response	Connection **left open** for more requests
Version	Before HTTP 1.1	**Default in HTTP 1.1**
Page with N images (same server)	**N + 1** connections	One connection
Overhead	High: repeated handshakes; N + 1 sets of buffers	Low: one set of buffers; handshake time saved
Closing	After each response	On client request or **time-out**

Nonpersistent steps

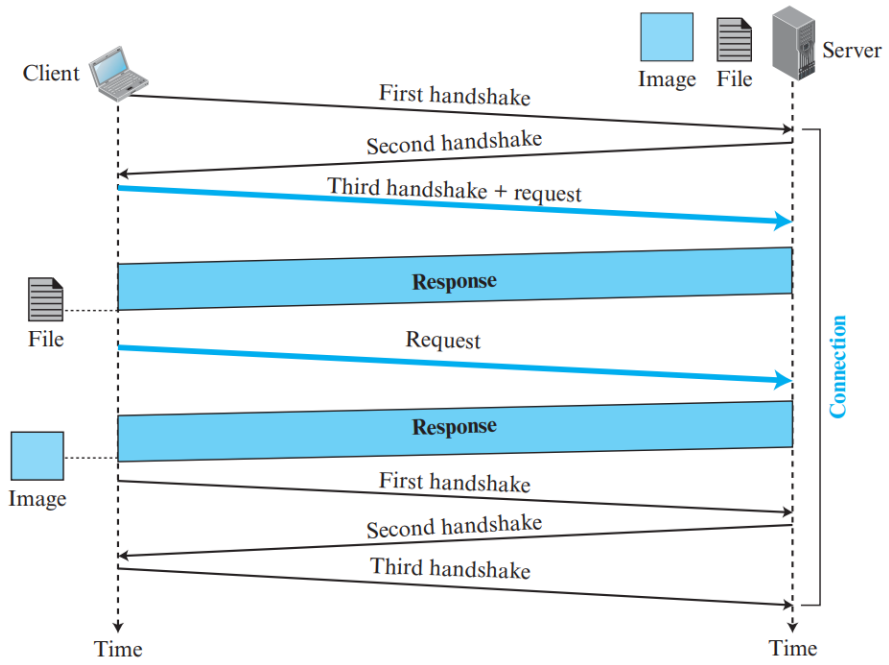
1. The client opens a TCP connection and sends a request.
2. The server sends the response and closes the connection.
3. The client reads until end-of-file and closes the connection.

- Each connection needs three handshake messages (the request rides on the third) and a termination exchange.
- In persistent connections, the server sends the data length. If it is unknown (dynamic or active documents), the server **closes the connection after sending**, so the client knows the data has ended.



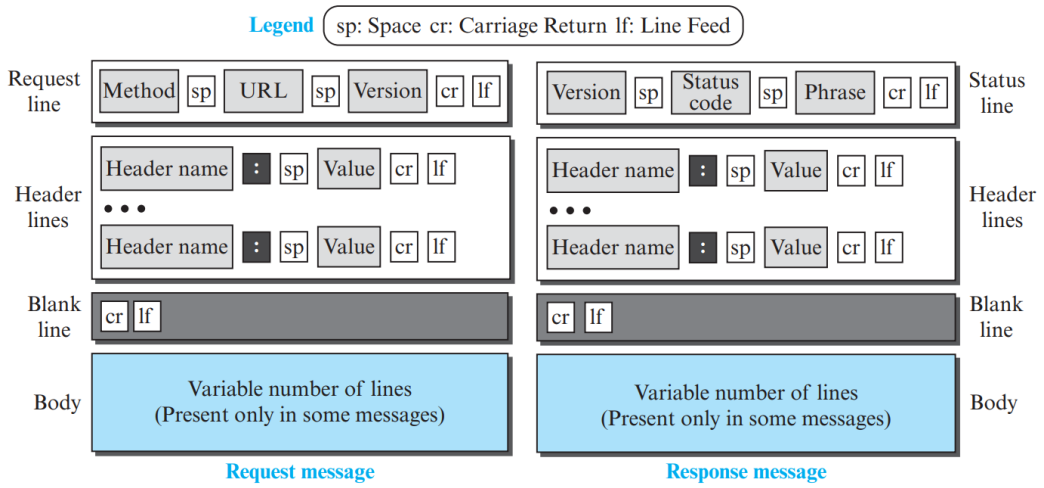
About Figure: A nonpersistent connection: two separate connections (handshake, transfer, termination), one for the file and one for the image.

About Figure: (Below Fig) A persistent connection: one connection is set up once, and the file and image requests are sent over it.



Message formats

Each message has four sections: a **request line** (request) or **status line** (response), **header lines**, a **blank line**, and an optional **body**.



About Figure: The request message (request line, headers, blank line, body) beside the response message (status line, headers, blank line, body).

Request message

- **Request line** has three fields: **method**, **URL**, **version**.
- **Header lines** (zero or more): header name, colon, value.
- **Body:** present mainly for PUT and POST.

Method	Action
GET	Requests a document (most common; no body)
HEAD	Requests information about a document only
PUT	Sends a document from client to server
POST	Sends information to the server to add to or modify a page
TRACE	Echoes the incoming request (debugging)
DELETE	Removes a web page

Method	Action
CONNECT	Reserved
OPTIONS	Inquires about available options

Request header	Meaning
User-agent	Identifies the client program
Accept	Media format the client accepts
Accept-charset	Character set accepted
Accept-encoding	Encoding scheme accepted
Accept-language	Language accepted
Authorization	Client's permissions
Host	Host and port of the client
Date	Current date
Upgrade	Preferred communication protocol
Cookie	Returns the cookie to the server
If-Modified-Since	Send the file only if modified since a date

Response message

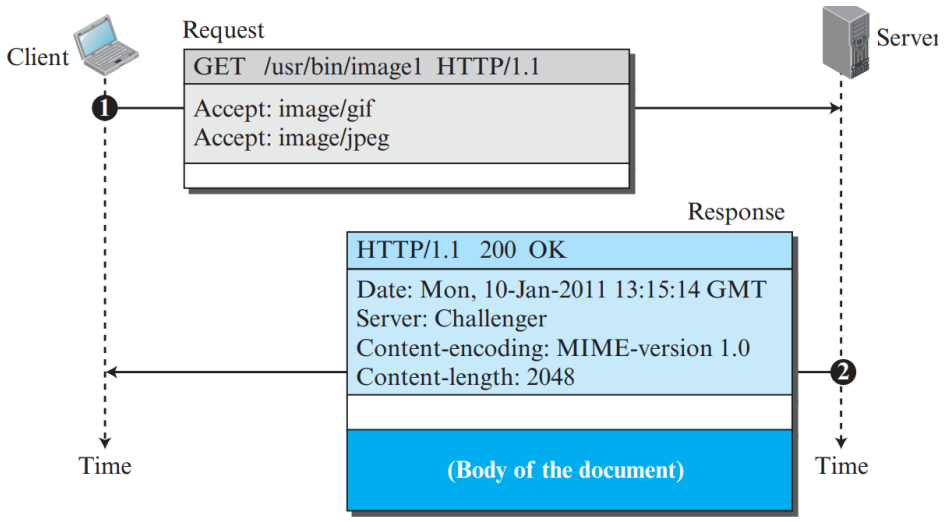
- **Status line** has three fields: **version**, **status code** (three digits), **status phrase**.

Code range	Meaning
100s	Informational
200s	Successful request
300s	Redirection to another URL
400s	Error at the client
500s	Error at the server

Response header	Meaning
Date	Current date
Upgrade	Preferred communication protocol
Server	Information about the server
Set-Cookie	Server asks the client to save a cookie
Content-Encoding	Encoding scheme
Content-Language	Language
Content-Length	Length of the document
Content-Type	Media type
Location	Asks the client to send the request to another site
Accept-Ranges	Server accepts byte-range requests
Last-modified	Date and time of last change

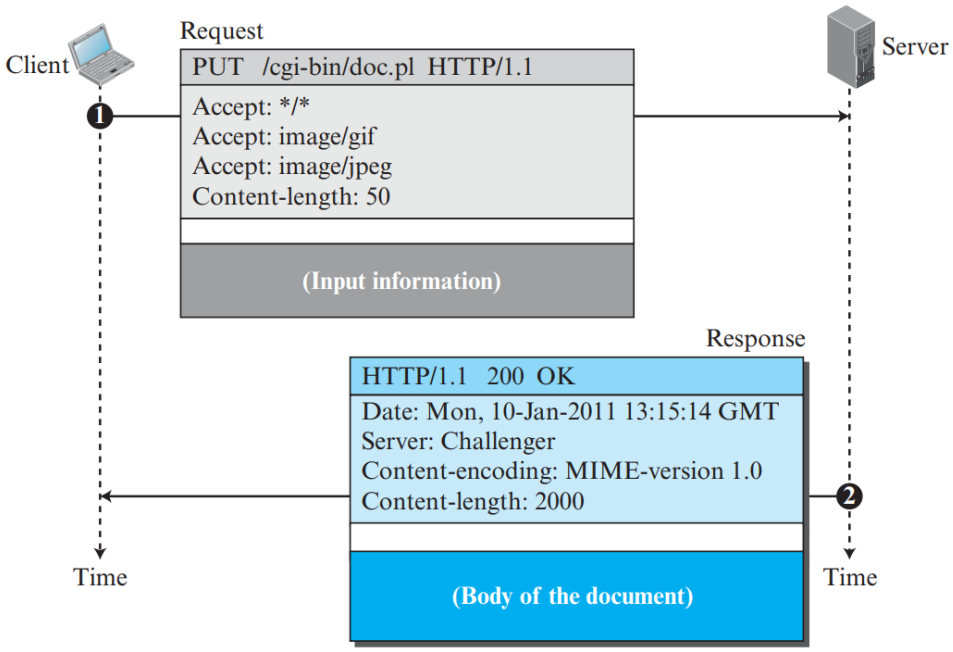
- The **body** holds the document and is absent for error messages.

Example (GET): request line GET /usr/bin/image1 HTTP/1.1 with Accept: image/gif and Accept: image/jpeg, no body. The response is HTTP/1.1 200 OK, with headers for date, server, content-encoding and content-length, followed by the body.



About Figure: A GET request for an image and the 200 OK response carrying the image.

Example (PUT): the client posts a web page using PUT. The request body holds the page, and the response carries the created document.



About Figure: A PUT request with its body and the server's response.

<u>Conditional request</u>

- The client adds a condition, usually the header **If-Modified-Since**.
- If the page has not changed, the server replies **304 Not Modified** with an empty body.

Cookies

- The Web was designed as **stateless**: the server forgets the client after each response.
- Many uses need memory: electronic shops (carts), registered-only sites, portals, and advertising agencies.
- The **cookie** mechanism solves this.

Creating and storing cookies

1. The server stores information about the client (domain name, cookie contents, timestamp).
2. The server sends the cookie in the response (**Set-Cookie** header).
3. The browser stores it in its **cookie directory**, sorted by server domain name.

