An undertaking of Bhaktapur Municipality
Khwopa College of Engineering
Affiliated to Tribhuvan Univeristy
Libali, Bhaktapur, Nepal

A
Note
on
Computer Network

Compiled By

Er. Dinesh Ghemosu

Table of Contents

The unexamined life is not worth living.

Socrates

6 Application Layer

- At the core of network application development is writing programs that run on different end systems and communicate with each other over the network.
- For example, in the web application there are two distinct programs that communicate with each other: the browser program running in the user's host; and the web server program running in the web server host.
- Thus, when developing your new application, you need to write software that will run on multiple end systems. This software could be written, for example, in C, Java, or Python.
- There is no need to write software for network-core devices.
- The network-core devices do not run user applications.

6.1 Network Application Architectures

- Designed by the application developer and dictates how the application is structured over the various end systems.
- Two types:
i. Client server architecture
ii. Peer-to-peer (P2P) architecture

6.1.1 Client Server Architecture

- There is always-on host, called the server, which services the requests from many other hosts, called clients. The clients host can be either sometimes-on or always-on.
- Often in a client-server application, a single server host is incapable of keeping up with all requests from the its clients.
-Fort his reason, a large cluster hosts-sometimes referred to as a data center -is often used to create a powerful virtual server in client-server architecture.
- Some of the better-known applications with a client-server architecture include the Web, FTP, Telnet, and e-mail.

6.1.2 Peer-to-Peer (P2P) Architecture

- There is no always-on server.
- An arbitrary end systems directly communicate.
- Peers request service from other peers, provide service in return to other peers.
    • self scalability – new peers bring new service capacity, as well as new service demands.
- Peers are intermittently connected and change IP addresses.
    • Complex management

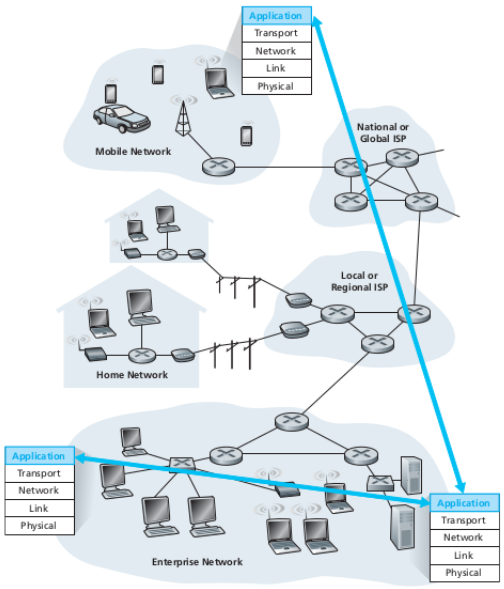


Figure 6.1: Communication for a network application takes place between end systems at the application layer.

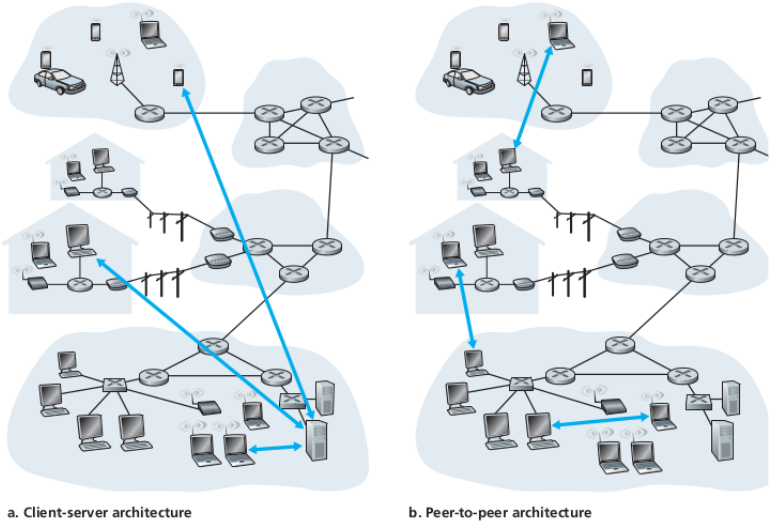


Figure 6.2: Network architecture.

- Cost effective: don't require significant server infrastructure and server bandwidth.
- These application include file distribution (e.g. BitTorrent), file sharing (eMule and LimeWire), Internet telephony(e.g. skype), and IPTV.

6.2 Process Communication

- Process is a program running within a host.
- Within same host, two processes communicate using inter-process communication.
- Processes in different host communicate by exchanging messages across the computer network.

Client and sever process:
- In the context of a communication session between a pair of processes, the process that initiates the communication (that is, initially contacts the other process at the beginning of the session) is labeled as the *client*. The process that waits to be contacted to begin the session is the *server*.

- A network application consists of pairs of processes that send messages to each other over a network. For example, in the Web application a client browser process exchanges messages with a Web server process. In a P2P file-sharing sys- tem, a file is transferred from a process in one peer to a process in another peer. For each pair of communicating processes, we typically label one of the two processes as the client and the other process as the server. With the Web, a browser is a *client process* and a *Web server* is a server process. With P2P file sharing, the peer that is downloading the file is labeled as the client, and the peer that is uploading the file is labeled as the server.

Socket
- A process sends message into, and receives message from, the network through a software interface called a **socket**.
- socket analogous to door:

  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process

- It is also referred to as the ***Application Programming Interface (API)*** between the application and the network, since the socket is the programming interface with which network applications are built.
- The application developer has control of everything on the application-layer side of the socket but has little control of the transport-layer side of the socket. The only control that the application developer has on the transport-layer side is (1) the choice of transport protocol and (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes . Once the application developer chooses a transport protocol (if a choice is available), the application is built using the transport-layer services provided by that protocol. (Figure 6.3)

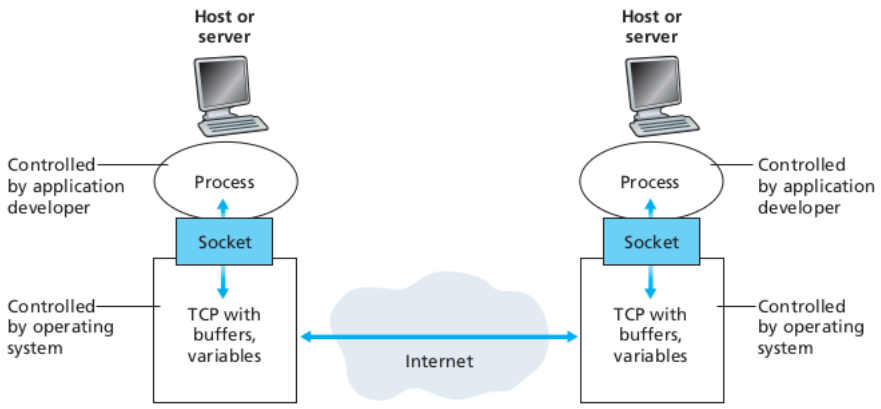


Figure 6.3: Application processes, sockets, and underlying transport protocol.

