MODULE 1: APPLICATION LAYER

MODULE 1: APPLICATION LAYER

1.1 Principles of Network Applications

- Network-applications are the driving forces for the explosive development of the internet.
- Examples of network-applications:
  1) Web
  2) File transfers
  3) E-mail
  4) P2P file sharing
  5) Social networking (Facebook, Twitter)
  6) Video distribution (YouTube)
  7) Real-time video conferencing (Skype)
  8) On-line games (World of Warcraft)
- In network-applications, program usually needs to
  → run on the different end-systems and
  → communicate with one another over the network.
- For ex: In the Web application, there are 2 different programs:
  1) The browser program running in the user's host (Laptop or Smartphone).
  2) The Web-server program running in the Web-server host.

1.1.1 Network Application Architectures

- Two approaches for developing an application:
  1) Client-Server architecture
  2) P2P (Peer to Peer) architecture

1.1.1.1 Client-Server Architecture

- In this architecture, there is a server and many clients distributed over the network (Figure 1.1a).
- The server is always-on while a client can be randomly run.
- The server is listening on the network and a client initializes the communication.
- Upon the requests from a client, the server provides certain services to the client.
- Usually, there is no communication between two clients.
- The server has a fixed IP address.
- A client contacts the server by sending a packet to the server's IP address.
- A server is able to communicate with many clients.
- The applications such as FTP, telnet, Web, e-mail etc use the client-server architecture.

1.1.1.1.1 Data Center

- Earlier, client-server architecture had a single-server host.
- But now, a single-server host is unable to keep up with all the requests from large no. of clients.
- For this reason, data-center a is used.
- A data-center contains a large number of hosts.
- A data-center is used to create a powerful virtual server.
- In date center, hundreds of servers must be powered and maintained.
- For example:
  - Google has around 50 data-centers distributed around the world.
  - These 50 data-centers handle search, YouTube, Gmail, and other services.

1.1.1.2 P2P Architecture

- There is no dedicated server (Figure 1.1b).
- Pairs of hosts are called peers.
- The peers communicate directly with each other.
- The peers are not owned by the service-provider. Rather, the peers are laptops controlled by users.
- Many of today's most popular and traffic-intensive applications are based on P2P architecture.
- Examples include file sharing (BitTorrent), Internet telephone (Skype) etc.
- Main feature of P2P architectures: self-scalability.
- For ex: In a P2P file-sharing system,
  - Each peer generates workload by requesting files.
  - Each peer also adds service-capacity to the system by distributing files to other peers.
- Advantage: Cost effective „.' Normally, server-infrastructure & server bandwidth are not required.
- Three challenges of the P2P applications:
  - Most residential ISPs have been designed for asymmetrical bandwidth usage.
  - Asymmetrical bandwidth means there is more downstream-traffic than upstream-traffic.
  - But P2P applications shift upstream-traffic from servers to residential I Ps, which stress on the ISPs.

2) Security

- Since the highly distribution and openness, P2P applications can be a challenge to security.

3) Incentive

- Success of P2P depends on convincing users to volunteer bandwidth & resources to the applications.

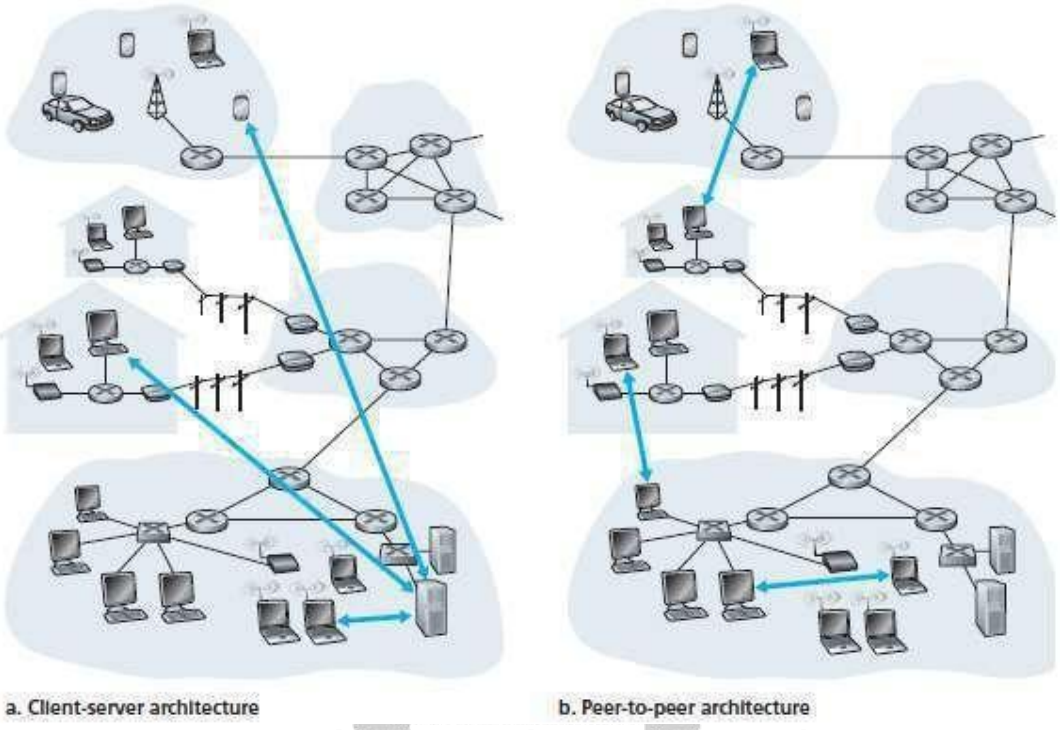


Figure 1.1: (a) Client-server architecture; (b) P2P architecture

1.1.2 Processes Communicating

1.1.2.1 Process

- A process is an instance of a program running in a computer.(IPC → inter-processcommunication).
- The processes may run on the 1) same system or 2) different systems.
  1) The processes running on the same end-system can communicate with each other using IPC.
  2) The processes running on the different end-systems can communicate by exchanging messages.
     i) A sending-process creates and sends messages into the network.
     ii) A receiving-process receives the messages and responds by sending messages back.

1.1.2.1.1 Client & Server Processes

- A network-application consists of pairs of processes:
  1) The process that initiates the communication is labeled as the client.
  2) The process that waits to be contacted to begin the session is labeled as the server.
- For example:
  1) In Web application, a client-browser process communicates with a Web-server-process.
  2) In P2P file system, a file is transferred from a process in one peer to a process in another peer.

1.1.2.1.2 Interface between the Process and the Computer Network Socket

- Any message sent from one process to another must go through the underlying-network.
- A process sends/receives message through a software-interface of underlying-network called socket.
- Socket is an API between the application-layer and the transport layer within a host (Figure 1.2).
- The application-developer has complete control at the application-layer side of thesocket.
- But, the application-developer has little control of the transport-layer side of thesocket.

  1) The choice of transport-protocol: TCP or UDP. (API → Application Programming Interface)
  2) The ability to fix parameters such as maximum-buffer & maximum-segment-sizes.

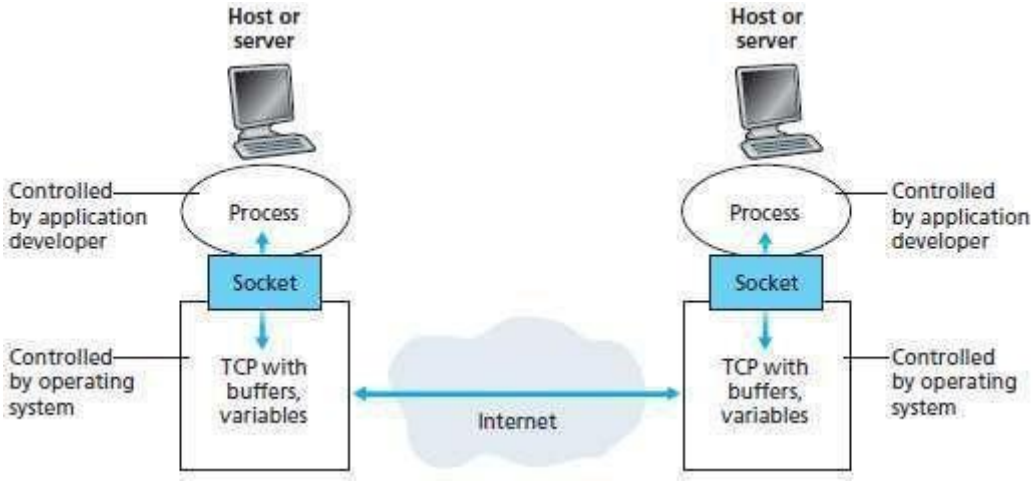


Figure 1.2: Application processes, sockets, and transport-protocol

1.1.2.1.3 Addressing Processes

- To identify the receiving-process, two pieces of information need to be specified:
  1) IP address of the destination-host.
  2) Port-number that specifies the receiving-process in the destination-host.
- In the Internet, the host is identified by IP address.
- An IP address is a 32-bit that uniquely identify the host.
- Sending-process needs to identify receiving-process ‚Äì' a host may run several network-applications.
- For this purpose, a destination port-number is used.
- For example,

  A Web-server is identified by port-number 80. A mail-server is identified by port-number 25.