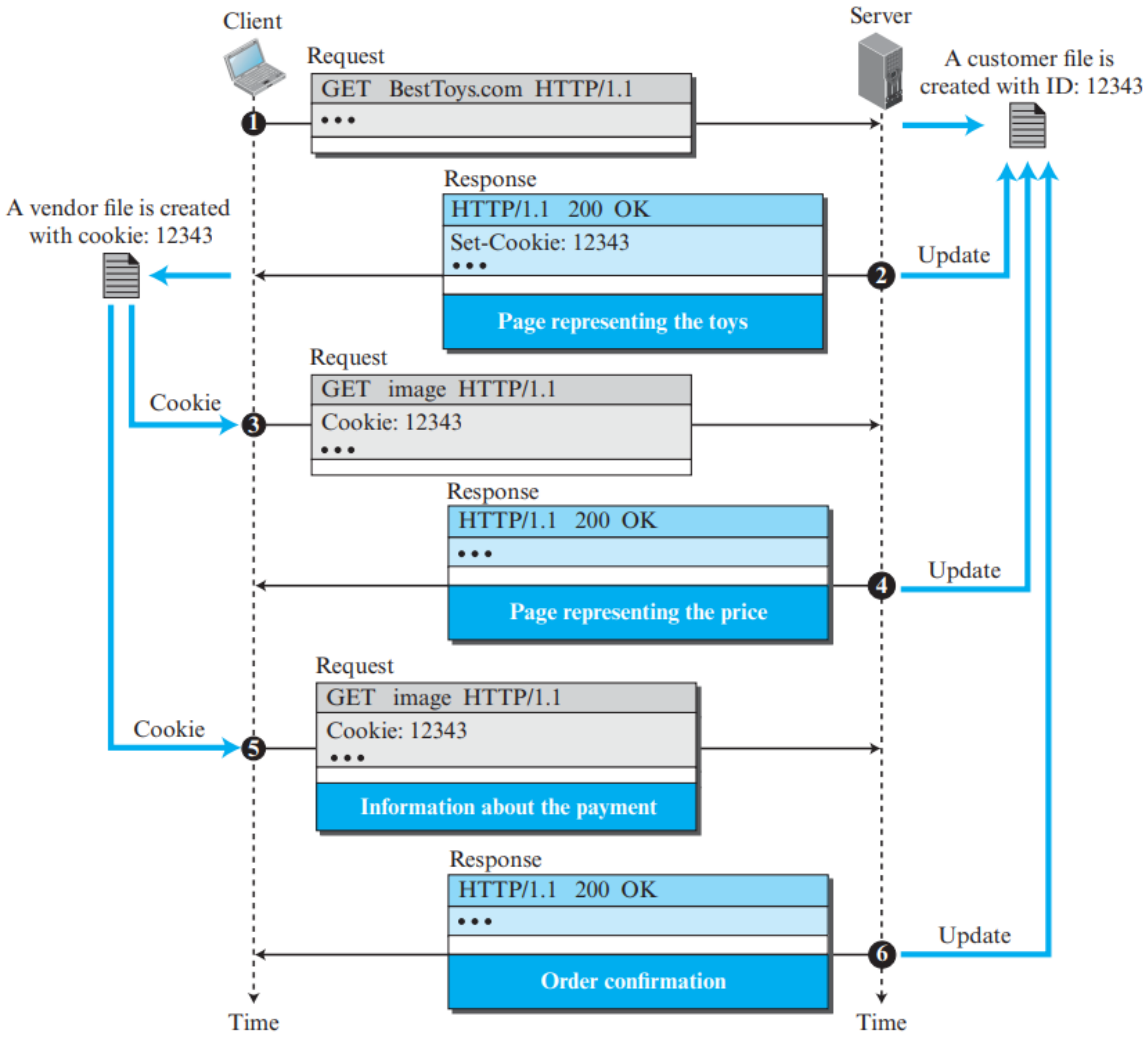
Using cookies

- Before sending a request, the browser looks for a cookie from that server and, if found, sends it (**Cookie** header).
- The server then knows it is an **old client**.
- The browser never reads the cookie. It is *made by the server and eaten by the server.*

Uses

- **E-commerce:** the cookie records the cart items and prices; the last cookie gives the total at checkout.
- **Registered-only sites:** a cookie is issued at registration, and only clients sending it are allowed.
- **Portals:** the cookie records the user's favourite pages.
- **Advertising agencies:** a cookie with the user's ID builds a profile of browsing behaviour. This raises **privacy** concerns.

Example (BestToys): the server makes a cart with ID 12343 and sends Set-Cookie: 12343. Each later request carries Cookie: 12343, so the server finds the cart, adds the toy, takes payment and confirms the order.



About Figure: Successive requests and responses between client and BestToys server, with the cookie 12343 set once and returned with every later request.

Web Caching: Proxy Server

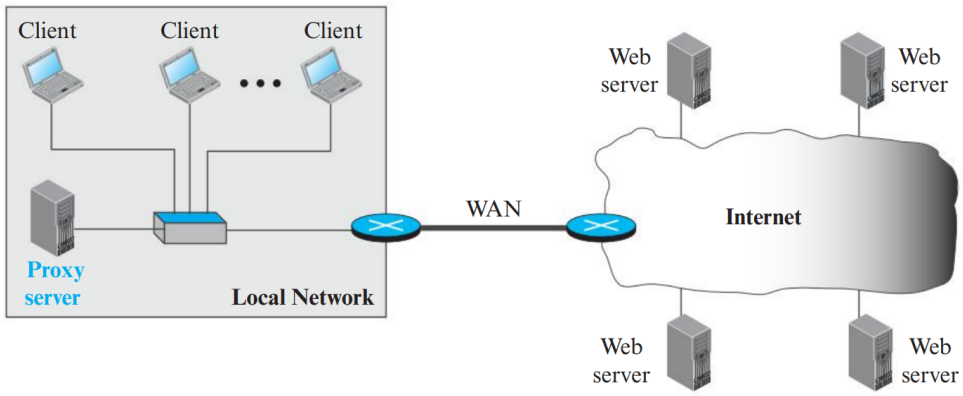
A **proxy server** keeps **copies of responses to recent requests**.

1. The client (configured to use the proxy) sends its request to the proxy.
2. If the response is in the cache, the proxy sends it.
3. If not, the proxy acts as a **client** and requests it from the target server, stores the response, and then acts as a **server** and sends it to the client.

Benefits: less load on the original server, less traffic, better latency.

Location: at the client site, in a hierarchy: (1) the client computer itself, (2) a company LAN, (3) an ISP.

Cache update: keep a list of sites whose content stays the same for a while (for example a news page updated each morning), or use headers showing the last modification time.



Explanation of the Figure: Clients on a local network send requests to a proxy server. The proxy fetches pages from web servers on the Internet only when it has no copy.

HTTP Security

HTTP alone has no security. HTTP over the **Secure Socket Layer (SSL)** is **HTTPS**. It gives **confidentiality, client and server authentication, and data integrity**.

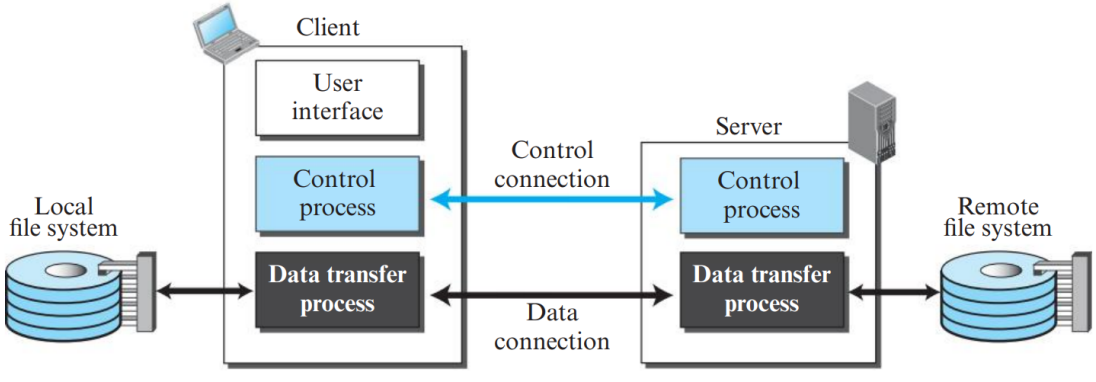
FTP & EMAIL

File Transfer Protocol (FTP)

FTP is the standard TCP/IP protocol for **copying a file from one host to another**.

Problems it solves: the two systems may use different **file-name conventions**, different **data representations** and different **directory structures**. HTTP can also transfer files, but FTP is better for **large files** and **different formats**.

FTP Model



About Figure: The client (user interface, control process, data transfer process) and the server (control process, data transfer process) are joined by a control connection and a data connection. Each side also has a local or remote file system.

- The **client** has three parts: **user interface**, **client control process** and **client data transfer process**.
- The **server** has two parts: **server control process** and **server data transfer process**.
- The **control connection** joins the two control processes.
- The **data connection** joins the two data transfer processes.

Why two connections?

Commands and data have different needs.

- Control connection: very simple rules; one line of command or response at a time.
- Data connection: more complex rules, because of the variety of data types.

Lifetimes

- **Control connection:** stays open for the **whole FTP session**.
- **Data connection:** opened and closed **for each file transfer**. Many data connections can occur in one session.

Well-known ports:

21 for the control connection, **20** for the data connection.

Control Connection

- Uses the same approach as TELNET: **NVT ASCII** characters.
- Communication is by **commands** and **responses**, one at a time.
- Each line ends with a two-character (carriage return, line feed) token.
- **Commands** go from client to server: uppercase ASCII, with an optional argument.
- **Responses** go from server to client.

Command	Argument	Description
ABOR		Abort the previous command
CDUP		Change to parent directory
CWD	Directory name	Change to another directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
MKD	Directory name	Create a new directory
PASS	User password	Password
PASV		Server chooses a port
PORT	Port identifier	Client chooses a port
PWD		Display current directory
QUIT		Log out
RETR	File name(s)	Retrieve files (server to client)
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify file to be renamed
RNTO	File name (new)	Rename the file
STOR	File name(s)	Store files (client to server)
STRU	F, R or P	Data organisation (file, record, page)
TYPE	A, E or I	File type (ASCII, EBCDIC, image)

Command	Argument	Description
USER	User ID	User information
MODE	S, B or C	Transmission mode (stream, block, compressed)

Responses

- Every command gets at least one response: a **three-digit code** plus text.
- **First digit:** status of the command.
- **Second digit:** the area to which the status applies.
- **Third digit:** additional information.

Code	Description	Code	Description
125	Data connection open	250	Requested file action OK
150	File status OK	331	User name OK; password needed
200	Command OK	425	Cannot open data connection
220	Service ready	450	File action not taken; file unavailable
221	Service closing	452	Action aborted; insufficient storage
225	Data connection open	500	Syntax error; unrecognised command
226	Closing data connection	501	Syntax error in parameters or arguments
230	User login OK	530	User not logged in

Data Connection

The data connection uses **port 20** at the server. It is created differently from the control connection:

1. The **client**, not the server, issues a **passive open** using an **ephemeral port**. The client does this because it issues the file-transfer commands.
2. The client sends this port number to the server using the **PORT** command.
3. The server issues an **active open** from well-known port 20 to the received ephemeral port.

Communication over the data connection. Before sending a file, the client prepares the transfer through the control connection. The differences between systems are handled by three attributes:

Attribute	Options
Data structure	**File structure** (default; continuous stream of bytes), **record structure** (file divided into records; text files only), **page structure** (pages with page number and header; random or sequential access)
File type	**ASCII file**, **EBCDIC file**, **image file**
Transmission mode	**Stream mode** (default; continuous byte stream to TCP), **block mode** (blocks, each with a 3-byte header: 1 byte block descriptor + 2 bytes block size), **compressed mode**

MODEL QUESTION (3 Marks): Describe the roles of the control connection and the data connection in FTP file transfer.