TCP Services
- reliable transport between sending and receiving process
- flow control: sender won't overwhelm receiver
- congestion control: throttle sender when network overloaded
- does provide: timing, minimum throughput guarantee, security
- connection-oriented: setup required between client and server processes

UDP Services
- UDP is a no-frills, lightweight transport protocol, providing minimal services.
- UDP is connectionless, so there is no handshaking before the two processes start to communicate.
- UDP provides an unreliable data transfer service—that is, when a process sends a message into a UDP socket, UDP provides no guarantee that the message will ever 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 mechanism, so the sending side of UDP can pump data into the layer below (the network layer) at any rate it pleases. (Note, however, that the actual end-to-end throughput may be less than this rate due to the limited transmission capacity of intervening links or due to congestion).

When should we use the UDP instead of TCP?
- UDP has a few properties that make it useful in ways that TCP doesn't readily support.
- UDP has a lower overhead, since there is no ongoing connection to maintain. Faster transmission. - UDP can be broadcast or multicast, so it allows a one-to-many protocol to exist.
- UDP is used when some loss is acceptable. (VoIP). It is up to the upper layer protocols to compensate for loss, delay, and jitter by doing things like reducing the codec quality and therefore reducing the bandwidth consumption.
- This is particularly useful in querying an unknown network for the existence of some device or property. (ICMP)
- These things are hard to implement efficiently in TCP.

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Figure 6.4: Popular Internet applications, their application-layer protocols, and their underlying transport protocols.

Figure 6.4 indicates the transport protocols used by some popular Internet applications. We see that e-mail, remote terminal access, the Web, and file trans- fer all use TCP. These applications have chosen TCP primarily because TCP pro- vides reliable data transfer, guaranteeing that all data will eventually get to its destination. Because Internet telephony applications (such as Skype) can often tolerate some loss but require a minimal rate to be effective, developers of Inter- net telephony applications usually prefer to run their applications over UDP, thereby circumventing TCP's congestion control mechanism and packet over- heads. But because many firewalls are configured to block (most types of) UDP traffic, Internet telephony applications often are designed to use TCP as a backup if UDP communication fails.

6.3 Application Layer Protocols

- An application layer protocol defines how an application's processes, running on different end systems, pass messages to each other.
- In particular, an application layer protocol defines:

i The types of message exchanged, for example, request messages and respond messages.
ii The syntax of the various message types, such as the fields in the message and how the fields are delineated.
iii The semantics of the fields, that is, the meaning of information in the fields.
iv Rules for determining when and how a process sends messages and responds to messages

6.4 The Web, HTTP and HTTPs

- World Wide Web(WWW) developed by Berneres-Lee in 1994.
- It was the first Internet application program.
- Hyper Text Transfer Protocol (HTTP) is the web's application-layer protocol for distributed, collaborative, hypermedia information systems.
- This is the foundation for data communication for the World Wide Web (i.e. internet).
- Allows the transmitting and receiving of information across the internet.
- It is defined in RFC 1945 and RFC 2616 (Request for Comments)
- HTTP is implemented in two program: client program and server program, executing on different end systems, talks to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and how the client and server exchange the messages.
- It used TCP connection using port number 80.

- The '**S**' Stands for "**Secure**" in HTTPS i.e. *Secure hypertext transfer protocol*.
- HTTPS is HTTP using a Secure Socket Layer (SSL).
  - Secure HTTP (HTTPS) is one of the popular protocols to transfer sensitive data over the Internet.
  - A secure socket is an encryption protocol invoked on a Web Server that uses HTTPS.
- Most Implementations of the HTTPS protocol involve online purchasing or the exchange of private information.
- HTTPS is only slightly slower than HTTP.
- Why is HTTPS not used for web traffic?
  - Slows down web servers
  - Breaks Internet Caching
    - ISP cannot cache HTTPS traffic
    - Results in increased traffic at web site.

WWW: Wordl Wide Web

- It is a repository of information linked together from points all over the world.
- It consists of a vast, worldwide collection of documents or web pages. Each page may contain link to other pages anywhere in the world, called *hyperlink*.
- The pages are retrieved and viewed by using a *browser*.

Web Page

- A web page consist of objects.
- An object can be HTML file, JPEG image, Java applet, audio file etc.
- Web page consists of base HTML-file which includes several referenced objects.
- Each object is addressable by URL. (Universal Resource Locator).

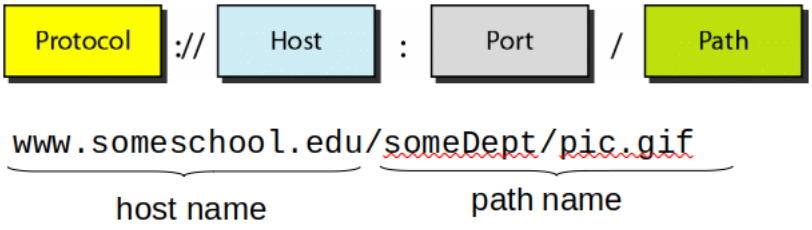


Figure 6.5: URL example.

Browser

- A web browser is an application program that interpret and display the contents of web pages.
- Each browser has three parts: a controller, client protocol, and interpreter.
- A controller
  - receives input from keyboard or mouse and uses the client program to access the document.
  - use interpreter to display the content of web page
- Client protocol : HTTP, FTP, FILE etc.
- Interpreter : HTML, Java, JavaScript etc.

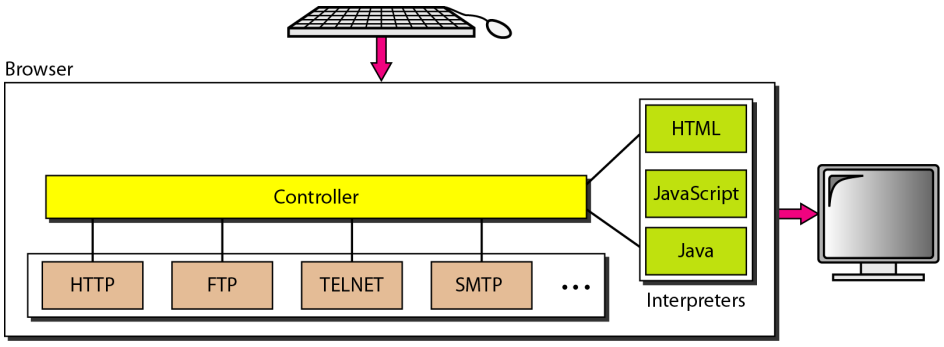


Figure 6.6: A broswer architecture.