1.1.3 Transport Services Available to Applications

- Networks usually provide more than one transport-layer protocols for different applications.
- An application-developer should choose certain protocol according to the type of applications.
- Different protocols may provide different services.

1.1.3.1 Reliable Data Transfer

- Reliable means guaranteeing the data from the sender to the receiver is delivered correctly. For ex: TCP provides reliable service to an application.
- Unreliable means the data from the sender to the receiver may never arrive.
- Unreliability may be acceptable for loss-tolerant applications, such as multimedia applications.
- In multimedia applications, the lost data might result in a small glitch in the audio/video.

1.1.3.2 Throughput

- Throughput is the rate at which the sending-process can deliver bits to the receiving-process.
- Since other hosts are using the network, the throughput can fluctuate with time.
- Two types of applications:

1) Bandwidth Sensitive Applications

- These applications need a guaranteed throughput. For ex: Multimedia applications
- Some transport-protocol provides guaranteed throughput at some specified rate (r bits/sec).

2) Elastic Applications

- These applications may not need a guaranteed throughput. For ex: Electronic mail, File transfer & Web transfers.

1.1.3.3 Timing

- A transport-layer protocol can provide timing-guarantees.
- For ex: guaranteeing every bit arrives at the receiver in less than 100 msec.
- Timing constraints are useful for real-time applications such as

  → Internet telephony

→ Virtual environments
→ Teleconferencing and
→ Multiplayer games

1.1.3.4 Security

- A transport-protocol can provide one or more security services.
- For example,
  1) In the sending host, a transport-protocol can encrypt all the transmitted-data.
  2) In the receiving host, the transport-protocol can decrypt the received-data.

1.1.4 Transport Services Provided by the Internet

- The Internet makes two transport-protocols available to applications, UDP and TCP.
- An application-developer who creates a new network-application must use either: UDP or TCP.
- Both UDP & TCP offers a different set of services to the invoking applications.
- Table 1.1 shows the service requirements for some selected applications.

Table 1.1: Requirements of selected network-applications

Application	Data Loss	Throughput Time	Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet-telephony/ Video-conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of ms
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of ms
Instant messaging	No loss	Elastic	Yes and no

1.1.4.1 TCP Services

- An application using transport-protocol TCP, receives following 2 services.

  - Before the start of communication, client & server need to exchange control-information.
  - This phase is called handshaking phase.
  - Then, the two processes can send messages to each other over the connection.
  - After the end of communication, the applications must tear down the connection.

2) Reliable Data Transfer Service

  - The communicating processes must deliver all data sent without error & in the proper order.

- TCP also includes a congestion-control.
- The congestion-control throttles a sending-process when the network is congested.

1.1.4.2 UDP Services

- UDP is a lightweight transport-protocol, providing minimal services.
- UDP is connectionless, so there is no handshaking before the 2 processes start to communicate.
- UDP provides an unreliable data transfer service.
- Unreliable means providing no guarantee that the message will reach the receiving-process.
- Furthermore, messages that do arrive at the receiving-process may arrive out-of-order.
- UDP does not include a congestion-control.
- UDP can pump data into the network-layer at any rate.

1.2 The Web & HTTP

- The appearance of Web dramatically changed the Internet.
- Web has many advantages for a lot of applications.
- It operates on demand so that the users receive what they want when they want it.
- It provides an easy way for everyone make information available over the world.
- Hyperlinks and search engines help us navigate through an ocean of Web-sites.
- Forms, JavaScript, Java applets, and many other devices enable us to interact with pages and sites.
- The Web serves as a platform for many killer applications including YouTube, Gmail, and Facebook.

1.2.1 Overview of HTTP

1.2.1.1 Web

- A web-page consists of objects (HTML → Hyper Text Markup Language).
- An object is a file such as an HTML file, a JPEG image, a Java applet, a video chip.
- The object is addressable by a single URL (URL → Uniform Resource Locator).
- Most Web-pages consist of a base HTML file & several referenced objects.
- For example:

  If a Web-page contains HTML text and five JPEG images; then the Web-page has six objects:
  1) Base HTML file and
  2) Five images.
- The base HTML file references the other objects in the page with the object's URLs.
- URL has 2 components:
  1) The hostname of the server that houses the object and
  2) The object's path name.
- For example:

  –http://www.someSchool.edu/someDepartment/picture.gif| In above URL,
  1) Hostname = –www.someSchool.edu |
  2) Path name = —/someDepartment/picture.gif|.
- The web browsers implement the client-side of HTTP. For ex: Google Chrome, Internet Explorer
- The web-servers implement the server-side of HTTP. For ex: Apache

1.2.1.2 HTTP

- HTTP is Web's application-layer protocol (Figure 1.3) (HTTP → HyperText Transfer Protocol).
- HTTP defines

  → how clients request Web-pages from servers and
  → how servers transfer Web-pages to clients.

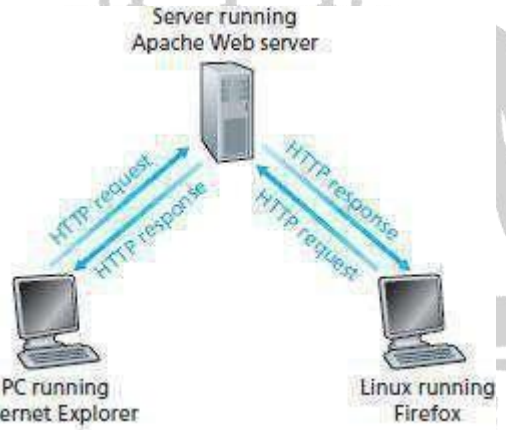


Figure 1.3: HTTP request-response behavior

- When a user requests a Web-page, the browser sends HTTP request to the server.
- Then, the server responds with HTTP response that contains the requested-objects.
- HTTP uses TCP as its underlying transport-protocol.
- The HTTP client first initiates a TCP connection with the server.
- After connection setup, the browser and the server-processes access TCP through their sockets.
- HTTP is a stateless protocol.
- Stateless means the server sends requested-object to client w/o storing state-info about the client.
- HTTP uses the client-server architecture:

  Browser that requests receive and displays Web objects.

  2) Server

  Web-server sends objects in response to requests.

1.2.2 Non-Persistent & Persistent Connections

- In many internet applications, the client and server communicate for an extended period of time.
- When this client-server interaction takes place over TCP, a decision should be made:
  1) Should each request/response pair be sent over a separate TCP connection or
  2) Should all requests and their corresponding responses be sent over same TCP connection?
- These different connections are called non-persistent connections (1) or persistent connections (2).
- Default mode: HTTP uses persistent connections.

1.2.2.1 HTTP with Non-Persistent Connections

- A non-persistent connection is closed after the server sends the requested-object to the client.
- In other words, the connection is used exactly for one request and one response.
- For downloading multiple objects, multiple connections must be used.
- Suppose user enters URL:
  "http://www.someSchool.edu/someDepartment/home.index"
- Assume above link contains text and references to 10 jpeg images.