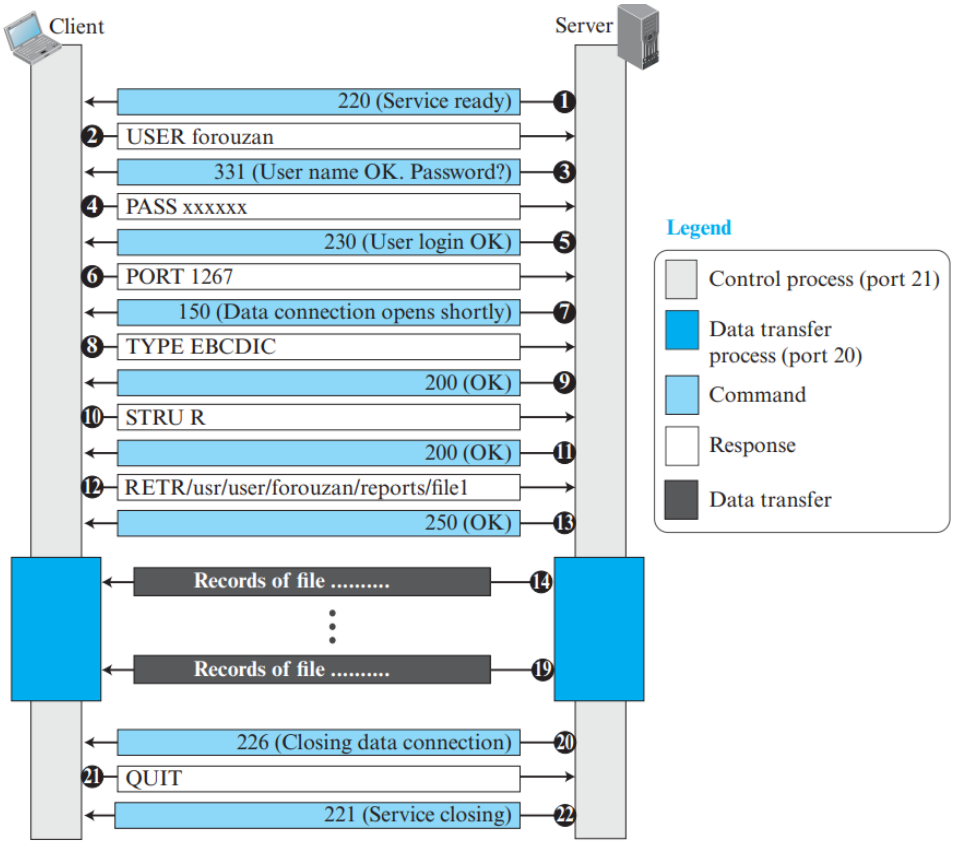
File Transfer

File transfer happens over the data connection, under the commands sent on the control connection. It means one of three things:

- **Retrieving a file** (server to client).
- **Storing a file** (client to server).
- **Directory listing** (server to client).

Example (retrieving a file):

Step	Client command	Server response
Connect		220 (Service ready)
Login	USER forouzan	331 (User name OK, password?)
	PASS xxxxxx	230 (User login OK)
Set type	TYPE EBCDIC	200 (OK)
Set structure	STRU R	200 (OK)
Data port	PORT 1267	150 (Data connection opens shortly)
Retrieve	RETR (file path)	File sent record by record, then 226 (Closing data connection)
Quit	QUIT	221 (Service closing)



About Figure: The control connection (port 21) stays open throughout. The data connection (port 20) opens for the file transfer and closes afterwards. Numbered arrows show the commands, responses and data.

Security for FTP

- FTP was designed when security was not a concern.
- The **password is sent as plaintext**, and the **data is also plaintext**, so both can be intercepted.
- Security is added by putting a **Secure Socket Layer (SSL)** between FTP and TCP. This is called **SSL-FTP**.

Electronic Mail (E-mail)

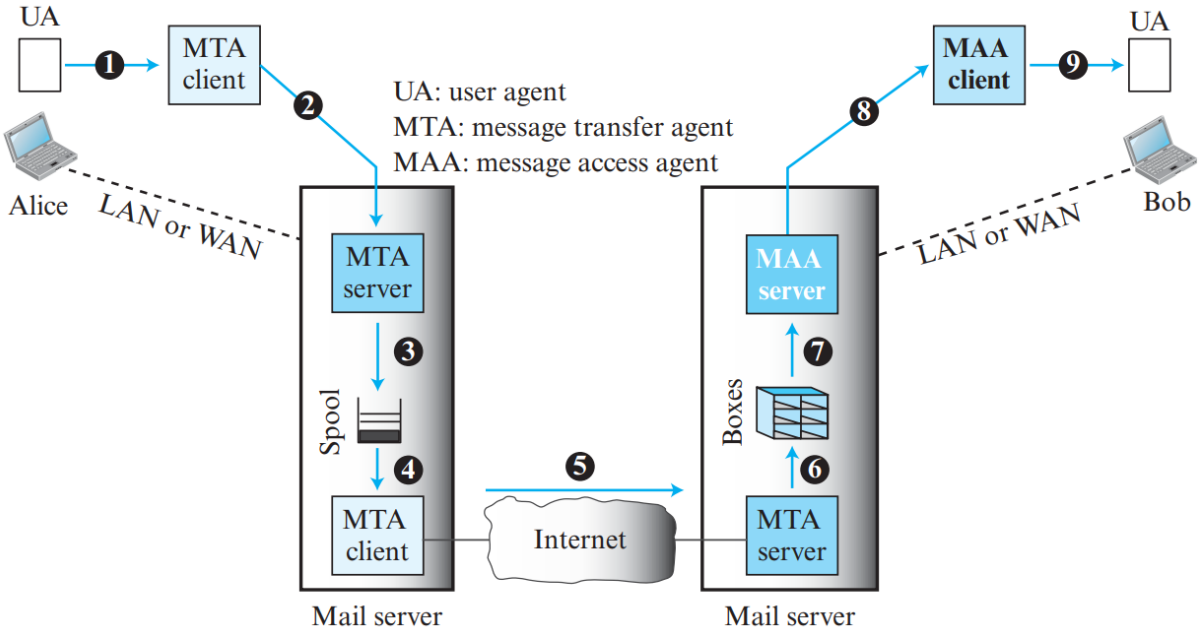
E-mail is different from HTTP or FTP:

- It is a **one-way transaction**. A reply is a new one-way transaction.
- The receiver cannot run a server all the time and wait for mail, because his computer may be off.
- So **intermediate mail servers** are used, and the client-server idea is applied in stages.

Architecture of E-mail

Common scenario: Alice sends an e-mail to Bob. Each is connected through a LAN or WAN to a **mail server**.

- Each user has a **mailbox** on the server. It is a special file with access restricted to its owner.
- The server keeps a **queue (spool)** of messages waiting to be sent.
- Three agents are used: **User Agent (UA)**, **Message Transfer Agent (MTA)** and **Message Access Agent (MAA)**.
- A simple e-mail takes **nine steps**.



About Figure: Alice's UA sends the message to her mail server (steps 1–2), where it waits in the spool. An MTA client sends it over the Internet to the MTA server of Bob's mail server (steps 3–5), which puts it in Bob's mailbox. Bob's MAA client pulls it from the MAA server (steps 6–9) and his UA shows it.

Agent	Role
User Agent (UA)	Program that **composes, reads, replies to and forwards** messages and handles local mailboxes on the user's computer
Message Transfer Agent (MTA)	Client-server pair that **transfers (pushes) the message** from the sender's site to the receiver's mail server. Protocol: **SMTP**
Message Access Agent (MAA)	Client-server pair that lets the receiver **pull (retrieve) mail** from his mailbox on the mail server. Protocols: **POP3, IMAP4**

Two important points

1. Bob cannot use the MTA server directly, because he would have to keep his computer on all the time.
2. MTA is a **push** protocol. Bob needs a **pull** protocol, so a separate MAA client-server pair is needed.

The e-mail system needs **two UAs, two pairs of MTAs (client and server) and one pair of MAAs (client and server)**.

User Agent

Types of user agents

- **Command-driven:** early agents using one-character commands (r = reply, R = reply to all). Examples: mail, pine, elm.
- **GUI-based:** icons, menus and windows. Examples: Eudora, Outlook.

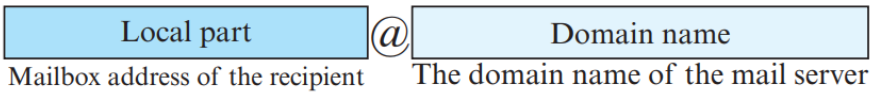
Sending mail. The UA creates a message like postal mail, with an **envelope** and a **message**.

- **Envelope:** sender address, receiver address and other information.
- **Message:** a **header** (sender, receiver, subject, date) and a **body** (the information to be read).

Receiving mail. The UA is triggered by the user or a timer. It gives a notice and shows a list with one summary line per message (sender address, subject, time). The user selects a message to read.

E-mail Addresses and Mailing Lists

An Internet e-mail address has two parts separated by @:



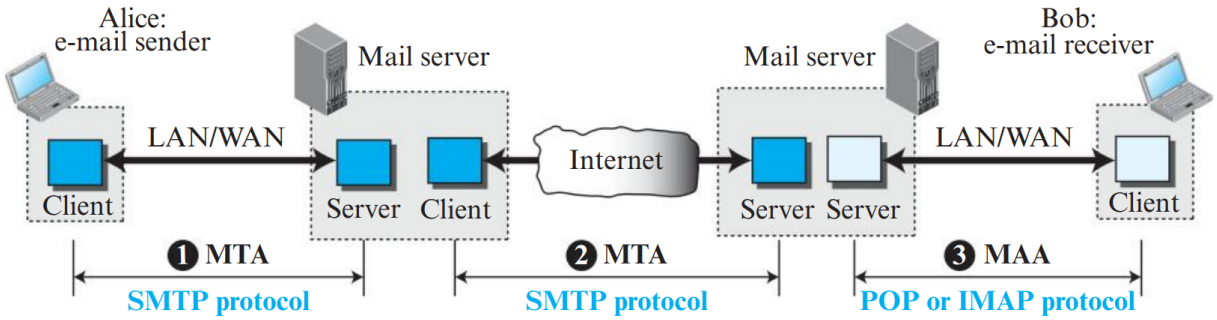
About Figure: The local part is the mailbox address of the recipient. The domain name is that of the mail server (exchanger).

- **Local part:** name of the **user mailbox**, the special file where the user's mail is stored for the MAA to retrieve.
- **Domain name:** the mail server or **mail exchanger** of the organisation. It comes from the **DNS database** or is a logical name.

Mailing list (group list): one **alias** stands for several e-mail addresses. The system checks the recipient name against the alias database. If a list exists, a **separate message** is prepared for each entry and handed to the MTA.

Message Transfer Agent: SMTP

E-mail uses three client-server pairs: two MTAs (UA to its mail server, and server to server) and one MAA (mail server to receiving UA).



About Figure: SMTP is used between Alice and her mail server and between the two mail servers. POP or IMAP is used between Bob's mail server and Bob.

Simple Mail Transfer Protocol (SMTP): It is the formal protocol for the MTA client and server. It uses **commands** (client to server) and **responses** (server to client), each ending with carriage return and line feed.