6.4.1 Architecture of WWW

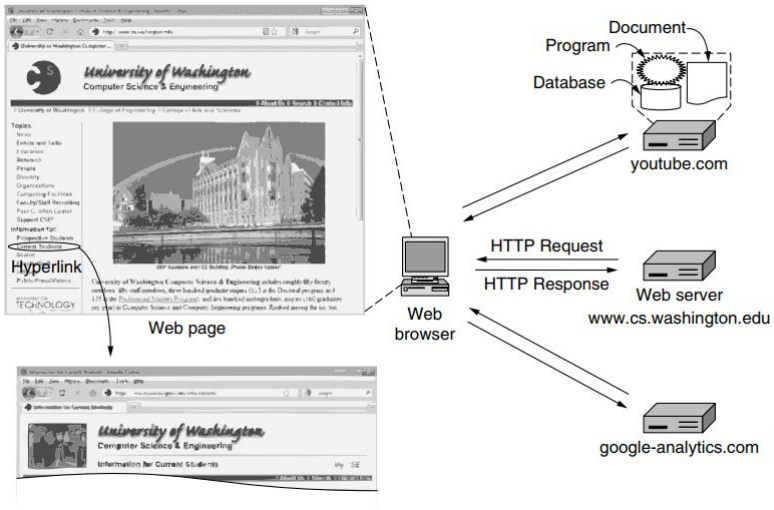


Figure 6.7: A WWW architecture.

- WWW is based on HTTP request and HTTP response. Client initiates HTTP request and server proivides th object requested by client as HTTP response.
- When a user clicks on a hyperlink, the browser carries out a series of steps in order to fetch the page pointed to.
- Let us trace the steps that occur when our example link is selected:

1. The browser determines the URL (by seeing what was selected).

2. The browser asks DNS for the IP address of the server www.cs.washington.edu.
3. DNS replies with 128.208.3.88.
4. The browser makes a TCP connection to 128.208.3.88 on port 80, the well-known port for the HTTP protocol.
5. It sends over an HTTP request asking for the page /index.html.
6. The www.cs.washington.edu server sends the page as an HTTP response, for example, by sending the file /index.html.
7. If the page includes URLs that are needed for display, the browser fetches the other URLs using the same process. In this case, the URLs include multiple embedded images also fetched from www.cs.washington.edu, an embedded video from youtube.com, and a script from google-analytics.com.
8. The browser displays the page /index.html as it appears in above figure.
9. The TCP connections are released if there are no other request to the same servers for a short period.

6.4.2 HTTP Transaction

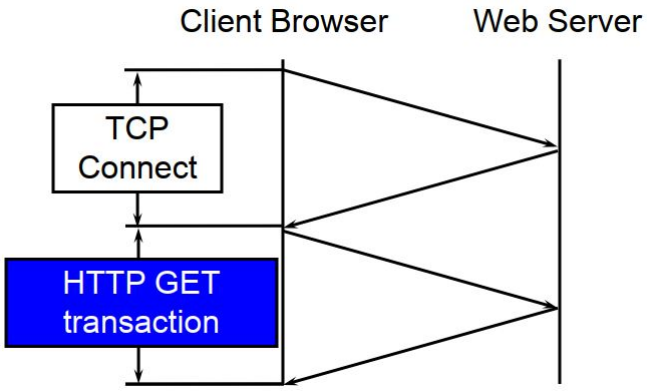


Figure 6.8: HTTP transaction.

- HTTP transaction between the client and the server.
- There are two transaction messages.
i. Request (sent from client to server for requesting a page of other resource).
ii. Response (sent from server to client).

Client/Server model
- client: browser that requests, receives, (using HTTP protocol) and "displays" Web objects.
- server: Web server sends (using HTTP protocol) objects in response to requests.

Figure 6.9: HTTP transaction example.

HTTP uses TCP as its underlying transport protocol.
Process
- The HTTP client first initiates a TCP connection with the server.
- Once the connection is established, the browser and the server processes access TCP through their socket interfaces (uses port 80).

- HTTP messages are then (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server) through their socket interface.
- TCP connection closed.

Note:
- HTTP *need not worry about lost data or the details of how TCP recovers from loss or reordering of data within the network*. That is the job of TCP and the protocols in the lower layers of the protocol stack.
- HTTP is *stateless* sever maintains no past client requests. i.e. even a client requests same object twice in few seconds, the server does not respond by saying the object has been served to the client; instead, the server resends the object, as it has completely forgotten about what it did earlier.

6.4.3 HTTP Connection

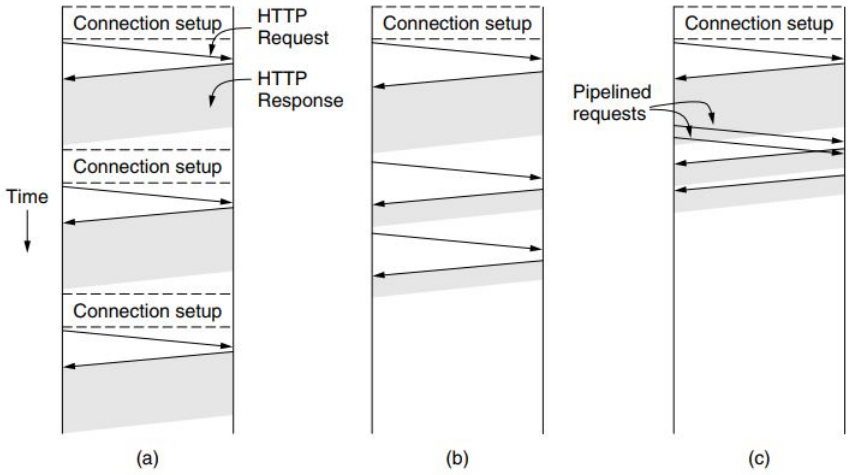


Figure 6.10: HTTP with (a) multiple connections and sequential requests. (b) A persistent connection and sequential requests. (c) A persistent connection and pipelined requests.