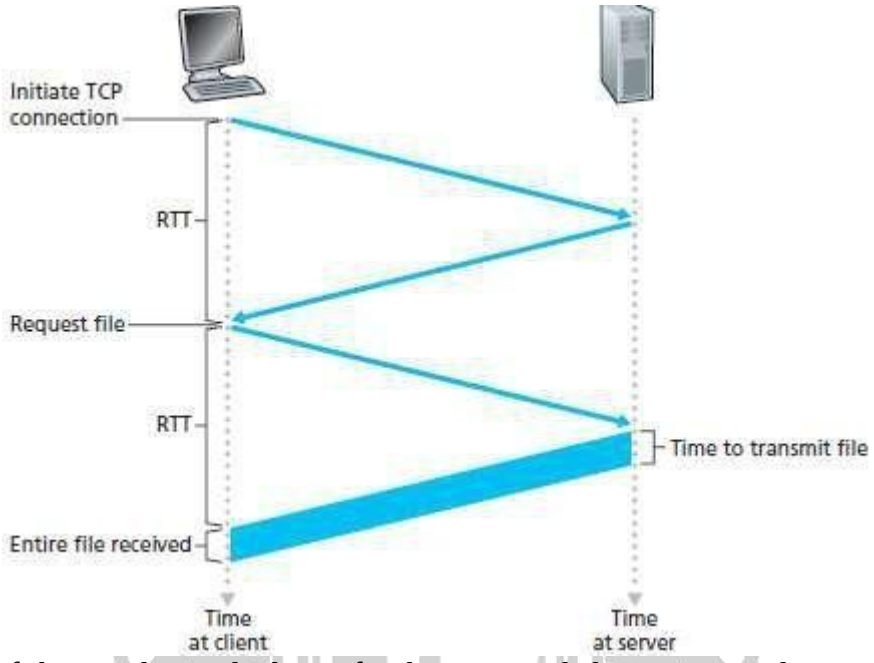
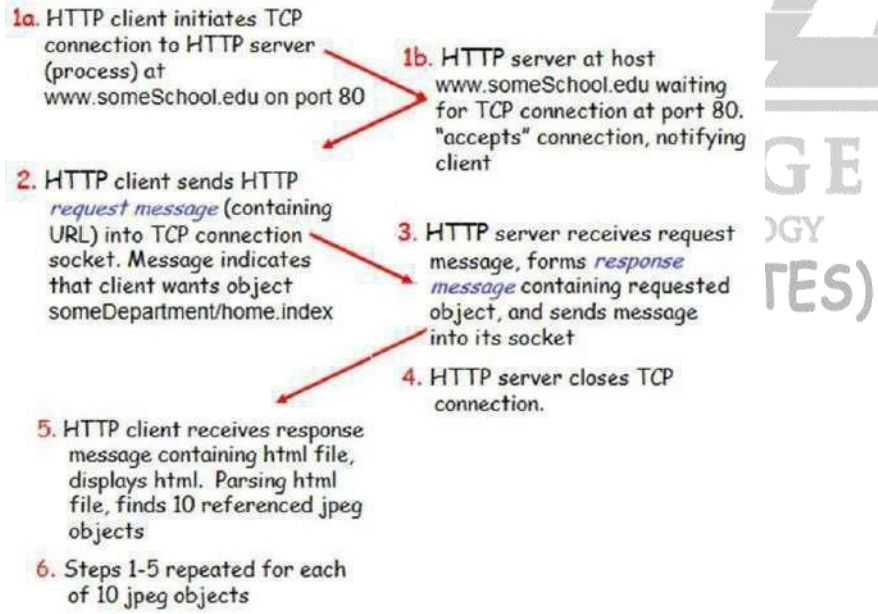


Figure 1.4: Back-of-the-envelope calculation for the time needed to request and receive an HTML file

- Here is how it works:



- RTT is the time taken for a packet to travel from client to server and then back to the client.
- The total response time is sum of following (Figure 1.4):
  i) One RTT to initiate TCP connection (RTT → Round Trip Time).
  ii) One RTT for HTTP request and first few bytes of HTTP response to return.
  iii) File transmission time.

i.e. Total response time = (i) + (ii) + (iii) = 1 RTT+ 1 RTT+ File transmission time
= 2(RTT) + File transmission time

1.2.2.2 HTTP with Persistent Connections

- Problem with Non-Persistent Connections:

1) A new connection must be established and maintained for each requested-object.
   - Hence, buffers must be allocated and state info must be kept in both the client and server.
   - This results in a significant burden on the server.
2) Each object suffers a delivery delay of two RTTs:
   i) One RTT to establish the TCP connection and
   ii) One RTT to request and receive an object.

- Solution: Use persistent connections.
- With persistent connections, the server leaves the TCP connection open after sending responses.
- Hence, subsequent requests & responses b/w same client & server can be sent over same connection
- The server closes the connection only when the connection is not used for a certain amount of time.
- Default mode of HTTP: Persistent connections with pipelining.
- Advantages:
  1) This method requires only one RTT for all the referenced-objects.
  2) The performance is improved by 20%.

1.2.3 HTTP Message Format

- Two types of HTTP messages: 1) Request-message and 2) Response-message.

1.2.3.1 HTTP Request Message

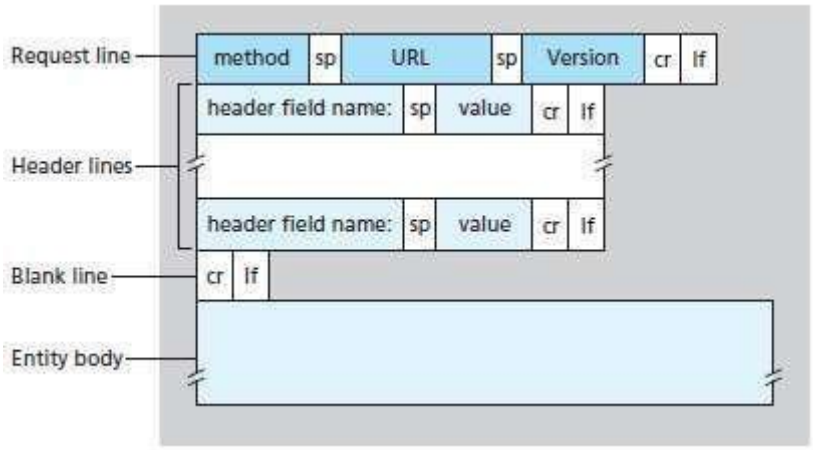


Figure 1.5: General format of an HTTP request-message

- An example of request-message is as follows: GET

```
/somedir/page.html HTTP/1.1 Host:
www.someschool.edu Connection:
close
User-agent:
Mozilla/5.0 Accept-
language: eng
```

- The request-message contains 3 sections (Figure 1.5):
  1) Request-line
  2) Header-line and
  3) Carriage return.
- The first line of message is called the request-line. he subsequent lines are called the header-lines.
- The request-line contains 3 fields. The meaning of the fields is as follows:
  1) Method
  - —GET‖: This method is used when the browser requests an object from the server.
  2) URL
  - —/somedir/page.html‖: This is the object requested by the browser.
  - —HTTP/1.1‖: This is version used by the browser.
- The request-message contains 4 header-lines. The meaning of the header-lines is as follows:
  1) –Host: www.someschool.edu‖ specifies the host on which the object resides.
  2) –Connection: close‖ means requesting a non-persistent connection.
  3) –User-agent:Mozilla/5.0‖ means the browser used is the Firefox.
  4) –Accept-language:eng‖ means English is the preferred language.

- The method field can take following values: GET, POST, HEAD, PUT and DELETE.
  1) GET is used when the browser requests an object from the server.
  2) POST is used when the user fills out a form & sends to the server.
  3) HEAD is identical to GET except the server must not return a message-body in the response.
  4) PUT is used to upload objects to servers.
  5) DELETE allows an application to delete an object on a server.

1.2.3.2 HTTP Response Message

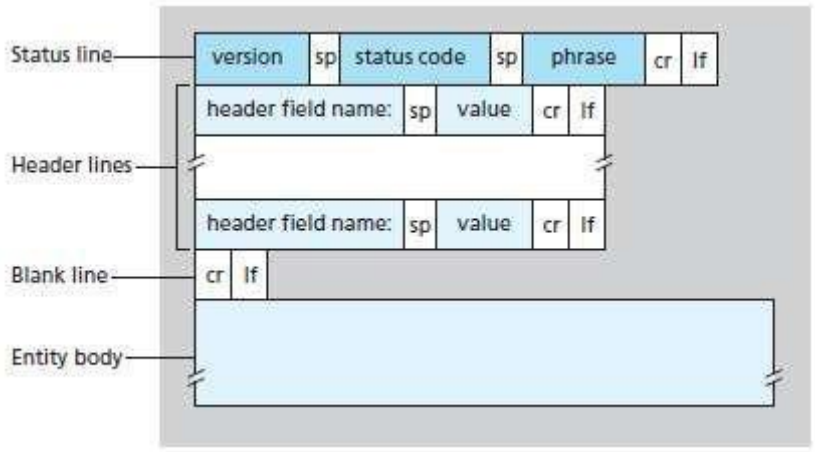


Figure 1.6: General format of an HTTP response-message

- An example of response-message is as follows:

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821 Content-
Type: text/html (data data data
data data ...)
```

- The response-message contains 3 sections (Figure 1.6):
  1) Status line
  2) Header-lines and
  3) Data (Entity body).
- The status line contains 3 fields:
  1) Protocol version
  2) Status-code and
  3) Status message.
- Some common status-codes and associated messages include:
  1) 200 OK: Standard response for successful HTTP requests.
  2) 400 Bad Request: The server cannot process the request due to a client error.
  3) 404 Not Found: The requested resource cannot be found.
- The meaning of the Status line is as follows:
  —HTTP/1.1 200 OK‖: This line indicates the server is using HTTP/1.1 & that everything is OK.
- The response-message contains 6 header-lines. The meaning of the header-lines is as follows:
  1) Connection: This line indicates browser requesting a non-persistent connection.
  2) Date: This line indicates the time & date when the response was sent by the server.
  3) Server: his line indicates that the message was generated by an Apache Web-server.
  4) Last-Modified: This line indicates the time & date when the object was last modified.
  5) Content-Length: This line indicates the number of bytes in the sent-object.
  6) Content- ype: This line indicates that the object in the entity body is HTML text.

1.2.4 User-Server Interaction: Cookies

- Cookies refer to a small text file created by a Web-site that is stored in the user's computer.
- Cookies are stored either temporarily for that session only or permanently on the hard disk.
- Cookies allow Web-sites to keep track of users.

- Cookie technology has four components:
  1) A cookie header-line in the HTTP response-message.
  2) A cookie header-line in the HTTP request-message.
  3) A cookie file kept on the user's end-system and managed by the user's browser.
  4) A back-end database at the Web-site.

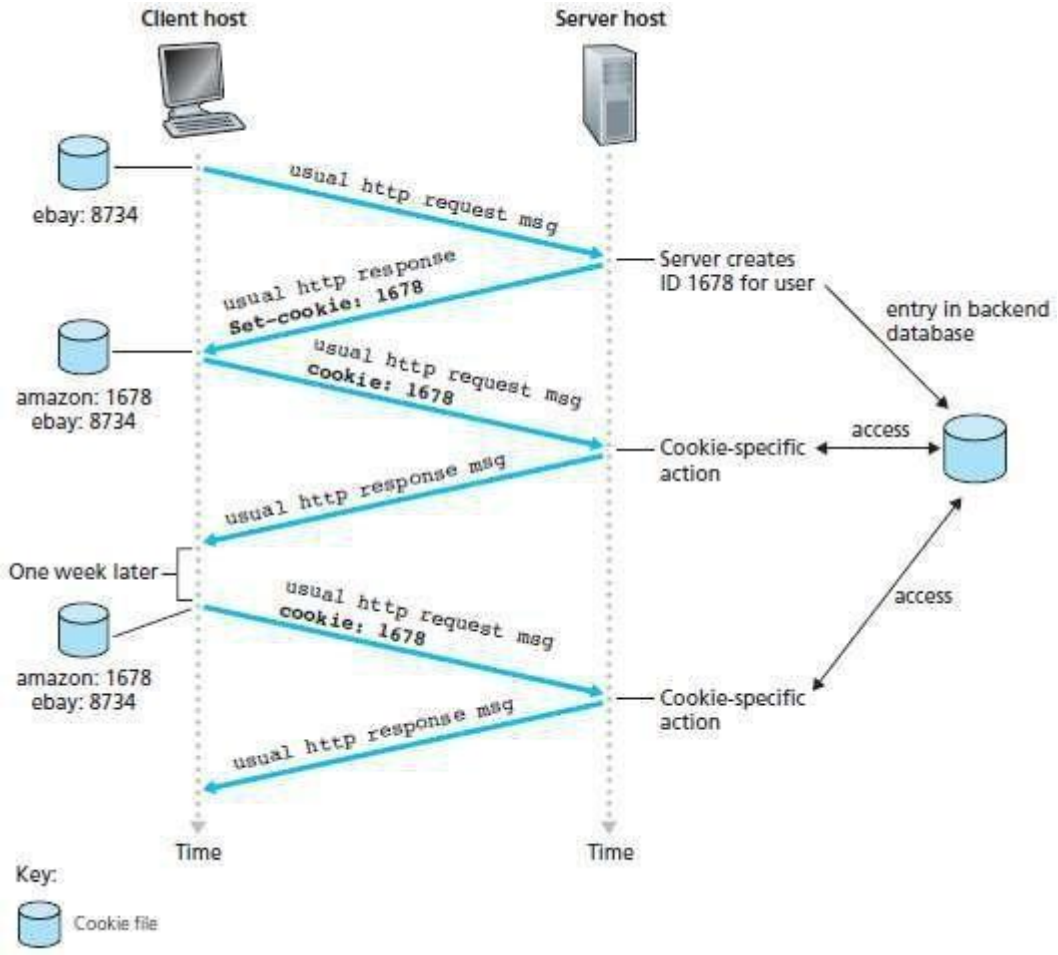


Figure 1.7: Keeping user state with cookies

- Here is how it works (Figure 1.7):
  1) When a user first time visits a site, the server
     → creates a unique identification number (1678) and
     → creates an entry in its back-end database by the identification number.
  2) he server then responds to user's browser.
     H P response includes Set-cookie: header which contains the identification number (1678)
  3) he browser then stores the identification number into the cookie-file.
  4) Each time the user requests a Web-page, the browser
     → extracts the identification number from the cookie file, and
     → puts the identification number in the HTTP request.
  5) In this manner, the server is able to track user's activity at the web-site.

1.2.5 Web Caching

- A Web-cache is a network entity that satisfies HTTP requests on the behalf of an original Web-server.
- The Web-cache has disk-storage.
- The disk-storage contains copies of recently requested-objects.

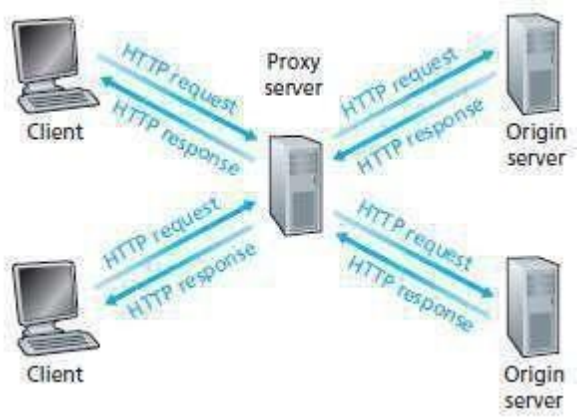


Figure 1.8: Clients requesting objects through a Web-cache (or Proxy erver)

- Here is how it works (Figure 1.8):
  1) The user's HTTP requests are first directed to the web-cache.
  2) If the cache has the object requested, the cache returns the requested-object to the client.
  3) If the cache does not have the requested-object, then the cache
     → connects to the original server and
     → asks for the object.
  4) When the cache receives the object, the cache
     → stores a copy of the object in local-storage and
     → sends a copy of the object to the client.
- A cache acts as both a server and a client at the same time.
  1) The cache acts as a server when the cache
     → receives requests from a browser and
     → sends responses to the browser.
  2) The cache acts as a client when the cache
     → requests to an original server and
     → receives responses from the origin server.
- Advantages of caching:
  1) To reduce response-time for client-request.
  2) To reduce traffic on an institution‘s access-link to the Internet.
  3) To reduce Web-traffic in the Internet.

1.2.6 The Conditional GET

- Conditional GET refers a mechanism that allows a cache to verify that the objects are up to date.
- An HTTP request-message is called conditional GET if
  1) Request-message uses the GET method and
  2) Request-message includes an If-Modified-Since: header-line.
- The following is an example of using conditional GET: GET

```
/fruit/kiwi.fig HTTP1.1
Host: www.exoriguecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

- The response is:

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
```

1.3 File Transfer: FTP

- FTP is used by the local host to transfer files to or from a remote-host over the network.
- FTP uses client-server architecture (Figure 1.9).
- FTP uses 2 parallel TCP connections (Figure 1.10):

1) Control Connection

- The control-connection is used for sending control-information b/w local and remote-hosts.
- The control-information includes:
  → user identification
  → password

→ commands to change directory and
→ commands to put & get files.

2) Data Connection

- The data-connection is used to transfer files.

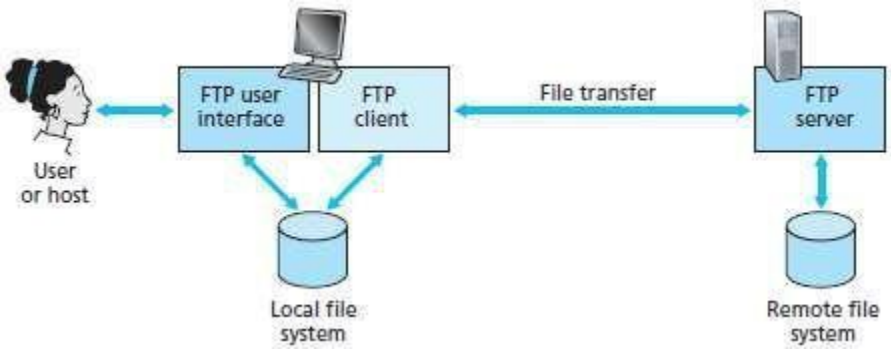


Figure 1.9: FTP moves files between local and remote file systems

Figure 1.10: Control and data-connections

- Here is how it works:
  1) When session starts, the client initiates a control-connection with the server on port 21.
  2) The client sends user-identity and password over the control-connection.
  3) Then, the server initiates data-connection to the client on port 20.
  4) FTP sends exactly one file over the data-connection and then closes the data-connection.
  5) Usually, the control-connection remains open throughout the duration of the user-session.
  6) But, a new data-connection is created for each file transferred within a session.
- During a session, the server must maintain the state-information about the user.
- For example:
  The server must keep track of the user's current directory.
- Disadvantage:
  Keeping track of state-info limits the no. of sessions maintained simultaneously by a server.

1.3.1 FTP Commands & Replies

- The commands are sent from client to server.
- The replies are sent from server to client.
- The commands and replies are sent across the control-connection in 7-bit ASCII format.
- Each command consists of 4-uppercase ASCII characters followed by optional arguments.
- For example:
  1) USER username
  - Used to send the user identification to the server. 2) PASS password
  - Used to send the user password to the server.
  3) LIST
  - Used to ask the server to send back a list of all the files in the current remote directory. 4) RETR filename
  - Used to retrieve a file from the current directory of the remote-host.
  5) STOR filename
  - Used to store a file into the current directory of the remote-host.
- Each reply consists of 3-digit numbers followed by optional message.
- For example:
  1) 331 Username OK, password required
  2) 125 Data-connection already open; transfer starting
  3) 425 Can‘t open data-connection
  4) 452 Error writing file

1.4 Electronic Mail in the Internet

- e-mail is an asynchronous communication medium in which people send and read messages.
- e-mail is fast, easy to distribute, and inexpensive.
- e-mail has features such as
  → messages with attachments
  → hyperlinks
  → HTML-formatted text and
  → embedded photos.
- Three major components of an e-mail system (Figure 1.11):

  - User-agents allow users to read, reply to, forward, save and compose messages.
  - For example: Microsoft Outlook and Apple Mail

  - Mail-servers contain mailboxes for users.
  - A message is first sent to the sender's mail-server.
  - Then, the sender‘s mail-server sends the message to the receiver's mail-server.
  - If the sender‘s server cannot deliver mail to receiver‘s server, the sender‘s server
    → holds the message in a message queue and
    → attempts to transfer the message later.

3) SMTP (Simple Mail Transfer Protocol)

- SMTP is an application-layer protocol used for email.
- SMTP uses TCP to transfer mail from the sender‘s mail-server to the recipient‘s mail-server.
- SMTP has two sides:
  1) A client-side, which executes on the sender‘s mail-server.
  2) A server-side, which executes on the recipient‘s mail-server.
- Both the client and server-sides of SMTP run on every mail-server.
- When a mail-server receives mail from other mail-servers, the mail-server acts as a server. When a mail-server sends mail to other mail-servers, the mail-server acts as a client.

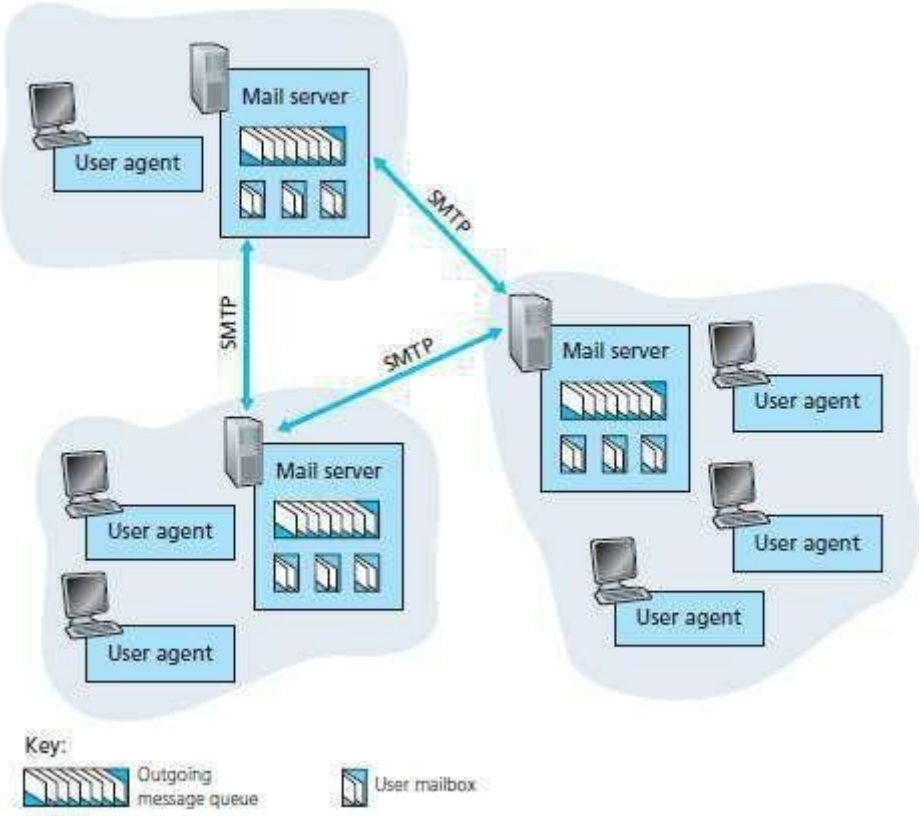


Figure 1.11: A high-level view of the Internet e-mail system

1.4.1 SMTP

- SMTP is the most important protocol of the email system.
- Three characteristics of SMTP (that differs from other applications):

1) Message body uses 7-bit ASCII code only.
2) Normally, no intermediate mail-servers used for sending mail.
3) Mail transmissions across multiple networks through mail relaying.

- Here is how it works:
  1) Usually, mail-servers are listening at port 25.
  2) The sending server initiates a TCP connection to the receiving mail-server.
  3) If the receiver's server is down, the sending server will try later.
  4) If connection is established, the client & the server perform application-layer handshaking.
  5) Then, the client indicates the e-mail address of the sender and the recipient.
  6) Finally, the client sends the message to the server over the same TCP connection.

1.4.2 Comparison of SMTP with HTTP

1) HTTP is mainly a pull protocol. This is because
   → someone loads information on a web-server and
   → users use HTTP to pull the information from the server.
   - On the other hand, SMTP is primarily a push protocol. This is because
     → the sending mail-server pushes the file to receiving mail-server. 2) SMTP requires each message to be in seven-bit ASCII format.
   - If message contains binary-data, the message has to be encoded into 7-bit ASCII format.
   - HTTP does not have this restriction.

3) HTTP encapsulates each object of message in its own response-message.
   - SMTP places all of the message's objects into one message.

1.4.3 Mail Access Protocols

- It is not realistic to run the mail-servers on PC & laptop. This is because
  → mail-servers must be always-on and
  → mail-servers must have fixed IP addresses
- Problem: How a person can access the email using PC or laptop?
- Solution: Use mail access protocols.
- Three mail access protocols:
  1) Post Office Protocol (POP)
  2) Internet Mail Access Protocol (IMAP) and
  3) HTTP.

1.4.3.1 POP

- POP is an extremely simple mail access protocol.
- POP server will listen at port 110.
- Here is how it works:
  - The user-agent at client's computer opens a TCP connection to the main server.
  - POP then progresses through three phases:

  - The user-agent sends a user name and password to authenticate the user.

  2) Transaction
  - The user-agent retrieves messages.
  - Also, the user-agent can
    → mark messages for deletion
    → remove deletion marks &
    → obtain mail statistics.
  - The user-agent issues commands, and the server responds to each command with a reply.
  - There are two responses:
    i) +OK: used by the server to indicate that the previous command was fine.
    ii) –ERR: used by the server to indicate that something is wrong.

  - After user issues a quit command, the mail-server removes all messages marked for deletion.
- Disadvantage:
  The user cannot manage the mails at remote mail-server. For ex: user cannot delete messages.

1.4.3.2 IMAP

- IMAP is another mail access protocol, which has more features than POP.
- An IMAP server will associate each message with a folder.

- When a message first arrives at server, the message is associated with recipient's INBOX folder
- Then, the recipient can
  → move the message into a new, user-created folder
  → read the message
  → delete the message and
  → search remote folders for messages matching specific criteria.
- An IMAP server maintains user state-information across IMAP sessions.
- IMAP permits a user-agent to obtain components of messages.