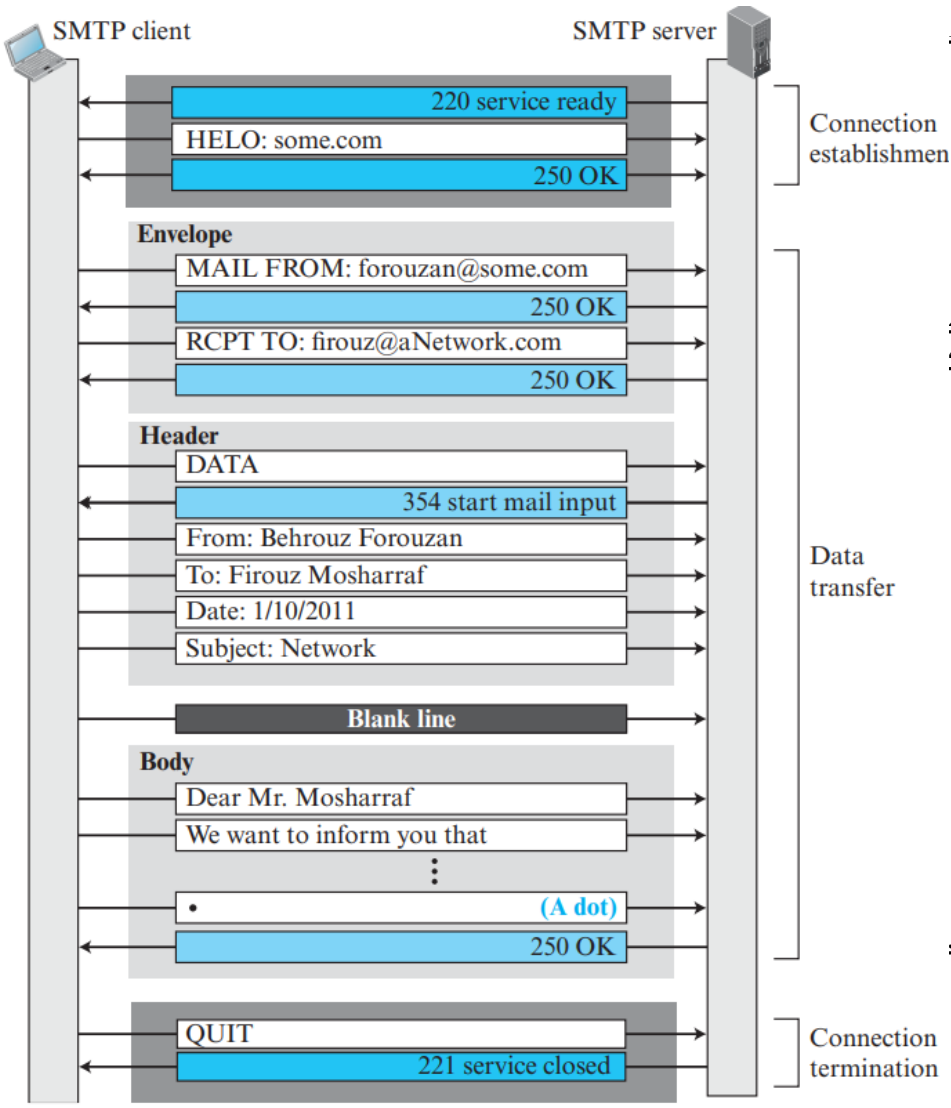
SMTP commands have the format Keyword: argument(s). SMTP defines 14 commands:

Keyword	Argument	Description
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender
RCPT TO	Intended recipient	Identifies the recipient
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current transaction
VRFY	Name of recipient	Verifies the recipient's address
NOOP		Checks the status of the recipient
TURN		Switches sender and recipient
EXPN	Mailing list	Expands the mailing list
HELP	Command name	Gives help on the command
SEND FROM	Intended recipient	Deliver only to the recipient's terminal, not mailbox
SMOL FROM	Intended recipient	Deliver to the terminal or the mailbox
SMAL FROM	Intended recipient	Deliver to the terminal and the mailbox

SMTP responses are three-digit codes, optionally followed by text:

Category	Codes
Positive completion	211 system status or help reply; 214 help message; 220 service ready; 221 service closing; 250 request completed; 251 user not local, will be forwarded
Positive intermediate	354 start mail input
Transient negative	421 service not available; 450 mailbox not available; 451 command aborted, local error; 452 insufficient storage
Permanent negative	500 syntax error; 501 error in parameters; 502 command not implemented; 503 bad sequence of commands; 504 command parameter not implemented; 550 mailbox unavailable; 551 user not local; 552 exceeded storage; 553 mailbox name not allowed; 554 transaction failed

Mail transfer phases (after the client makes a TCP connection to port 25)



1. Connection establishment

1. Server sends **220** (service ready), or **421** if not ready.
2. Client sends **HELO** with its domain name.
3. Server responds **250** (or another code).

***2. Message transfer* (steps 3 and 4 repeat for each recipient)**

1. Client sends **MAIL FROM** (sender's address, used for error reports).
2. Server responds **250**.
3. Client sends **RCPT TO** (recipient's address).
4. Server responds **250**.
5. Client sends **DATA**.
6. Server responds **354** (start mail input).
7. Client sends the message lines. A line with **only a period** ends the message.
8. Server responds **250**.

3. Connection termination

1. Client sends **QUIT**.
2. Server responds **221**.

About Figure: The SMTP client and server exchange HELO, MAIL FROM, RCPT TO, DATA (with header and body), and QUIT, with the responses 220, 250, 354 and 221. The steps repeat twice per mail: sender to local server, and local server to remote server.

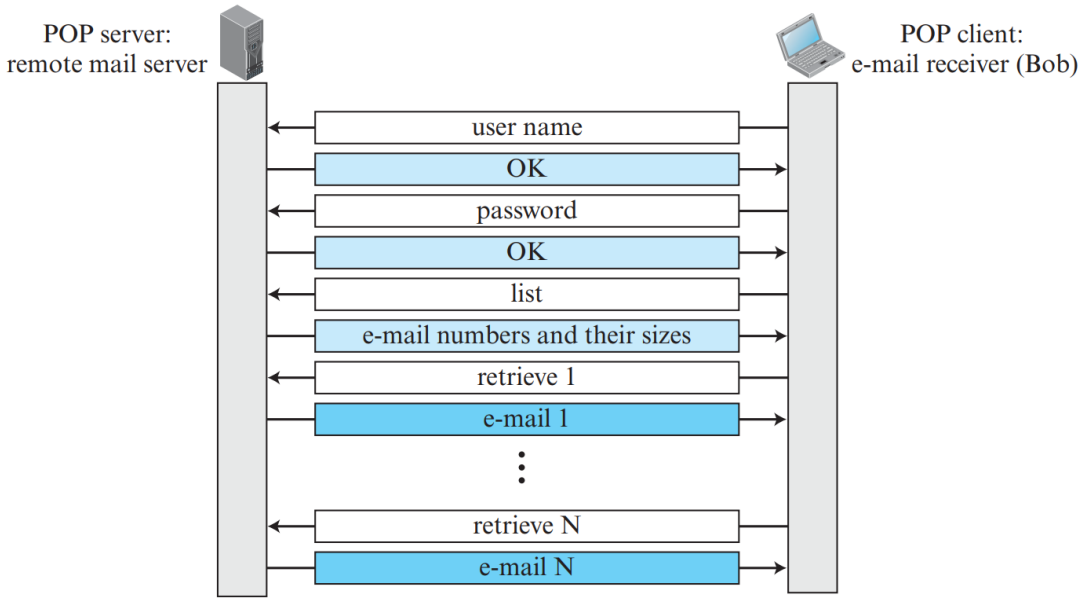
Message Access Agent: POP3 and IMAP4

SMTP is a **push** protocol: the bulk data flows from client to server. The last stage needs a **pull** protocol, so a message access protocol is used. Two are available: **POP3** and **IMAP4**.

1. POP3 (Post Office Protocol, version 3)

- Simple but limited. Client software is on the receiver's computer; server software is on the mail server.
- The client opens a connection to the server on **TCP port 110**.
- It sends the **user name and password**, then can **list** and **retrieve** messages one by one.

About Figure: (Below): The POP client (Bob) sends the user name and password, then LIST, and receives the numbers and sizes of e-mails. It then retrieves messages one by one from the POP server (the remote mail server).



There is 2 POP3 Modes:

POP3 mode	Behaviour	Used when
Delete mode	Mail is deleted from the mailbox after each retrieval	User works at her permanent computer
Keep mode	Mail stays in the mailbox after retrieval	User reads mail away from the main computer (for example a laptop)

2. IMAP4 (Internet Mail Access Protocol, version 4)

- Similar to POP3 but with **more features**; more powerful and more complex.
- POP3 **cannot** organise mail on the server (no folders) and **cannot** check contents before download.

IMAP4 adds these functions:

- Check the e-mail **header** before downloading.
- **Search** the contents for a string of characters before downloading.
- **Partially download** an e-mail (useful when bandwidth is limited and the mail has multimedia).
- **Create, delete or rename** mailboxes on the server.
- Create a **hierarchy of mailboxes** (folders).

Parameter	POP3	IMAP4
Port	110	(not given in the text)
Complexity	Simple, limited	More powerful, complex
Folders on server	Not possible	Possible
Check header or search before download	No	Yes
Partial download	No	Yes

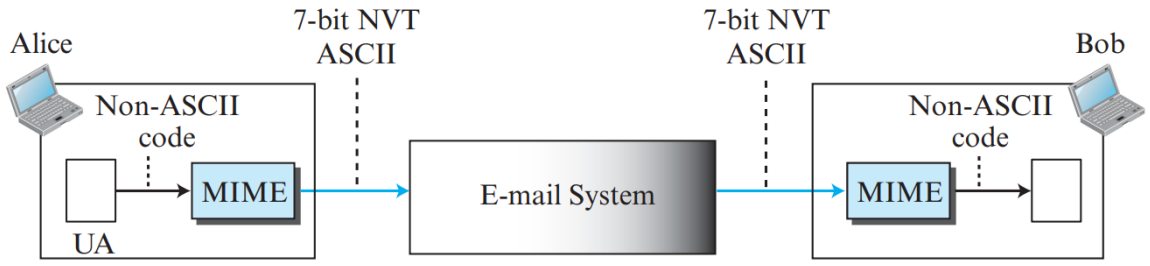
MODEL QUESTION (6 Marks): With a neat diagram, explain the architecture of electronic mail. Describe the roles of the user agent, message transfer agent and message access agent. Explain the use of SMTP, POP3 and IMAP in the delivery of a mail message.

MIME

Basic e-mail can send only **NVT 7-bit ASCII**. So it cannot carry languages other than English (French, Hebrew, Chinese and so on), or **binary files, audio or video**.

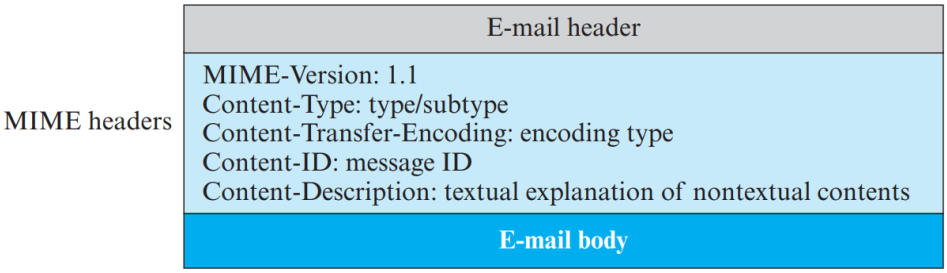
Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows **non-ASCII data** to be sent through e-mail.

- At the sender, MIME converts non-ASCII data to NVT ASCII and gives it to the MTA client.
- At the receiver, it converts it back to the original data.



About Figure: MIME sits between the user agent and the e-mail system at both ends. It converts non-ASCII code to 7-bit NVT ASCII before sending and reverses this at the receiver.

MIME headers. Five headers are added to the e-mail header:



About Figure: The MIME headers appear after the normal e-mail header and before the body: MIME-Version, Content-Type, Content-Transfer-Encoding, Content-ID and Content-Description.

Header	Meaning
MIME-Version	Version of MIME used (current: 1.1)
Content-Type	Type of data in the body, written as type/subtype
Content-Transfer-Encoding	Method used to encode the message into 0s and 1s for transport
Content-ID	Uniquely identifies the whole message in a multiple-message environment
Content-Description	Tells whether the body is image, audio or video

Content types (seven)

Type	Subtype	Description
Text	Plain	Unformatted
	HTML	HTML format
Multipart	Mixed	Ordered parts of different data types
	Parallel	Same as mixed, but no order
	Digest	Similar to mixed; default is message/RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message

Type	Subtype	Description
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG, GIF	Image in that format
Video	MPEG	Video in MPEG format
Audio	Basic	Single-channel encoding of voice at 8 kHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (8-bit bytes)

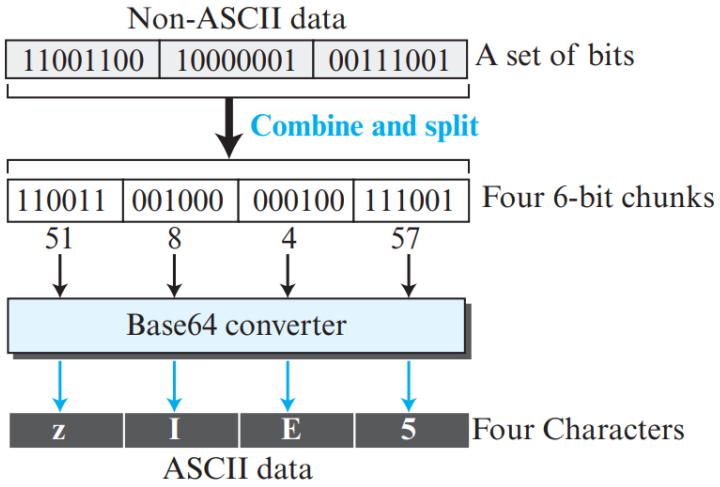
Content-Transfer-Encoding methods

Type	Description
7-bit	NVT ASCII characters, each line under 1000 characters
8-bit	Non-ASCII characters, each line under 1000 characters
Binary	Non-ASCII characters, unlimited line length
Base64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters encoded as an equal sign plus the ASCII code

Base64

1. The bit string is divided into **6-bit chunks**.
2. Each 6-bit value is converted to an ASCII character using the Base64 table (0–25 = A–Z, 26–51 = a –z, 52–61 = 0–9, 62 = +, 63 = /).
3. Every 6 bits become an **8-bit** character, so the overhead is **25 percent**.