6.4.3.1 Nonperistent connection
- In non-persistent connection, one TCP connection is made for each request/response.
- The following list the steps in this strategy:
  - The client opens a TCP connection and sends a request.
  - The server sends the response and close the connection.
  - The client reads the data until it encounters end-of-file marker; it then closes the connection.

Round-Time Trip (RTT):
- RTT is the time it takes for a small packet to travel from client to server and back.
- The RTT includes packet propagation delay, packet queuing delay, packet processing delay.
consider what happens the user clicks the hyperlink.

HTTP response time:
- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return file transmission time
- non-persistent HTTP response time = 2RTT+ file transmission time

Shortcomings of Non-persistent connections
- First, a brand-new connection must be established and maintained for each requested object. For each these connections, TCP buffers must be allocated and TCP variables must be kept in both the client and server. This can place a significant burden on the web server.
- Secondly, each object suffers a delivery delay of two RTTs-one RTT to establish the TCP connection and one RTT to request and receive and object.

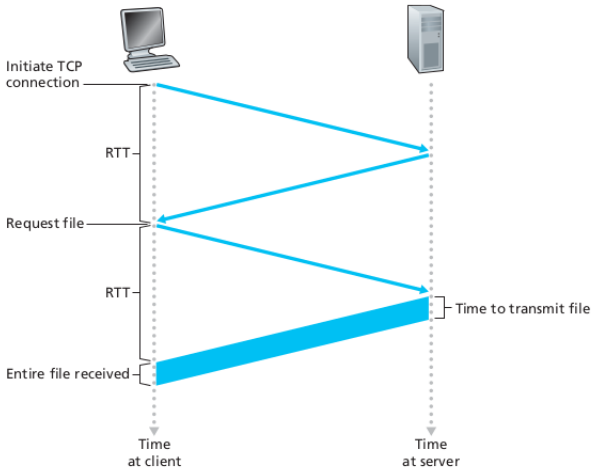


Figure 6.11: Back-of-the-envelope calculation for the time needed to request and receive an HTML file.

6.4.3.2 Persistance Connection

- The server leaves the TCP connection open after sending a response.
- Subsequent requests and responses between the same client and server can be sent over the same connection.
- Typically, the HTTP serer closes a connection when it isn't used for a certain time (a configurable timeout interval)
- When the server receives the back-to-back request, it sends the objects back-to-back.
- The default mode of HTTP uses persistent connections with pipelining.

6.4.4 HTTP Methods

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

Figure 6.12: HTTP methods.

6.4.5 HTTP Request

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Figure 6.13: HTTP responses.

6.4.6 HTTPS

- HTTPS stands for Hypertext Transfer Protocol over Secure Socket Layer, or HTTP over SSL. SSL acts like a sub layer under regular HTTP application layering.
- HTTPS encrypts an HTTP message prior to transmission and decrypts a message upon arrival. - HTTPS by default uses port 443 as opposed to the standard HTTP port of 80.
- URL's beginning with HTTPS indicate that the connection between client and browser is encrypted using SSL.

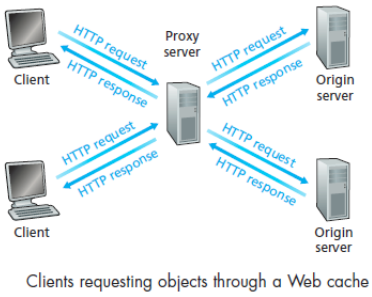


Figure 6.14: Proxy server.

6.4.7 Proxy Server/Web Caching

- Proxy server is a computer that keeps the copies of responses to recent requests.
- The HTTP client sends request to the proxy server. The proxy server checks its cache. It the response is not stored in the cache, the proxy server sends the request to the corresponding server. Incoming responses are sent to the proxy server and stored for future requests from other clients.
- The proxy server reduces the load on the original server, decreases traffic, and improves latency.

- A web cache is purchased and installed by an ISP.

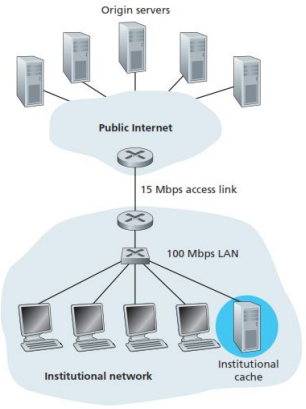


Figure 6.15: Web caching.

6.4.8 Web Application Server

- Web server is a computer where the web content is stored.
- It is used to host the web sites but there exists other web servers also such as gaming, storage, email etc.

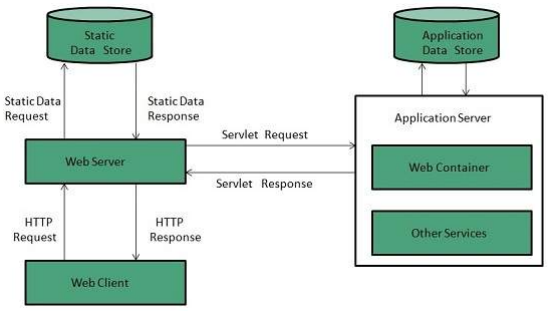


Figure 6.16: Web applicaiton server.

6.4.9 Mail Application Server

- A mail sever is an application that receives incoming e-mail from the local users and forwards outgoing for delivery. A computer dedicated to running such application is mail server. Microsoft Exchange, sendmail, qmail

are some common mail server programs.
- It is also a mail transfer agent or MTA, a mail transport agent , a mail router or an Internet mailer.

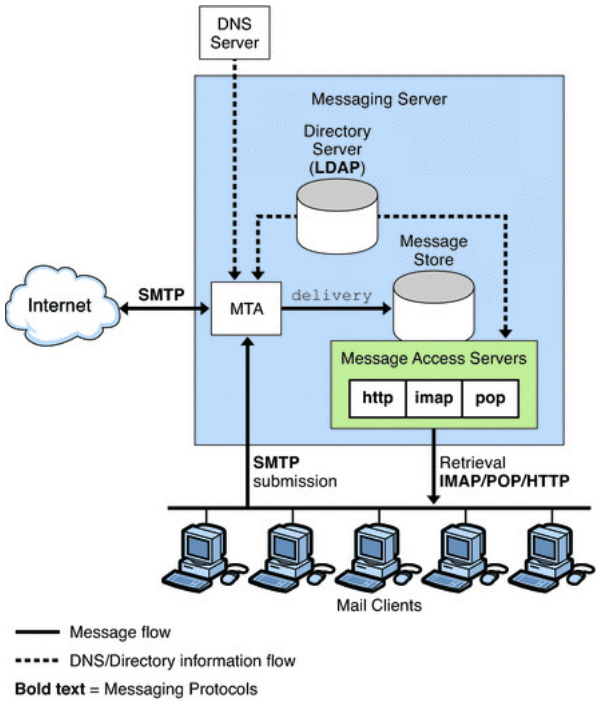


Figure 6.17: Mail server.