  For example, a user-agent can obtain just the message header of a message.

1.4.3.3 Web-Based E-Mail

- HTTPs are now used for Web-based email accessing.
- The user-agent is an ordinary Web browser.
- The user communicates with its remote-server via HTTP.
- Now, Web-based emails are provided by many companies including Google, Yahoo etc.

1.5 DNS — The Internet's Directory Service

- DNS is an internet service that translates domain-names into IP addresses.
  For ex: the domain-name ―www.google.com‖ might translate to IP address ―198.105.232.4‖.
- Because domain-names are alphabetic, they are easier to remember for human being.
- But, the Internet is really based on IP addresses (DNS → Domain Name System).

1.5.1 Services Provided by DNS

- The DNS is
  1) A distributed database implemented in a hierarchy of DNS servers.
  2) An application-layer protocol that allows hosts to query the distributed database.
- DNS servers are often UNIX machines running the BIND software.
- The DNS protocol runs over UDP and uses port 53. (BIND → Berkeley Internet Name Domain)
- DNS is used by application-layer protocols such as HTTP, SMTP, and FTP.
- Assume a browser requests the URL www.someschool.edu/index.html.
- Next, the user's host must first obtain the IP address of www.someschool.edu
- This is done as follows:
  1) The same user machine runs the client-side of the DNS application.
  2) The browser
  → extracts the hostname ―www.someschool.edu‖ from the URL and
  → passes the hostname to the client-side of the DNS application.
  3) The client sends a query containing the hostname to a DNS server.
  4) The client eventually receives a reply, which includes the IP address for the hostname.
  5) After receiving the IP address, the browser can initiate a TCP connection to the HTTP server.
- DNS also provides following services:

1) Host Aliasing

- A host with a complicated hostname can have one or more alias names.

2) Mail Server Aliasing

- For obvious reasons, it is highly desirable that e-mail addresses be mnemonic.

3) Load Distribution

- DNS is also used to perform load distribution among replicated servers.
- Busy sites are replicated over multiple servers & each server runs on a different system.

1.5.2 Overview of How DNS Works

- Distributed database design is more preferred over centralized design because:

1) A Single Point of Failure

- If the DNS server crashes then the entire Internet will not stop.

2) Traffic Volume

- A Single DNS Server cannot handle the huge global DNS traffic.
- But with distributed system, the traffic is distributed and reduces overload on server.

3) Distant Centralized Database

- A single DNS server cannot be ―close to‖ all the querying clients.
- If we put the single D S server in Mysore,
  then all queries from USA must travel to the other side of the globe.
- This can lead to significant delays.

4) Maintenance

- The single DNS server would have to keep records for all Internet hosts.
- his centralized database has to be updated frequently to account for every new host.

1.5.2.1 A Distributed, Hierarchical Database

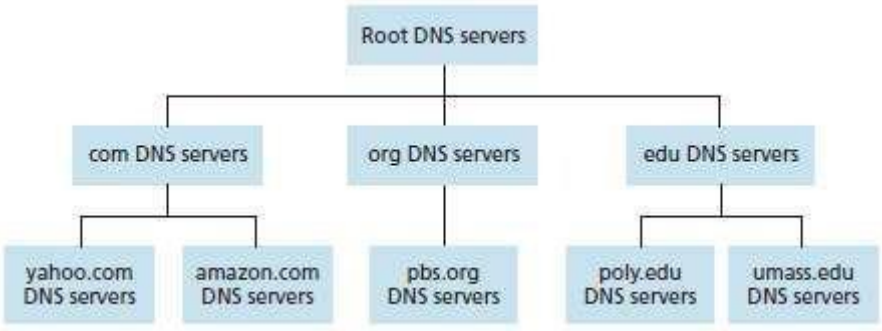


Figure 1.12: Portion of the hierarchy of DNS servers

- Suppose a client wants to determine IP address for hostname ―www.amazon.com‖ (Figure 1.12):
  1) The client first contacts one of the root servers, which returns IP addresses for TLD servers
  2) Then, the client contacts one of these TLD servers.
  - The TLD server returns the IP address of an authoritative-server for ―amazon.com‖. 3) Finally, the client contacts one of the authoritative-servers for amazon.com.
  - The authoritative-server returns the IP address for the hostname ―www.amazon.com‖.

1.5.2.1.1 Recursive Queries & Iterative Queries

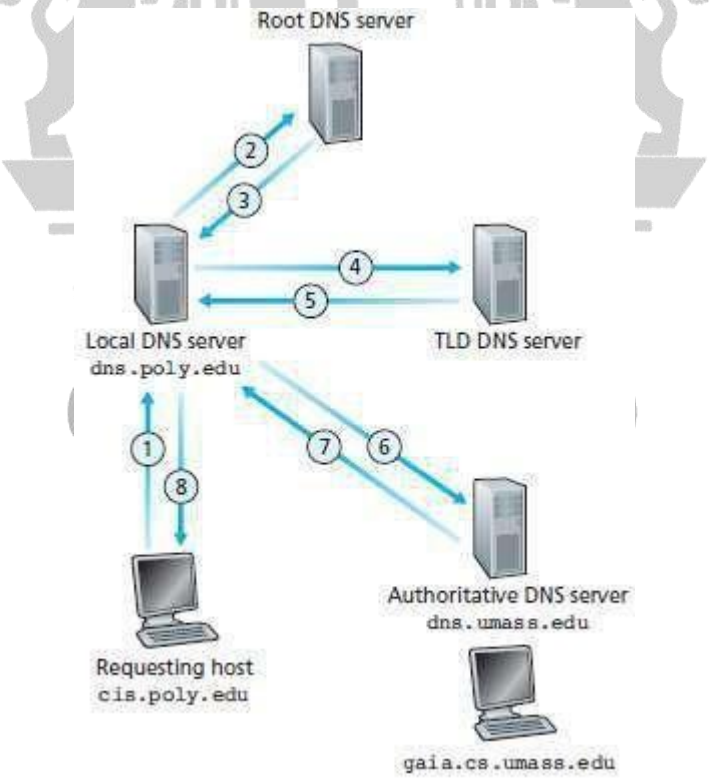


Figure 1.13: Interaction of the various DNS servers

- The example shown in Figure 1.13 makes use of both recursive queries and iterative queries.
- The query 1 sent from cis.poly.edu to dns.poly.edu is a recursive query. This is because
  → the query asks dns.poly.edu to obtain the mapping on its behalf.
- But the subsequent three queries 2, 4 and 6 are iterative. This is because
  → all replies are directly returned to dns.poly.edu.

1.5.3 DNS Records & Messages

- The DNS server stores resource-records (RRs).
- RRs provide hostname-to-IP address mappings.
- Each DNS reply message carries one or more resource-records.
- A resource-record is a 4-tuple that contains the following fields: (Name, Value, Type, TTL)
- TTL (time to live) determines when a resource should be removed from a cache.
- The meaning of Name and Value depend on Type:

1) If Type=A, then Name is a hostname and Value is the IP address for the hostname.
   - Thus, a Type A record provides the standard hostname-to-IP address mapping. For ex: (relay1.bar.foo.com, 145.37.93.126, A)

2) If Type=NS, then
   i) Name is a domain (such as foo.com) and
   ii) Value is the hostname of an authoritative DNS server.
   - This record is used to route DNS queries further along in the query chain.

3) If Type=CNAME, then Value is a canonical hostname for the alias hostname Name.
   - This record can provide querying hosts the canonical name for a hostname. For ex: (foo.com, relay1.bar.foo.com, CNAME) is a CNAME record.

4) If Type=MX, Value is the canonical name of a mail-server that has an alias hostname Name. ➢ MX records allow the hostnames of mail-servers to have simple aliases.
   For ex: (foo.com, mail.bar.foo.com, MX) is an MX record.

1.5.3.1 DNS Messages

- Two types of DNS messages: 1) query and 2) reply.
- Both query and reply messages have the same format.

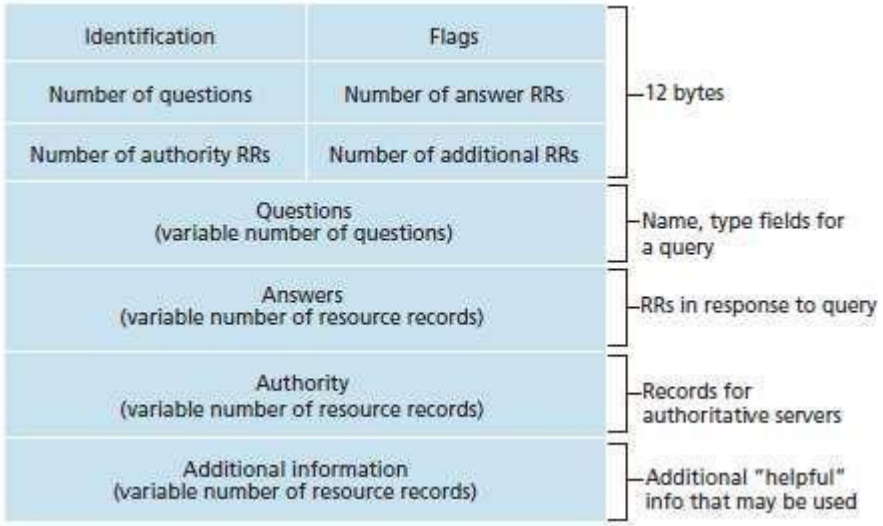


Figure 1.14: DNS message format

- The various fields in a DNS message are as follows (Figure 1.14):

1) Header Section

- The first 12 bytes is the header-section.
- This section has following fields:

i) Identification

- This field identifies the query.
- This identifier is copied into the reply message to a query.
- This identifier allows the client to match received replies with sent queries.

ii) Flag

- This field has following 3 flag-bits:

a) Query/Reply

¤ This flag-bit indicates whether the message is a query (0) or a reply (1). **b) Authoritative**

¤ This flag-bit is set in a reply message when a DNS server is an authoritative-server.

c) Recursion Desired

¤ This flag-bit is set when a client desires that the DNS server perform recursion.

iii) Four Number-of-Fields

 These fields indicate the no. of occurrences of 4 types of data sections that follow the header.

2) Question Section

- This section contains information about the query that is being made.
- This section has following fields:

i) Name

 This field contains the domain-name that is being queried.

ii) ype

 his field indicates the type of question being asked about the domain-name.

3) Answer Section

- This section contains a reply from a DNS server.
- This section contains the resource-records for the name that was originally queried.
- A reply can return multiple RRs in the answer, since a hostname can have multiple IP addresses.

4) Authority Section

- This section contains records of other authoritative-servers.
- This section contains other helpful records.

1.6 Peer-to-Peer Applications

- Peer-to-peer architecture is different from client-server architecture.
- In P2P, each node (called peers) acts as a client and server at the same time.
- The peers are not owned by a service-provider.
- The peers not supposed to be always listening on the Internet.
- The peers are dynamic, i.e., some peers will join some peers will leave from time to time.