Example: the 24 bits 11001100 10000001 00111001 become 110011 001000 000100 111001 = 51, 8, 4, 57 = **z I E 5**.

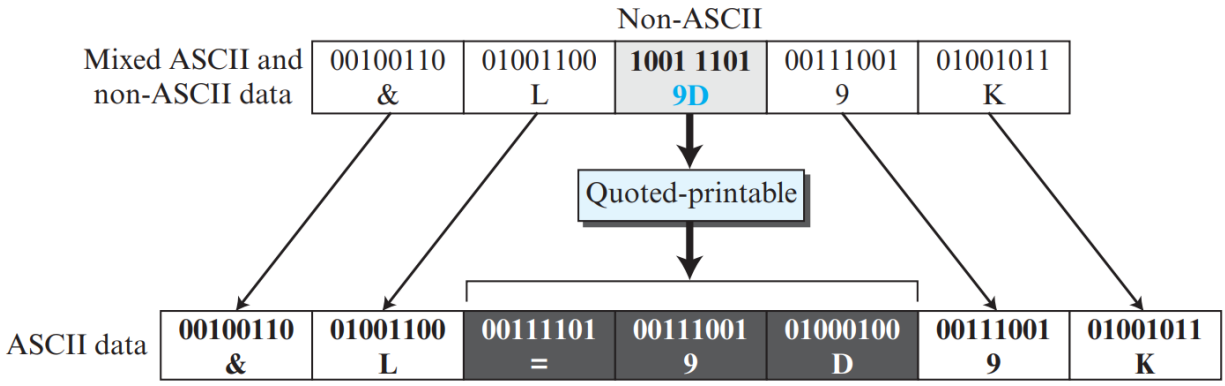


About Figure: Three bytes of non-ASCII data are combined and split into four 6-bit chunks. A Base64 converter turns each chunk into one ASCII character.

Quoted-printable

- Used when the data is mostly ASCII with only a small non-ASCII part.
- An **ASCII character** is sent as it is.
- A **non-ASCII character** is sent as **three characters**: = followed by the two **hexadecimal digits** of the byte.

Example: bytes 01001100 (L), 10011101 (non-ASCII = 9D hex) and 00111001 (9) are sent as **L =9D 9** (the middle byte becomes =, 9, D).

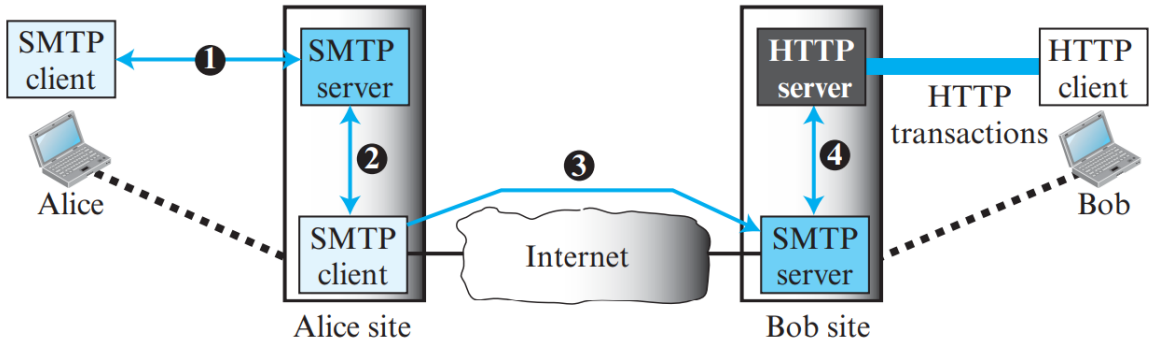


About Figure: In mixed data, ASCII bytes are sent unchanged. The non-ASCII byte 9D (it starts with bit 1) is replaced by the three ASCII characters =, 9 and D.

Web-Based Mail

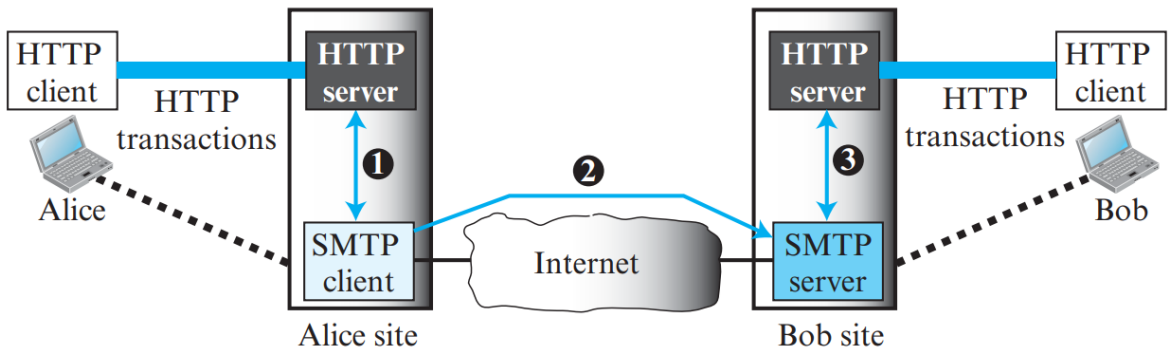
Sites such as Hotmail, Yahoo and Gmail give e-mail service to anyone who visits.

Case I: only the receiver uses web mail



- Alice's browser to her mail server: **SMTP**.
- Between the two mail servers: **SMTP**.
- From Bob's web server to Bob's browser: **HTTP** (instead of POP3 or IMAP4).
- Bob sends an HTTP request. The site returns a form for **login name and password**. After a match, the e-mail list comes in HTML, and each e-mail is fetched by further HTTP transactions.

Case II: both sender and receiver use web mail



- Alice sends the message to her web server using **HTTP** (Bob's mailbox address is the URL).
- Her server passes it to its SMTP client, which sends it to Bob's server by **SMTP**.
- Bob reads it by **HTTP**.

E-mail Security

The protocols above have **no security** provisions. E-mail can be secured by two application-layer protocols designed for it:

- **Pretty Good Privacy (PGP)**
- **Secure/Multipurpose Internet Mail Extensions (S/MIME)**

DNS & P2P NETWORKS

Domain Name System (DNS)

TCP/IP identifies a host by its **IP address**, but people prefer **names**. A **directory system** is needed to map a name to an address, like a telephone directory.

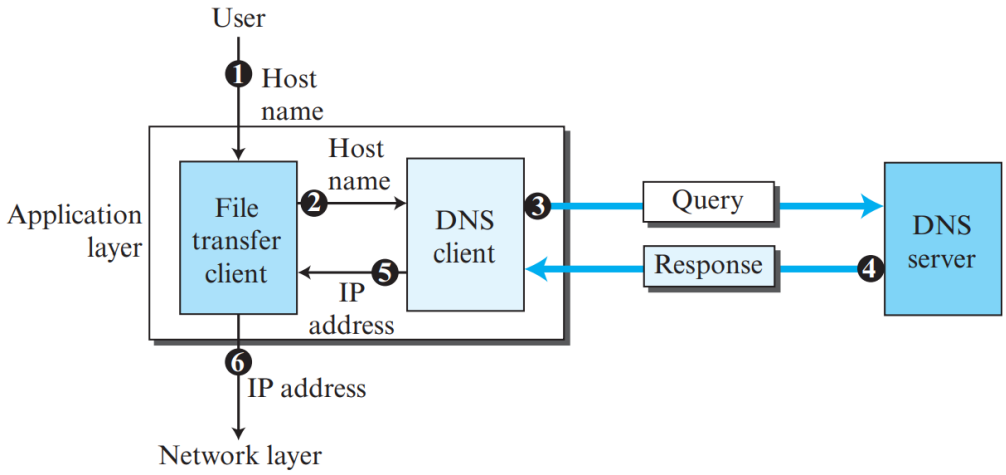
- A single central directory is **not practical**: the Internet is huge, and if the central computer failed, communication would collapse.
- DNS therefore **distributes** the information among many computers.
- A host that needs a mapping contacts the **closest** computer holding it.

Purpose of DNS

DNS is a client-server application that **helps other application programs**. Example: a user wants an FTP server but knows only its name, afilesource.com.

1. The user passes the host name to the file transfer client.
2. The file transfer client passes it to the **DNS client**.
3. Every computer knows the address of **one DNS server** after booting. The DNS client sends a query to it with the server name.
4. The DNS server replies with the **IP address**.
5. The DNS client passes the IP address to the file transfer client.
6. The file transfer client uses the IP address to contact the file transfer server.

Two connections are needed: one to **map the name**, one to **transfer the file**.



About Figure: The file transfer client hands the host name to the DNS client (steps 1–2). The DNS client queries the DNS server and receives the IP address (steps 3–4). The address goes back to the file transfer client (steps 5–6).

Name Space

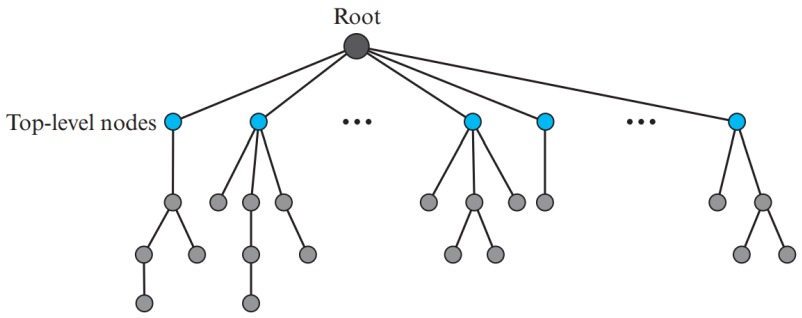
Names must be **unique**, since addresses are unique. A name space can be flat or hierarchical.

Parameter	Flat name space	Hierarchical name space
Structure	A name is a sequence of characters without structure	A name has several parts (nature of organisation, organisation name, department, host)
Control	Must be **centrally controlled** to avoid duplicates	Authority can be **decentralised**
Suitability	Not usable for a large system like the Internet	Used by DNS

Example: two organisations may each name a computer caesar. With first.com and second.com, the full names caesar.first.com and caesar.second.com stay unique.