6.5 Internet Cookies

Actual definition:
- Short pieces of data used by web servers to help identify web users.
- Places on your hard drive by server.
- Identifies your IP address during your visit.
- Helpful on shopping sites.

- Web browser request a file from a server.
- If no cookie, serer issues one.
- If already existent, cookie is sent to server.
- Server reads cookie and sends it back with update.

What information can coolies return?
- IP address
- Your type of browser
- Operating system
- Number of visit to web site

Who can see the cookies?
- Only the server that put the cookie on your hard drive.

Why a bad reputation for cookies?
- When used as a tracking device.
- Can see what web pages your visit, how often, and how long.
- Clicking on ad banners.
- Used to infer your interests.

6.6 File Transfer Protocol

- File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file to/from one host to another.
  - FTP establishes two connections between the hosts.
  - One connection is used for data transfer, the other for control information (commands and responses).
- FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection.
- Separation of commands and data transfer makes FTP more efficient.

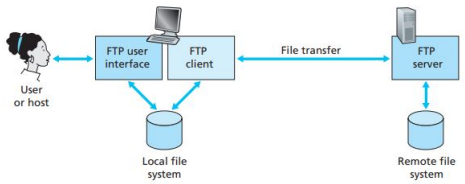


Figure 6.18: FTP moves files between local and remote file systems.

Figure 6.19: Data and control connection.

- In a typical FTP session, the user is sitting in front of one host (the local host) and wants to transfer files to or from a remote host.
- In order for the user to access the remote account, the user must provide a user identification and a password.
- After providing this authorization information, the user can transfer files from the local file system to the remote file system and vice versa.
- As shown in Figure 6.18, the user interacts with FTP through an FTP user agent.
- The user first provides the hostname of the remote host, causing the FTP client process in the local host to establish a TCP connection with the FTP server process in the remote host.
- The user then provides the user identification and password, which are sent over the TCP connection as part of FTP commands.
- Once the server has authorized the user, the user copies one or more files stored in the local file system into the remote file system (or vice-versa).

6.6.1 FTP Architecture

The client has three components:
i. User interface
ii. Control process
iii. Data transfer process
The server has two components:
i. Control process
ii. Data transfer process

FTP Working Principle

- FTP client contacts FTP server at port 21, using TCP.

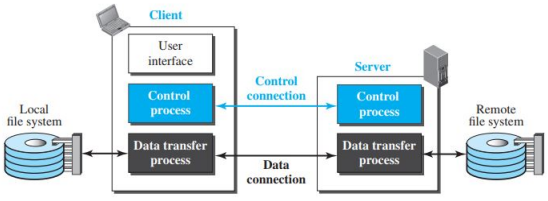


Figure 6.20: FTP architecture.

- Client sends the user identification and password over control connection.
- Client also sends commands over control connection to change the remote directory.
- When server receives file transfer command, server opens 2nd TCP data connection (for file) to client
- After transferring one file, server closes data connection.
- Server opens another TCP data connection to transfer another file.
- Sontrol connection: "*out of band*".
- FTP server maintains "state": current directory, earlier authentication.

6.6.2 FTP: Commands and Response

- Sent across the control connection in 7-bit ASCII format.
- Like HTTP, FTP commands are readable by people.
- In order to delineate successive commands, a carriage return and line feed end each command.
- Each command consists of four uppercase ASCII characters, some with optional arguments.

Commands	Meaning
USER username	Used to send the user identification to the server
PASS password	Used to send the user password to the server
LIST	Used to ask the server to send back a list of all the files in the current remote directory. The list of files is sent over a (new and non-persistent) data connection rather than the control TCP connection.
RETR filename	Used to retrieve (that is, get) a file from the current directory of the remote host. This command causes the remote host to initiate a data connection and to send the requested file over the data connections
STOR filename	Used to store (that is, put) a file into the current directory of the remote host.

Figure 6.21: Some FTP commands.

- Each command is followed by a reply, sent form server to client.
- The replies are three-digit numbers, with an optional message following the number.

Response	Meaning
331	Username OK, password required
125	Data connection already open; transfer starting
425	Can't open data connection
452	Error writing file

Figure 6.22: Some FTP responses.

6.6.3 Data Connection

The data connection uses the well-known port 20 at the server site. However, the creation of a data connection is different from the control connection. The following shows the steps:

1. The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files.
2. Using the PORT command the client sends this port number to the server.
3. The server receives the port number and issues an active open using the well-known port 20 and the received ephemeral port number.

Communication over data connetion

- We want to transfer files through the data connection.

The client must define the *type of file to be transferred*, the *structure of the data*, and the t*ransmission mode*.

- Before sending the file through the data connection, we prepare for transmission through the control connection.

File Type

- FTP can transfer one of the following file types across the data connection:
  - *ASCII file*,
  - *EBCDIC file*, or
  - *image file*.

Data Structure

FTP can transfer a file across the data connection using one of the following interpretations of the structure of the data:

- *file structure*,
- *record structure*, or
- *page structure*.

Transmission Mode

- FTP can transfer a file across the data connection using one of the following three transmission modes:
  - *stream mode*,
  - *block mode*, or
  - *compressed mode*.
- The stream mode is the default mode; data are delivered from FTP to TCP as a continuous stream of bytes.

File Transfer

- File transfer occurs over the data connection under the control of the commands sent over the control connection.
- However, we should remember that file transfer in FTP means one of three things: *retrieving a file* (server to client), *storing a file (client to server), and directory listing (server to client).*

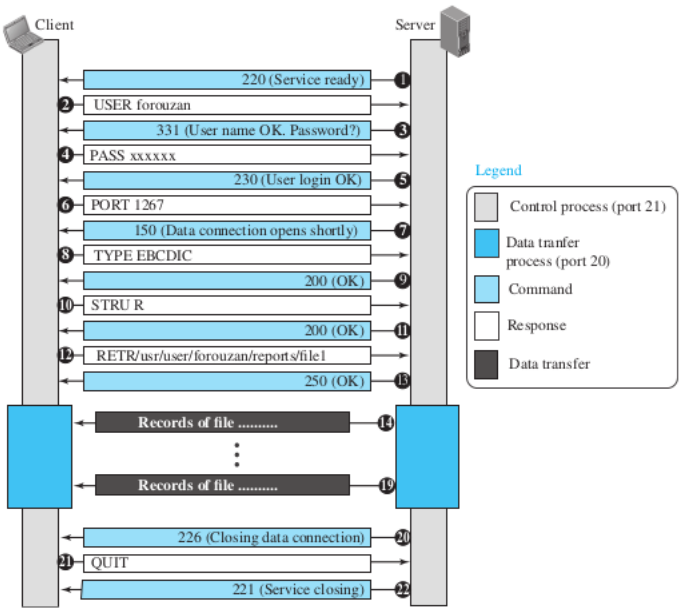


Figure 6.23: Example of FTP retriving a file.