1.6.1 P2P File Distribution

- One popular P2P file distribution protocol is BitTorrent
- Consider the following scenarios:

Suppose a server has a large file and ‗N' computers want to download the file (Figure 1.15).

1) In client-server architecture, each of the N computers will
   → connect to the server &
   → download a copy of the file to local-host.
2) In P2P architecture, a peer need not necessarily download a copy from the server. ➢ Rather, the peer may download from other peers.

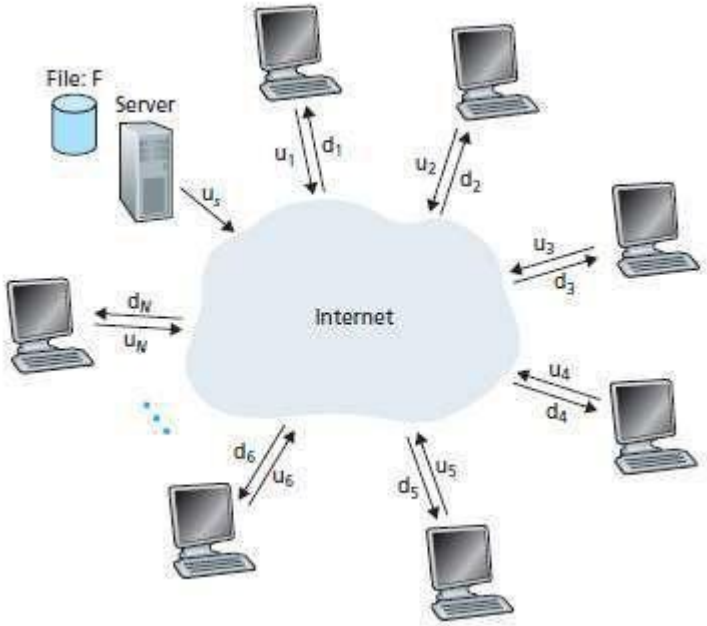


Figure 1.15: An illustrative file distribution problem

- Let us define the following: Length of the file =
  F Upload rate for the server = u_s Upload

rate for ith computer = u_i Download-rate
for ith computer = d_i

Distribution-time for the client-server architecture = D_{cs}
Server needs transmit = N_F bits.
Lowest download-rate of peer = d_{min}
Distribution-time for the P2P architecture =
D_{P2P}

Case 1: Client-Server Architecture

- We have 2 observations:
  1) The server must transmit one copy of the file to each of the N peers.
     - Since the server's upload rate is u_s, the distribution-time is at least N_F/u_s.
  2) The peer with the lowest download-rate cannot obtain all F bits of the file in less than F/d_{min}.
     - Thus, the minimum distribution-time is at least F/d_{min}.
- Putting above 2 observations together, we have

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

Case 2: P2P Architecture

- We have 2 observations:
  1) At the beginning of the distribution, only the server has the file.
     - So, the minimum distribution-time is at least F/u_s.
  2) The peer with the lowest download-rate cannot obtain all F bits of the file in less than F/d_{min}.
     - Thus, the minimum distribution-time is at least F/d_{min}.
  3) The total upload capacity of the system as a whole is $u_{total} = u_s + u_1 + u_2 \ldots + u_N$.
     - The system must deliver F bits to each of the N peers.
     - Thus, the minimum distribution-time is at least $NF/(u_s + u_1 + u_2 \ldots + u_N)$.
- Putting above 3 observations together, we have

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

- Figure 1.16 compares the minimum distribution-time for the client-server and P2P architectures.

Figure 1.16: Distribution-time for P2P and client-server architectures

- The above comparison shows that when N is large,
  i) P2P architecture is consuming less distribution-time.
  ii) P2P architecture is self-scaling.

1.6.1.1 BitRorrent

- The collection of all peers participating in the distribution of a particular file is called atorrent.
- Peers download equal-size chunks of the file from one another. Chunk size = 256KBytes.

- The peer also uploads chunks to other peers.
- Once a peer has acquired the entire file, the peer may leave the torrent or remain in the torrent.
- Each torrent has an infrastructure node called tracker.
- Here is how it works (Figure 1.17):
  1) When a peer joins a torrent, the peer
     → registers itself with the tracker and
     → periodically informs the tracker that it is in the torrent.
  2) When a new peer joins the torrent, the tracker
     → randomly selects a subset of peers from the set of participating peers and
     → sends the IP addresses of these peers to the new peer.
  3) Then, the new peer tries to establish concurrent TCP connections with all peers on this list. ➢ All peers on the list are called neighboring-peers.
  4) Periodically, the new peer will ask each of the neighboring-peers for the set of chunks.
- To choose the chunks to download, the peer uses a technique called rarest-first.
- Main idea of rarest-first:
  → Determine the chunks that are the rarest among the neighbors and
  → Request then those rarest chunks first.

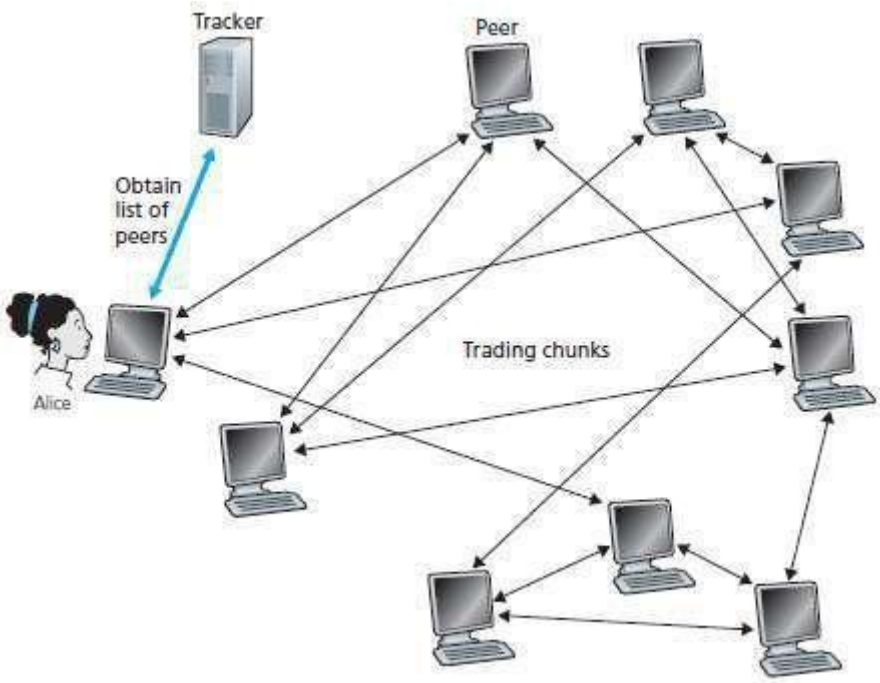


Figure 1.17: File distribution with BitTorrent

1.6.2 Distributed Hash Table

- We can use P2P architecture to form a distributed database.
- We consider a simple database, which contains (key, value) pairs.
- Each peer will only hold a small subset of the data.
- Any peer can query the distributed database with a particular key.
- Then, the database will
  → locate the peers that have the corresponding (key, value) pairs and
  → return the key-value to the querying peer.
- Any peer will also be allowed to insert new key-value pairs into the database.
- Such a distributed database is referred to as a distributed hash table (DHT).
- To construct the database, we need to use some hash-functions.
- The input of a hash-function can be a large number.

- The outline of building a DHT is as follows:
  1) Assign an identifier to each peer, where the identifier is an n-bit string.
     So, we can view the identifier as an integer at the range from 0 to 2^n-1.
  2) For a data pair (key, value), the hash value of the key is computed.
     Then the data is stored in the peer whose identifier is closest to the key.
  3) To insert or retrieve data, first we need to find the appropriate peer.

- Problem: It is not realistic to let the peer to store all of the other peer's identifiers. Solution: Use a circular arrangement.

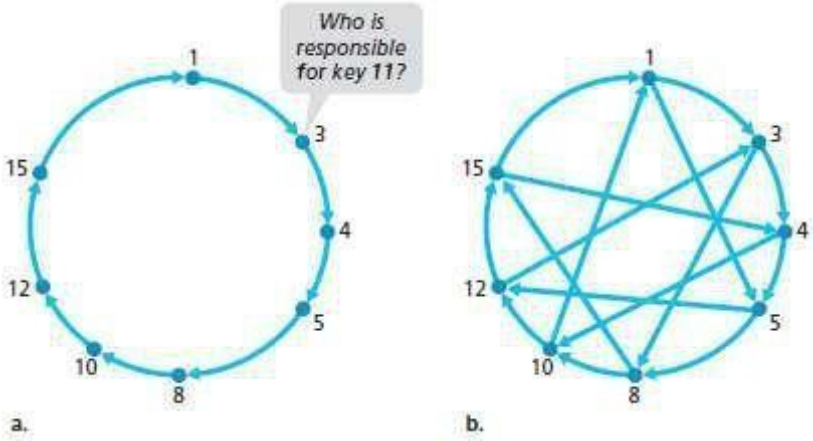


Figure 1.18: (a) A circular DHT. Peer 3 wants to determine who is responsible for key 11. (b) A circular DHT with shortcuts

1.6.2.1 Circular Arrangement

- In this example, the identifiers are range [0, 15] (4-bit strings) and there are eight peers.
- Each peer is only aware of its immediate successor and predecessor (Figure 1.18a).
- So, each peer just needs to track two neighbors.
- When a peer asks for a key, the peer sends the message clockwise around the circle.
- For example:
  - The peer 4 sends a message saying –Who is responsible for key 11?"
    - he message will forward through peers 5, 8, 10 and reach peer 12.
    - he peer 12 determines that it is the closest peer to key 11.
    - At this point, peer 12 sends a message back to the querying peer 4.
- But when the circle is too large, a message may go through a large number of peers to get answer.
- Problem: There is trade off between
  - i) Number of neighbors each peer has to track and
  - ii) Number of message to be sent to resolve a single query.
- Solution: To reduce the query time, add –shortcuts" to the circular arrangement (Figure 1.18b).

1.7 Socket Programming: Creating Network Applications

- Two types of network-applications:
  - 1) First type is an implementation whose operation is specified in a protocol standard (RFC)
    - Such an application is referred to as –open|.
    - The client & server programs must conform to the rules dictated by the RFC.
  - 2) Second type is a proprietary network-application.
    - The client & server programs use an application-layer protocol not openly published in a RFC.
    - A single developer creates both the client and server programs.
    - The developer has complete control over the code.
- During development phase, the developer must decide whether the application uses TCP or UDP.

1.7.1 Socket Programming with UDP

- Consider client-server application in which following events occurs:
  - 1) The client
    - → reads a line of characters (data) from the keyboard and
    - → sends the data to the server.
  - 2) The server receives the data and converts the characters to uppercase.
  - 3) The server sends the modified data to the client.
  - 4) The client receives the modified data and displays the line on its screen.

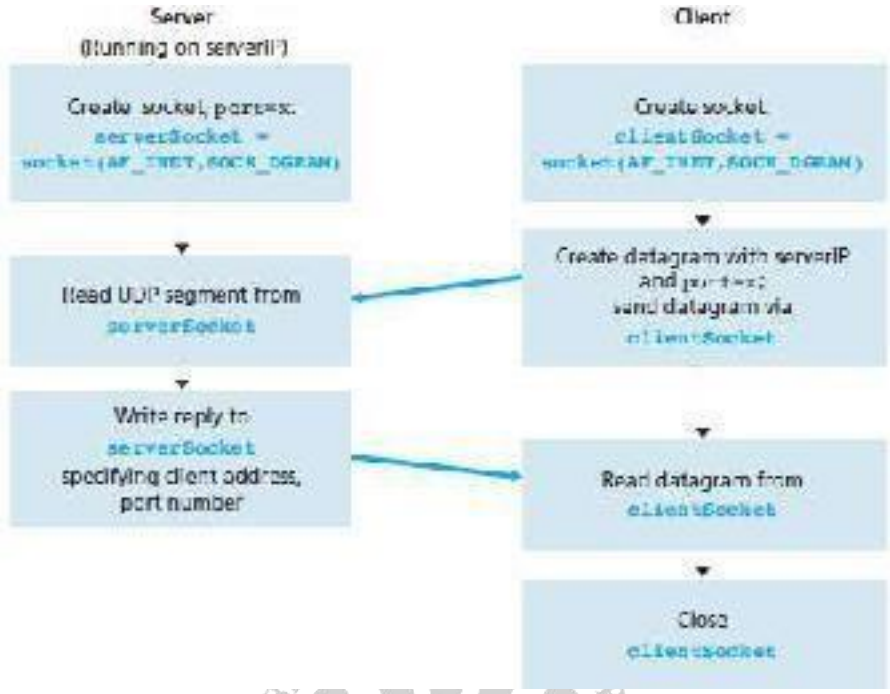


Figure 1.19: The client-server application using UDP

- The client-side of the application is as follows (Figure 1.19): from socket