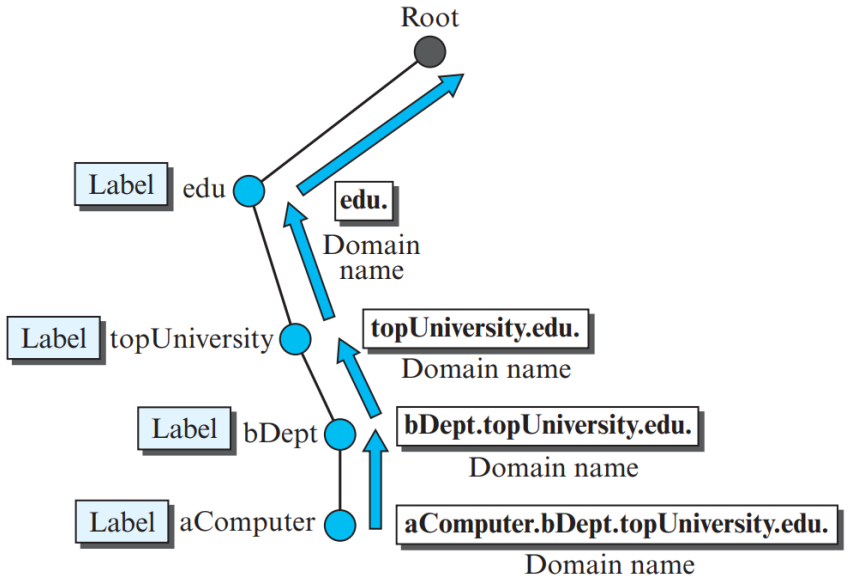
Domain Name Space

DNS uses an **inverted-tree** structure with the **root at the top**. The tree can have **128 levels** (level 0 = root to level 127).



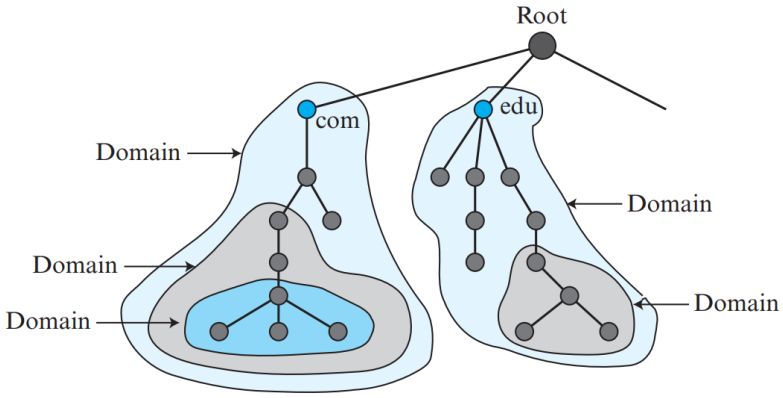
About Figure: The root is at the top, with the top-level nodes (such as edu) below it. Further levels branch down from them.

- **Label:** each node has a label of at most **63 characters**. The **root label is null** (empty). Children of a node must have different labels, which keeps names unique.
- **Domain name:** a sequence of labels separated by **dots**, **read from the node up to the root**. A full name ends with a null label, so it ends with a dot.



About Figure: The domain name aComputer.bDept.topUniversity.edu. is built from the labels aComputer, bDept, topUniversity and edu up to the root. Shorter names are the same path from higher nodes.

Term	Meaning
Fully Qualified Domain Name (FQDN)	A name ending in a null label (a dot). It reaches the root and is complete
Partially Qualified Domain Name (PQDN)	A name that does not end in a null label. It starts at a node but does not reach the root. Used when the name belongs to the client's own site; the resolver adds the missing **suffix** to make an FQDN
Domain	A **subtree** of the domain name space. Its name is the name of the node at the top of the subtree. A domain can itself be divided into domains



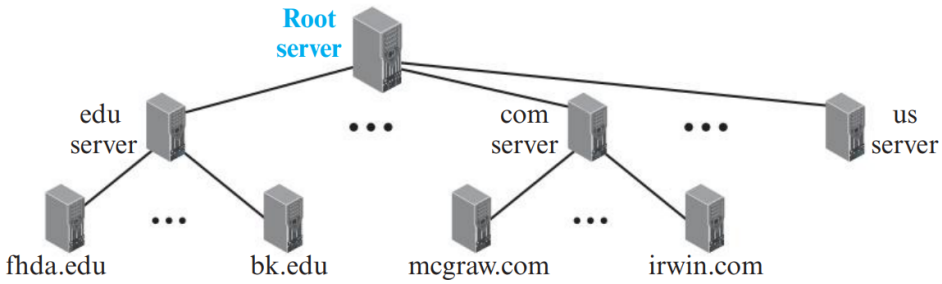
About Figure: Subtrees under the root, such as the domains under com and edu, are shown as domains. A domain may contain smaller domains.

Distribution of the Name Space

Storing all the information on one computer is **inefficient** (heavy load from worldwide requests) and **unreliable** (one failure makes the data unavailable). So it is distributed among many **DNS servers**.

Hierarchy of name servers

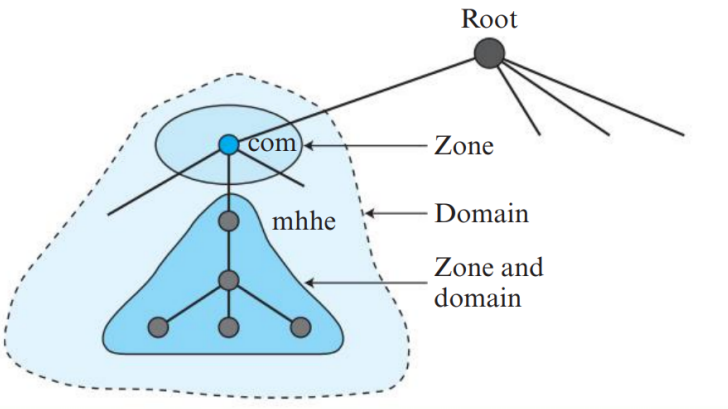
- The space is divided into domains, starting from the first-level nodes. Large domains are divided further into **subdomains**.
- Each server is **authoritative** for a large or small domain.
- This gives a hierarchy of servers matching the hierarchy of names.



About Figure: A root server is at the top. Below it are servers for edu, com and us, and below those servers for domains such as fhda.edu, bk.edu, mcgraw.com and irwin.com.

Zone

- A **zone** is what a server is responsible for, a **contiguous part** of the tree.
- If a server does not divide its domain, **domain and zone are the same**.
- If a server delegates subdomains to other servers, the zone and domain differ. The original server keeps references to the lower-level servers.
- The server keeps a **zone file** with all the information for the nodes in its zone.



About Figure: It contrasts a domain (the whole subtree) with a zone (the part the server controls itself, after delegating lower parts).

Root server

- Its zone is the **whole tree**.
- It normally stores no domain information but **delegates** to other servers and keeps references to them.
- Several root servers exist, **distributed worldwide**.

Primary and secondary servers:

Parameter	Primary server	Secondary server
Role	Stores the **zone file** for which it is authoritative	Transfers the complete zone information from another server (primary or secondary)
Creates and updates the file	**Yes**	**No**
Loads information from	Its local disk file	The primary server
Purpose	Authority	**Redundancy**: if one fails, the other continues

- Both are authoritative for their zones.
- A server can be primary for one zone and secondary for another.

DNS in the Internet

The Internet domain name space was originally divided into **generic domains, country domains and the inverse domain**. The **inverse domain** (finding a name from an IP address) is now **deprecated** (RFC 3425).

Generic domains: They define registered hosts by their **generic behaviour**. The first level has **14 labels**:

Label	Description	Label	Description
aero	Airlines and aerospace	info	Information service providers
biz	Businesses or firms	int	International organisations
com	Commercial organisations	mil	Military groups
coop	Cooperative organisations	museum	Museums
edu	Educational institutions	name	Personal names
gov	Government institutions	net	Network support centres
org	Nonprofit organisations	pro	Professional organisations

Country domains: They use **two-character country abbreviations** (for example us, fr). The second label can be organisational or a national designation. The US uses **state abbreviations**, so uci.ca.us. is the University of California, Irvine, in California, United States.

Resolution

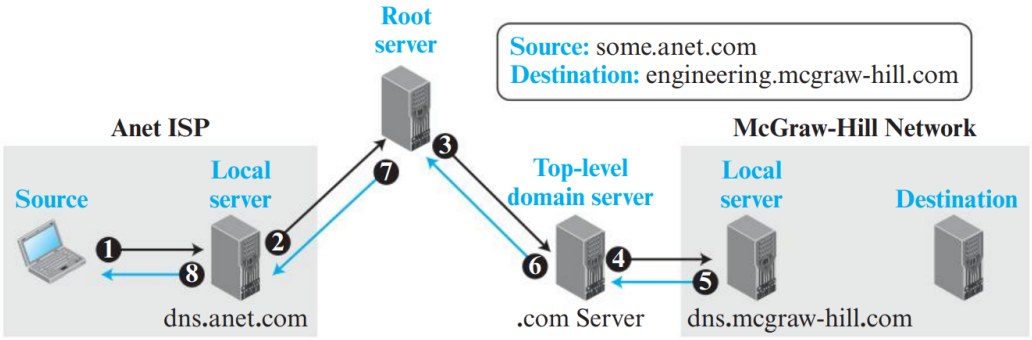
Name-address resolution is mapping a name to an address (or an address to a name).

- A host calls a DNS client called the **resolver**.
- The resolver asks the **closest DNS server**.
- If the server has the information, it answers. Otherwise it **refers** the resolver to other servers or **asks other servers** itself.
- The resolver checks the reply (real answer or error) and gives it to the requesting process.

Recursive resolution:

Example: host some.anet.com needs the IP address of engineering.mcgraw-hill.com.

1. The resolver sends the query to the **local DNS server** (dns.anet.com).
2. The local server does not know it and sends the query to a **root server**.
3. The root server knows a server for the **com** domain and sends the query to that **top-level domain server**.
4. That server knows the **local DNS server of McGraw-Hill** (dns.mcgraw-hill.com) and sends the query on.
5. dns.mcgraw-hill.com knows the IP address and **returns it to the top-level server**.
6. The address passes back to the **root server**.
7. Then back to the ISP's DNS server, which may **cache** it.
8. Finally it reaches the **source host**.

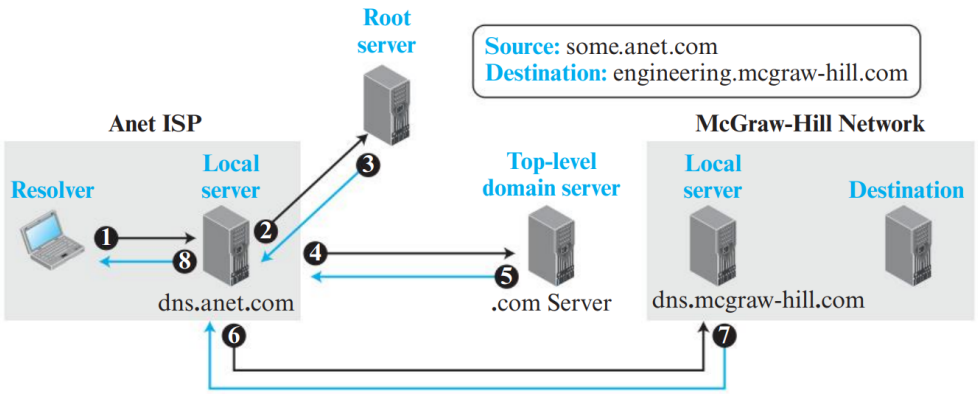


About Figure: The query travels from the source resolver through the local server, root server and top-level domain server to the McGraw-Hill local server (events 1–4). The answer returns along the same path (events 5–8).

Iterative resolution:

Each server that does not know the mapping sends **the address of the next server** back to the one that asked, instead of forwarding the query.

- Normally iterative resolution takes place **between two local servers**. The original resolver gets the final answer from its local server.
- The query in events 2, 4 and 6 is the same. Event 3 returns the **top-level domain server's address**, event 5 the **McGraw-Hill local server's address**, and event 7 the **destination's IP address**. Event 8 gives the answer to the resolver.



About Figure: The Anet local server itself asks the root server, the top-level server and the McGraw-Hill server in turn. Each replies with the next address, until the destination address is received and returned to the resolver.

Parameter	Recursive	Iterative
Who contacts the next server	The queried server forwards the query	The **original asker** contacts each server in turn
Reply from a server that does not know	None; it passes the query on	Address of the **next server**
Answer path	Returns back along the chain	Direct to the asker

Caching

- A server that receives a mapping from another server **stores it in its cache** before replying to the client.
- Later queries for the same name are answered from the cache, which **speeds up** resolution.
- Such answers are marked **unauthoritative**.