Figure 6.23 shows an example of using FTP for retrieving a file. The figure shows only one file to be transferred. The control connection remains open all the time, but the data connection is opened and closed repeatedly. We assume the file is transferred in six sections. After all records have been transferred, the server control process announces that the file transfer is done. Since the client control process has no file to retrieve, it issues the *QUIT* command, which causes the service connection to be closed.

6.7 HTTP and FTP:Similarites and Differences

- Both are file transfer protocols; both run on top of TCP.
- FTP uses two parallel connection to transfer a file, a control connection and a data connection.
- Because FTP uses a separate control connection, FTP is said to send its control information *out-of-band*.
- HTTP sends requests and response header lines into the same TCP connection that carries the transferred file itself. For this reason, HTTP is said to sent its control information *in-band*.
- HTTP is *stateless* while FTP server must maintain state about the user.

6.8 Putty, WinSCP

- Putty and WinSCP is an open source file transfer application.
- Popular SSH and Telnet client.
- Its main function is the secure file transfer between a local and a remote computer.

- For downloading and other information refer to:
`http://winscip.net`
`http://www.putty.org`

6.9 SFTP

- SFTP stands for **SSH File Transfer Protocol**. And, SSH stands for Secure Shell.
- SSH is a secure application program that can be used in several purposes such as remote logging and file transfer.
- SFTP is secure file transfer protocol that runs over the SSH protocol. If supports the full security and authentication functionality of SSH.

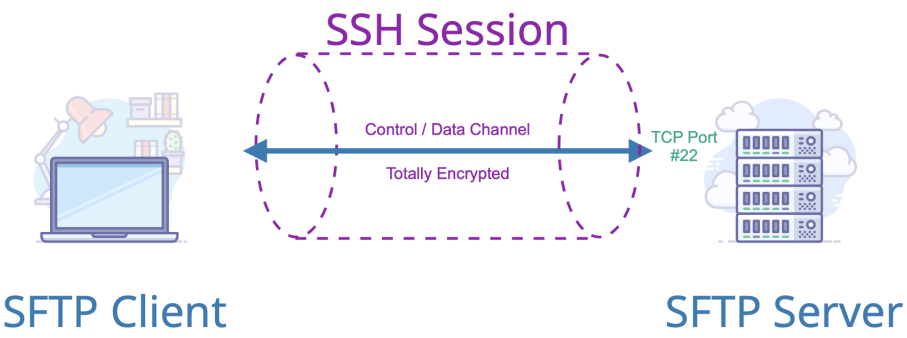


Figure 6.24: SFTP

6.10 Electronics Mail (E-mail)

- Three major components:
  i. user agents
  ii. mail servers
  iii. imple mail transfer protocol: SMTP

- User Agent
  - a.k.a. “mail reader”

- Allows user to read, reply to, forward, save and compose messages.
- e.g., Microsoft Outlook, Thunderbird, iPhone mail client outgoing, incoming messages stored on mail server.

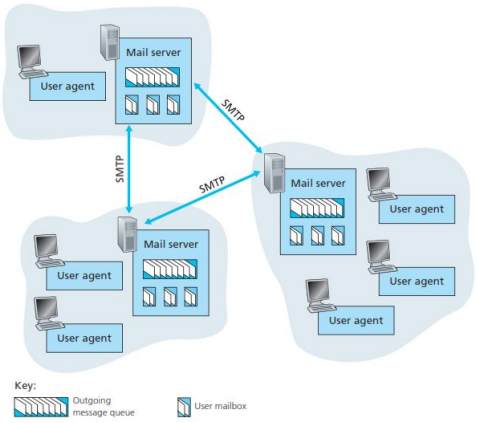


Figure 6.25: A high level view of the internet e-mail system.

Services of Agent

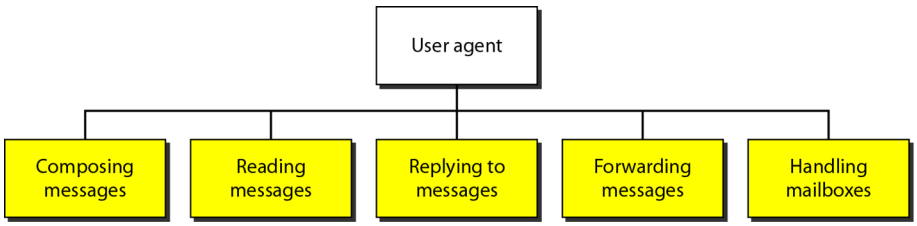


Figure 6.26: Types of services of user agent.

Mail Server
- Forms the core of the e-mail infrastructure.
- Outgoing, incoming mails are stored in mail servers.

6.11 SMPT

- Simple Mail Transport Protocol (SMTP) is the principal application-layer protocol for Internet electronic mail.
- It uses the reliable data transfer service of TCP to transfer mail from the sender's mail server to the recipient's mail server using port 25.
- As with most application-layer protocols, SMTP has two sides: a client side, which executes on the sender's mail server, and 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 sends mail to other mail servers, it acts as an SMTP client. When a mail server receives mail from other mail servers, it acts as an SMTP server.