```
import * //This line declares socket within the program. serverName =
„hostname"  // This line sets the server name to "hostname".
serverPort = 12000          // This line sets the server port# to "12000".
clientSocket = socket(socket.AF INET, socket.S CK DGRAM) //This line creates the
client" s socket message = raw input(" Input lowercase sentence:" ) //This line reads
the data at the client clientSocket.sendto(message,(server ame, serverPort)) //This
line sends data into the socket modifiedMessage, serverAddress =
```

Here, AF_INET indicates address family
SOCK_DGRAM indicates UDP as socket type contains server's IP address & port#

- The server-side of the application is as

```
follows: from socket
import * serverPort =
12000
serverSocket = socket(AF INET, SOCK DGRAM)
serverSocket.bind(("", serverPort)) // This line assigns the port# 12000 to the
server" s socket. print "The server is ready to receive"
while 1:
message, clientAddress = serverSocket.recvfrom(2048)
```


1.7.2 Socket Programming with TCP

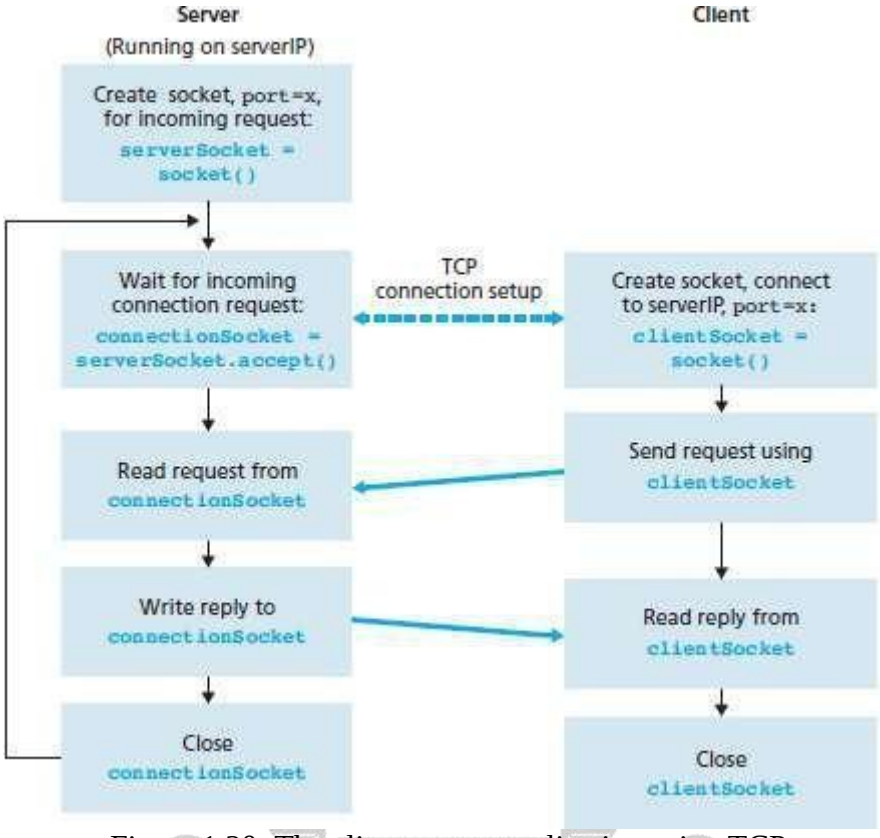


Figure 1.20: The client-server application using TCP

- The client-side of the application is as follows (Figure 1.20):

```
from socket import *
serverName =
" servername"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort)) // This line initiates TCP connection
b/w client & server sentence = raw_input(„Input lowercase sentence:" )
clientSocket.send(sentence)
modifiedSentence =
```


- The server-side of the application is as follows:

```
from socket
import *
serverPort =
12000
serverSocket = socket(AF_INET,SOCK
STREAM)
serverSocket.bind((„",serverPort))
serverSocket.listen(1) // This line specifies no. of connection-requests from the client to
server
print „ he server is ready to
```