Problem: a cached mapping may become **outdated**. Two techniques solve it:

1. The authoritative server adds a **Time To Live (TTL)** (in seconds) to the mapping.
2. Each server keeps a **TTL counter** for every cached mapping and periodically **purges** expired ones.

Resource Records

A name server keeps a database of **resource records**. Each record is a **5-tuple**:

(Domain Name, Type, Class, TTL, Value)

- **Domain name:** identifies the record.
- **Type:** says how to interpret the value.
- **Class:** type of network (only **IN**, Internet, is used).
- **TTL:** number of seconds the information is valid.
- **Value:** the information kept.

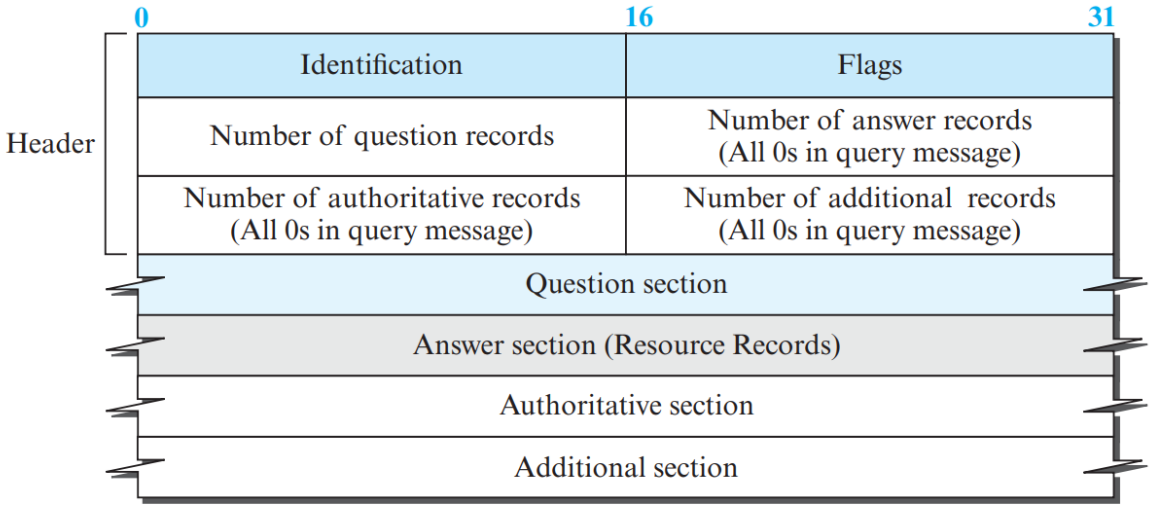
Type	Interpretation of value
A	A 32-bit IPv4 address
NS	Identifies the authoritative servers for a zone
CNAME	Alias for the official name of a host
SOA	Marks the beginning of a zone
MX	Redirects mail to a mail server
AAAA	An IPv6 address

DNS Messages

DNS uses two message types, **query** and **response**, with the **same format**.

Field or section	Purpose
Identification	Used by the client to match a response with its query
Flags	Tells whether the message is a query or response; includes error status
Number of records	Four fields giving the count of question, answer, authoritative and additional records (all 0s in a query except the question count)
Question section	One or more question records; in both query and response
Answer section	One or more resource records; **only in responses**

Field or section	Purpose
Authoritative section	Domain names of authoritative servers for the query
Additional section	Extra information that may help the resolver



Note:
The query message contains only the question section.
The response message includes the question section,
the answer section, and possibly two other sections.

About Figure: The header (identification, flags, number of question, answer, authoritative and additional records) is followed by the question, answer, authoritative and additional sections.

A **query** contains only the question section. A **response** has the question, the answer, and possibly the authoritative and additional sections.

Example: the nslookup utility (UNIX and Windows) gives the address for a given name, for example nslookup www.forouzan.biz returns the name and address 198.170.240.179.

Encapsulation

DNS can use **UDP or TCP**. The server's well-known port is **53** in both cases.

- **UDP** is used when the response is **less than 512 bytes** (the usual UDP packet limit).
- **TCP** is used when the response is **more than 512 bytes**, in two cases:
    1. The resolver **knows beforehand** that the response is large, for example a **zone transfer** by a secondary server from a primary server. It uses TCP directly.
    2. The resolver **does not know the size** and uses UDP. If the response is too big, the server **truncates** it and sets the **TC bit**. The resolver then opens a **TCP connection** and repeats the request.

Registrars

New domains are added through a **registrar**, a commercial entity **accredited by ICANN**.

- The registrar checks that the name is **unique** and enters it in the DNS database for a **fee**.
- The organisation gives the **name and IP address of its server**. Example: domain name ws.wonderful.com with IP address 200.200.200.5.

Dynamic DNS (DDNS)

In DNS, any change (adding or removing a host, changing an IP address) must be made in the **master file**, and it was done **manually**. This is not workable for today's Internet. **Dynamic Domain Name System (DDNS)** updates the master file **dynamically**.

- When a name-to-address binding is made, the information is sent, usually by **DHCP**, to the **primary DNS server**.
- The primary server **updates the zone**.
- The secondary servers are notified:
  - **Active notification:** the primary server sends a message about the change.
  - **Passive notification:** the secondary servers **periodically check** for changes.
- After notification, the secondary server requests the whole zone (**zone transfer**).
- An **authentication mechanism** can be used to prevent unauthorised changes.

Security of DNS

DNS is vital to the Internet. Three attacks are possible:

Attack	Effect	Prevention
Reading the DNS response	Reveals the sites a user visits (profile of the user)	DNS messages must be **confidential**
Modifying or faking the response	Directs the user to a site chosen by the attacker	**Message origin authentication** and **message integrity**
Flooding the server	Overwhelms or crashes it (denial of service)	Protection against denial-of-service attack

DNSSEC (DNS Security, by IETF) gives **message origin authentication and message integrity** using **digital signatures**.

- It does **not** give **confidentiality**.
- It has **no specific protection against denial of service**, although **caching** protects the upper-level servers to some extent.

Peer-to-Peer (P2P) Networks

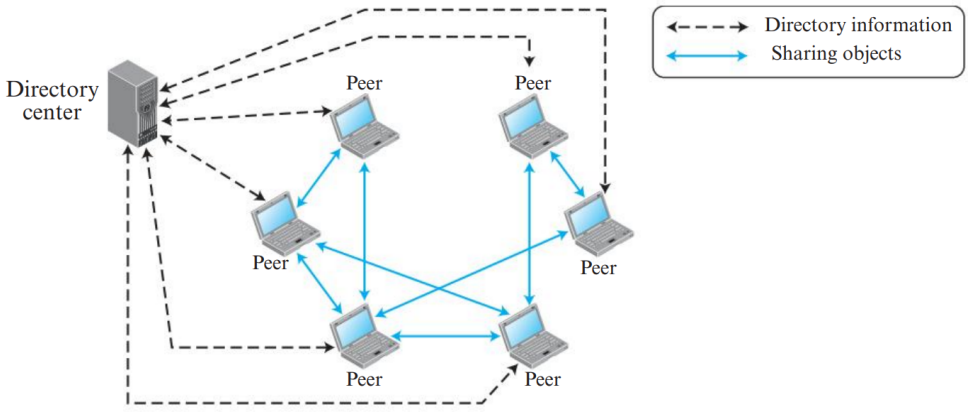
Internet users who share their resources become peers and form a network. A peer with a file (audio, video, etc.) makes it available to other peers. An interested peer connects to it and downloads the file. After downloading, that peer can also share it with others, so more copies become available.

- A peer can act as both a client and a server.
- File sharing takes place directly between peers.
- No dedicated server is required for actual file storage and transfer in a pure P2P network.

Key question: How are the list of peers and the location of files tracked?
P2P networks are broadly classified into centralized and decentralized networks.

1. Centralized Networks

In a centralized P2P network, a central server maintains a directory containing information about peers and the files they share.



About Figure: Peers register with a central directory center (directory information). Files are then shared directly between peers (sharing objects).

- The directory (list of peers and what they offer) uses the client-server approach.
- Storing and downloading files use the peer-to-peer approach.
- Therefore, it is also called a hybrid P2P network.
- Example: Napster.

Working

1. A peer registers with a central server by providing its IP address and the list of files it shares.
2. A peer searching for a file sends a query to the central server.
3. The server searches its directory and returns the IP addresses of peers having the requested file.
4. The requesting peer directly contacts one of those peers and downloads the file.
5. The directory is updated whenever peers join or leave the network.

Advantages

- Directory maintenance is simple.
- Searching for files is easy because the central server maintains the directory.

Disadvantages

- Accessing the directory creates heavy traffic and may slow down the system.
- The central server is vulnerable to attacks.
- If the central server fails, searching for files becomes unavailable.

2. Decentralized Networks

In a decentralized P2P network, there is no central directory server maintaining information about all peers and files.

- Peers communicate directly with one another.
- Each peer maintains information about some other peers.
- Peers form an overlay network, which is a logical network built on top of the physical network.
- Based on how nodes are connected, decentralized networks are classified into unstructured and structured networks.

Parameter	Centralized	Decentralized
Directory	Central server	Distributed among peers
Searching	Query sent to central server	Query routed through peers
File transfer	Directly between peers	Directly between peers
Failure	Central server is a single point of failure	No central point of failure
Traffic	Central server may become overloaded	Depends on network structure
Examples	Napster	Gnutella, Chord, Pastry, Kademlia

Now Structured and Unstructured Networks:

Unstructured Networks

In an unstructured P2P network, nodes are connected randomly without following predefined rules.

Working

1. A peer searches for a file by sending a query to one of its neighbouring peers.
2. If the neighbour knows which peer has the requested file, it returns that peer's address.
3. Otherwise, the query is forwarded to its other neighbours.
4. The query continues to spread through the network until the file is found or the search fails.

This process is called flooding.

Advantages

- Simple to construct.
- Peers can join and leave easily.
- No central directory is required.

Disadvantages

- Flooding generates heavy network traffic.
- Searching is inefficient because queries may reach many unnecessary peers.
- Poor scalability in large networks.

Examples: Gnutella and Freenet.

1. Gnutella

Gnutella is an example of an unstructured P2P network.

Working

1. A peer sends a query containing the identity of the requested file to one of its neighbours.
2. If the neighbour knows the peer having the requested file, it returns that peer's address.
3. Otherwise, the query is forwarded to all its neighbours.
4. This process continues until a peer having the requested file is identified.
5. The requesting peer directly downloads the file from that peer using a transfer protocol such as HTTP.

- Ping and pong messages are used to check whether neighbouring peers are active.
- The major disadvantage is flooding, which creates heavy traffic and limits scalability.

Structured Networks

In a structured P2P network, nodes are connected according to predefined rules.

- Each peer has a unique identifier.
- Files are identified using keys generated by a hash function.
- A Distributed Hash Table (DHT) is commonly used to maintain information about files and peers.
- Queries are routed towards the peer responsible for the requested key instead of being flooded throughout the network.

Advantages

- Efficient searching of files.
- Reduced unnecessary network traffic.
- Better scalability compared with unstructured networks.

Disadvantages

- More complex to implement.
- Additional operations are required to maintain the network when peers join or leave.

Examples: Chord, Pastry and Kademlia.

Distributed Hash Table (DHT)

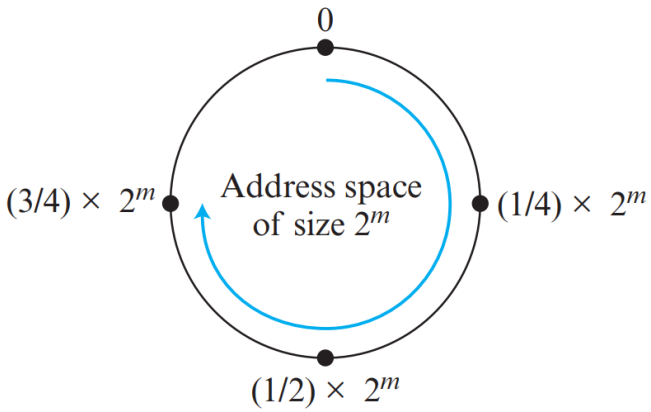
A Distributed Hash Table (DHT) is a distributed data structure used in structured P2P networks to store and locate information efficiently.