- Defined in RFC 5321
- SMTP is at the heart of Internet Electronic mail
- SMTP transfers messages from senders' mail servers to the recipients' mail server.
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction (like HTTP, FTP)
- commands: ASCII text
- response: status code and phrase

- messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

To illustrate the basic operation of SMTP, let's suppose Alice wants to send Bob a simple ASCII message:

1. Alice uses UA to compose message "to" bob@someschool.edu.
2. Alice's UA sends message to her mail server; message placed in message queue.
3. Client side of SMTP opens TCP connection with Bob's mail server.
4. SMTP client sends Alice's message over the TCP connection.
5. Bob's mail server places the message in Bob's mailbox.
6. Bob invokes his user agent to read message

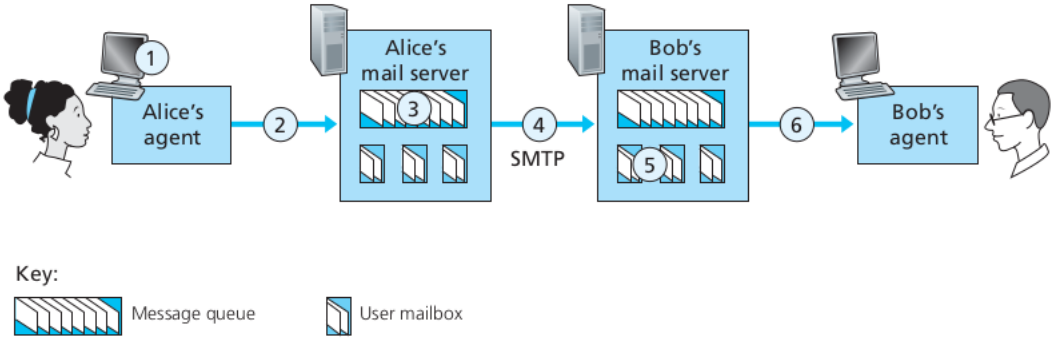


Figure 6.27: Alice sends message to Bob.

SMTP Connection and Transfer of Message

1. First, the client SMTP (running on the sending mail sever host) has TCP connection to port 25 at the server SMTP (running on the receiving mail sever host).
2. Once connection establish, perform application-layer handshaking operation between client and server.
   - During handshake SMTP client indicates the e-mail of the sender and the e-mail of the recipient.
3. The client sends the message using TCP service to the server.
4. The client then repeats this process over the same TCP connection if it has other messages to send to the server; otherwise, it instructs TCP to close the connection.

6.11.1 Comparison between SMTP and HTTP

- Both are file transfer protocol; HTTP transfers files (also called objects) from a web server to web client (typically a browser); SMTP transfers files (that is e-mail messages) from one mail server to another mail server.
- Both uses *persistent connection* for transfer.
- HTTP is a *pull protocol* and while SMTP is *push protocol*.
- SMTP required each message to be in 7-*bit ASCII* format; if message are not in ASCII, it has to be encoded into 7 bit ASCII. HTTP does not impose this restriction.
- HTTP: each object encapsulated in its own response message. In SMTP, multiple objects sent in multipart message.

6.11.2 Mail Access Protocol

- **SMTP**: delivery/storage to receiver's server.
- Mail access protocol: retrieval from server.
  - POP: Post Office Protocol [RFC 1939]: authorization, download - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server. - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

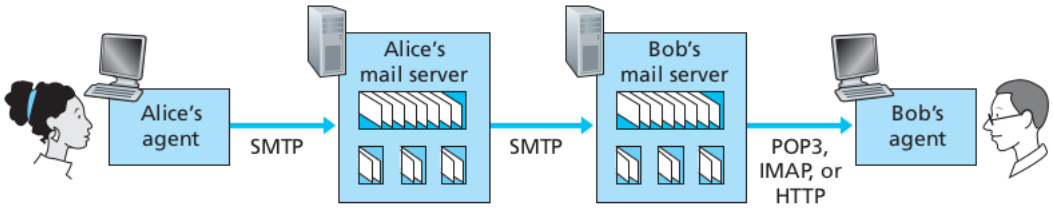


Figure 6.28: E-mail protocols and their communicating entities.

6.11.2.1 POP3

- Stands for **Post Office Protocol, version 3**, simple and limited in functionality.
- The client POP3 is installed on the recipient computer, the server POP3 software is installed on the mail server.

Process

- Mail access starts with the client when the user needs to download e-mail from the mail server to your local computer.
- The client opens a connection to the server on TCP port 110.
- It then sends its user name and password to access the mailbox.
- The user can then list and retrieve the mail message, one by one.

- Two mode of POP3

i. The delete mode
   - Mail is deleted from the mail box after each retrieval.

ii. The keep mode
   - The mail remains in the mailbox after retrieval.

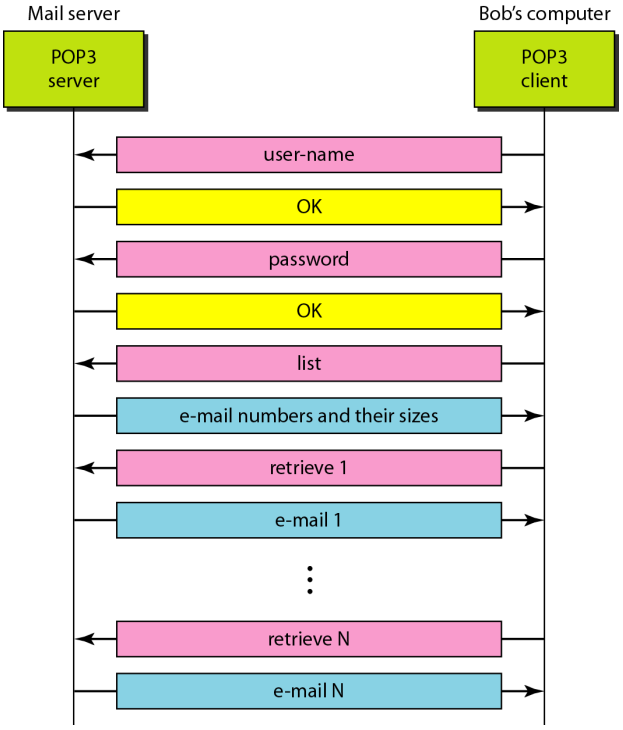


Figure 6.29: The exchange of commands and responses in POP3.

Drawbacks of POP3

- It does not allow the user to organize her mail on the server; the user cannot have different folders on the server.
- It does not allow the user to partially check the contents of the mail before downloading.

6.11.2.2 IMAP4

- Stands for **Internet Mail Access Protocol, version 4**.
- Similar to POP3, but it has more features than POP3; more powerful and complex.

IMAP4 provides the following extra functions:
- A user can check their mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.

- To use IMAP, the mail server runs an IMAP server that listens to port 143.
- The user agent runs an IMAP client.
- IMAP is defined in RFC 3501.

6.11.2.3 MIME

- MIME stands for **Multipurpose Internet Mail Extension**.
- Electronic mail has a simple structure. Its simplicity, however, comes with a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data.
- Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet. The message at the receiving site is transformed back to the original data.
- We can think of MIME as a set of software functions that transforms non-ASCII data to ASCII data and vice versa, as shown in Figure 6.30.