- A hash function generates identifiers for peers and files.
- The identifier of a peer is called a Node ID.
- The identifier of a file is called a Key.
- Each peer is responsible for a particular range of keys.
- Each peer maintains only partial information about the network.
- Queries are routed towards the peer responsible for the requested key.

Address Space

- Each peer and data item is mapped to a point in an address space of size (2^m).
- The possible identifiers range from 0 to ($2^m - 1$).
- The address space is commonly represented as a circular structure.
- Arithmetic is performed modulo (2^m).

Note:
1. Space range is 0 to $2^m - 1$.
2. Calculation is done modulo 2^m.



About Figure: A circular address space containing (2^m) possible points, starting from 0 and ending at ($2^m - 1$). Each peer and data item is assigned an identifier within this address space.

Hashing

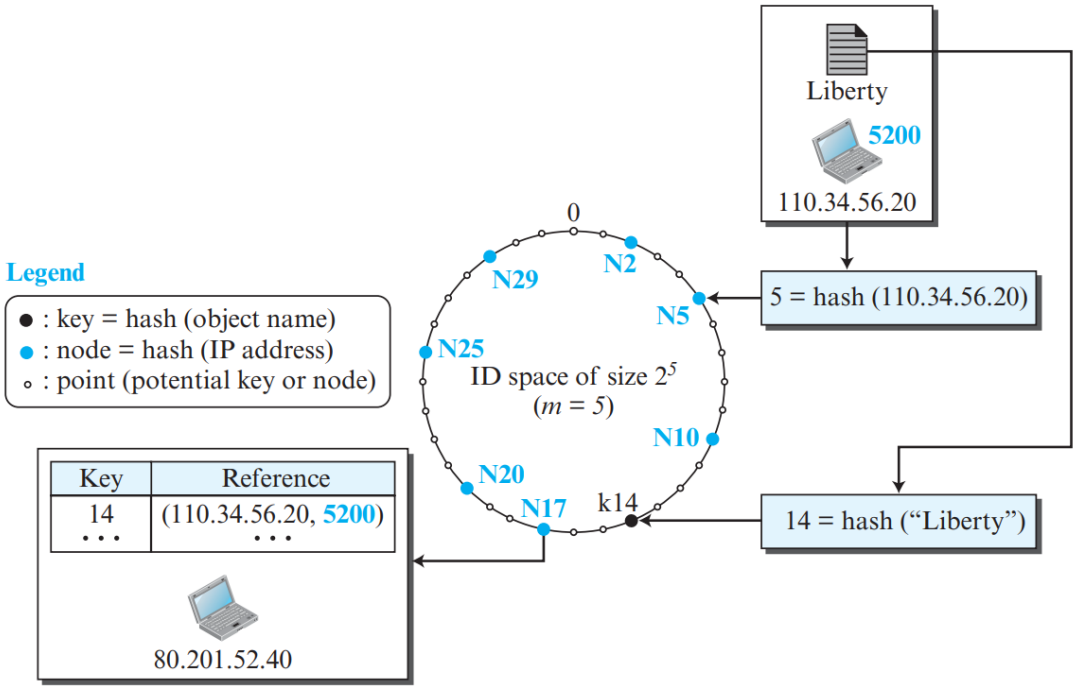
- Node ID = Hash of the peer identifier (normally its IP address).
- Key = Hash of the object or file name.
- A hash function maps the input into an identifier in the address space.
- The key is used to determine which peer is responsible for the requested file.

Storing the Object

A DHT can store objects in two ways:

1. **Direct storage:** The actual object is stored in the node whose ID is closest to the key.
2. **Indirect storage:** The actual object remains with its original owner, while a reference to the object is stored in the node whose ID is closest to the key.

Most DHT systems use indirect storage to improve efficiency.



About Figure: A file named "Liberty" is assigned the key k14 using hashing. Node N17 is the closest node to this key, so a reference containing the file's IP address and port number is stored at N17, while the actual file remains with its original peer N5.

Working of DHT

1. A peer searches for a file by providing its name.
2. A hash function generates the key corresponding to that file.
3. The DHT determines the peer responsible for the generated key.
4. The query is routed through the network towards that peer.
5. The responsible peer returns the location or reference of the requested file.
6. The requesting peer contacts the actual owner and downloads the file.

Advantages of DHT

- Efficient searching without flooding.
- No central directory is required.
- Supports large-scale P2P networks.
- Information is distributed among multiple peers.

Examples of DHT Protocols:

1. Chord
- Uses a circular identifier space.
- Uses finger tables to route queries efficiently.
- A file is stored at the successor node of its key.

2. Pastry
- Uses prefix-based routing.
- Routes queries based on the common prefix between identifiers.
- Stores a key at the numerically closest node.

3. Kademlia
- Uses XOR distance to determine the closeness between identifiers.
- Maintains routing information using k-buckets.
- Used in trackerless BitTorrent.

Parameter	Chord	Pastry	Kademlia
Routing method	Circular identifier space	Prefix-based routing	XOR distance
Data structure	Finger table	Routing table and leaf set	Routing table with k-buckets
Key location	Successor of key	Numerically closest node	Closest node using XOR distance
Application	CFS	PAST	BitTorrent

Case Study: BitTorrent

BitTorrent, designed by Bram Cohen, is a P2P protocol used for sharing large files among a group of peers.

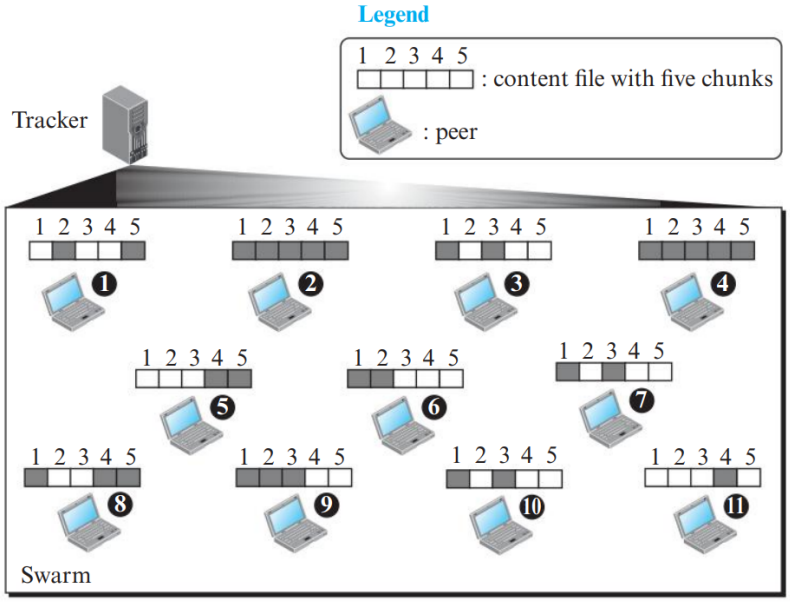
- Instead of one peer sending the entire file to another peer, multiple peers collaborate to distribute the file.
- The file is divided into smaller parts called pieces or chunks.
- Each peer downloads pieces from other peers and simultaneously uploads available pieces to them.
- This collaborative file-sharing process is called a torrent.

Important Terminologies

Term	Meaning
Torrent	Collaborative file-sharing process
Swarm	Set of all peers participating in a torrent
Seed	Peer having the complete content file
Leech	Peer having only part of the file and downloading the remaining pieces
Tracker	Entity that tracks the operation of the swarm
Content file	Actual file being shared, divided into pieces
Torrent file	Metadata file containing information about the pieces and the tracker

BitTorrent with a Tracker

A tracker is an entity that maintains information about peers participating in a torrent and helps new peers discover other peers.



Note: Peers 2 and 4 are seeds; others are leeches.

About Figure: A swarm of 11 peers and a tracker. The content file is divided into five pieces. Peers 2 and 4 are seeds having all five pieces, while the remaining peers are leeches having only some pieces.

Joining a Torrent – Working

1. A new peer obtains the torrent file containing information about the content file and the tracker's address.
2. The peer contacts the tracker.
3. The tracker returns the addresses of some peers participating in the same torrent.
4. The new peer connects to those peers and downloads available pieces.
5. At the same time, it uploads the pieces it already possesses to other peers.
6. After obtaining all pieces, the peer becomes a seed and can continue sharing the complete file.

- A peer can leave before downloading all pieces and may rejoin later.
- Multiple peers collaborate to distribute the file, reducing dependence on a single source.

Policies for Fairness and Efficiency

BitTorrent uses different policies to improve file-sharing performance and ensure efficient distribution of pieces.

1. Choking and Unchoking

- Each peer limits the number of simultaneous connections to avoid overloading.
- Unchoked peers are allowed to exchange data.
- Choked peers are temporarily prevented from receiving data from that peer.
- Interested and uninterested states indicate whether a peer wants pieces available from another peer.

2. Tit-for-Tat Policy

- Peers exchange data with other peers that provide good upload rates.
- A peer gives priority to neighbours that also contribute by uploading data.
- This encourages fair sharing of resources.

3. Optimistic Unchoking

- Every 30 seconds, a peer randomly selects one interested peer from the choked group.
- This peer is temporarily unchoked regardless of its upload rate.
- It allows newly joined peers to receive pieces even when they have nothing to share.

4. Rarest-First Policy

- A peer first downloads the pieces that have the fewest available copies among its neighbours.
- This ensures that rare pieces are distributed throughout the network.
- It helps maintain a balanced availability of file pieces.

Trackerless BitTorrent

In trackerless BitTorrent, a central tracker is not required to maintain information about peers.

- It uses a Distributed Hash Table (DHT), based on Kademlia.
- Information about peers is distributed among nodes in the P2P network.
- The hash of the torrent metadata file is used as the key.
- The value contains information about the list of peers participating in the swarm.

Working

1. A new peer generates the key by hashing the torrent metadata file.
2. It sends the key to a node it already knows.
3. Kademlia DHT identifies the node responsible for that key.
4. The responsible node returns the list of peers participating in that torrent.
5. The new peer connects to those peers and exchanges file pieces using the BitTorrent protocol.

Advantages

- Does not depend on a central tracker.
- Peer information is distributed among multiple nodes.
- New peers can discover other peers even without a dedicated tracker.

Parameter	Tracker-Based	Trackerless
Tracker	Required	Not required
Peer discovery	Through tracker	Through DHT
Architecture	Centralized peer discovery	Distributed peer discovery
Technology	Tracker protocol	Kademlia DHT
Dependency	Depends on tracker availability	No central tracker dependency

MODEL QUESTIONS

1. An application on a host generates 1460 bytes of data. The transport layer uses UDP with an 8-byte header, the network layer adds a 20-byte IP header, and the data link layer adds an Ethernet header and trailer totalling 18 bytes. Calculate the size of the frame transmitted over the network and the percentage of the frame that is actual application data (overhead efficiency). (3 Marks)
2. Differentiate between the client-server paradigm and the peer-to-peer (P2P) paradigm. Give one example of each. (3 Marks)
3. Explain the layered architecture of the TCP/IP protocol suite. State the function of each layer and the name of the data unit handled at each layer. (5 Marks)
4. Trace the journey of a message from an application on host A to the application on host B, showing encapsulation at the sender and decapsulation at the receiver. Draw a neat diagram showing the logical connection between peer layers. (4 Marks)
5. With a neat diagram, explain the architecture of electronic mail. Describe the roles of the user agent, message transfer agent and message access agent. Explain the use of SMTP, POP3 and IMAP in the delivery of a mail message. (6 Marks)
6. Describe roles of the control and data connections in FTP file transfer. (3 Marks)

© NOTES BY CLEARITT LEARNING
SUBSCRIBE ON YOUTUBE | DONATE TO US