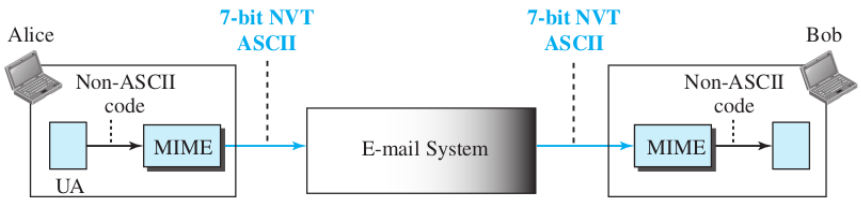


Figure 6.30: MIME.

6.12 Domain Name Server (DNS)

- DNS translates hostnames to IP address.
- The DNS is:
  - A distributed database implemented in a hierarchy of DNS servers, and
  - An application-layer protocol that allows a hosts to query the distributed database.
- The DNS protocol runs over UDP and uses *port 53*.

- DNS is commonly employed by other application-layer protocols- including HTTP, SMTP, and FTP—to translate user-supplied hostnames to IP addresses.

Example:
Consider what happens when a browser (that is, an HTTP client), running on some user's host, request the URL www.someschool.edu/index.html. In order for the user's host to be able to send an HTTP request message to the Web.someshcool.edu, 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 DNS client sends a *DNS query* within UDP datagram to port 53 containing the hostname to a DNS Server.
4. The DNS client eventually receives a *DNS reply* after a certain delay, which include the IP address for the hostname.
5. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP sever process located at port 80 at that IP Address.

6.12.1 Problem with Centerilzed Design

- **A single point of failure**
  If the DNS server crashes, do does the entire Internet.
- **Traffic Volume**
  A single DNS server would have to handle all DNS queries (for all the HTTP requests and e-mail messages generated from hundreds of millions of hosts.)
- **Distant Centralized Database**
  A single DNS server cannot be "close to" all the querying clients. If we put the single DNS server in New York City, then all queries from Australia must travel to the other side of the globe, perhaps over slow and congested links. This lead to significant delays.
- **Maintenance**
  The single DNS server is a single DNS server have to keep records for all Internet hosts. It is huge and would have to be updated frequently to account for every new host.

  In summary, *a centralized database in a single DNS server simply doesn't scale. Consequently, the DNS is distributed by design*.

6.12.2 A Distributed, Hierarchical Database

- In order to deal with the issue of scale, the DNS uses a large number of servers, organized in a hierarchical fashion and distributed around the world.
- No single DNS server has all of the mappings for all of the hosts in the Internet.
- Instead, the mappings are distributed across the DNS servers.
- There are three classes of DNS servers i. root DNS servers, ii. top-level domain (TLD) DNS servers, and iii. authoritative DNS servers organized in a hierarchy as shown in Figure 6.31.

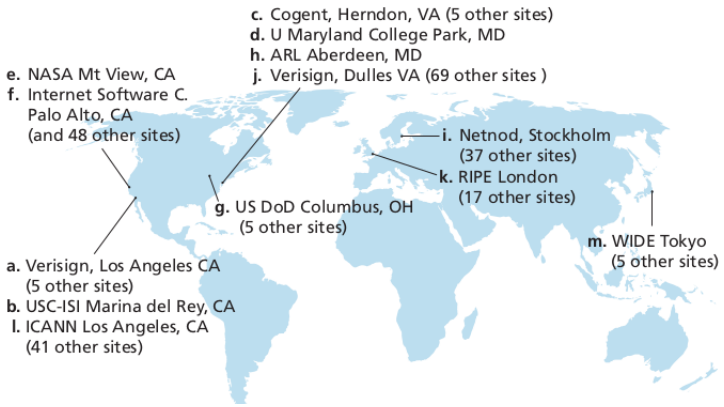


Figure 6.31: Portion of the hierarchy of DNS servers.

Root DNS Server

- In the Internet there are 13 root DNS servers, labeled A through M.
- There is a network of replicated servers, for both security and reliability purposes.
- Root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping

- returns mapping to local name server

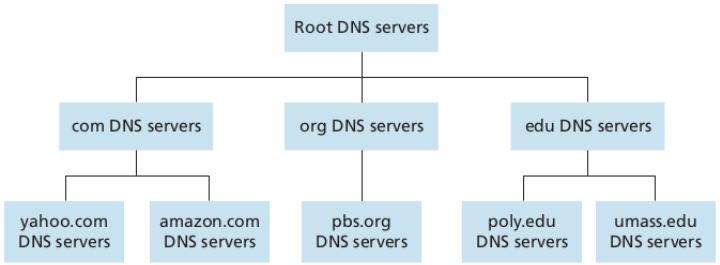


Figure 6.32: DNS root servers in 2012 (name, organization, location).

Top-level Domain (TLD) servers
- Responsible for com, org, net, edu, and gov, and all top-level country domains, e.g.: uk, fr, ca, jp.
- For example: the company Verisign Global Registry Services maintains the TLD servers for the com top-level domain, and the company Educause maintains the TLD servers for the edu top-level domain.

Authoritative DNS Servers
- Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts.
- can be maintained by organization or service provider. (e.g. web and mail)

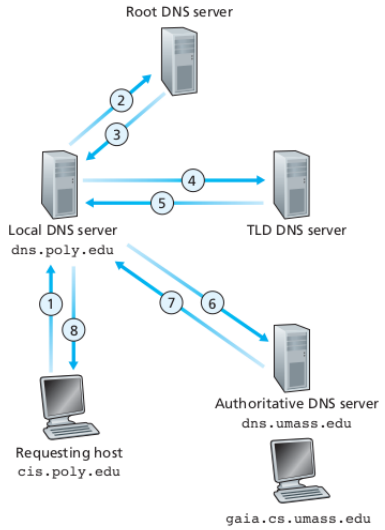
6.12.3 Local DNS
- It does not strictly belong to hierarchy.
- Each ISP (residential ISP, company, university) has a local DNS server.
    - Also called "default name server".
- When a host makes a DSN query,
    - query is sent to its local DSN server, which acts as a proxy, forwarding the query into the DNS server hierarchy.

6.12.4 Resolution
- Mapping a name to an address or an address to a name is called name-address *resolution*.
- A host that needs to map an address to a name or a name to an address calls a DNS client called a *resolver*.
- The resolver accesses the closest DNS server with a mapping request. - If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

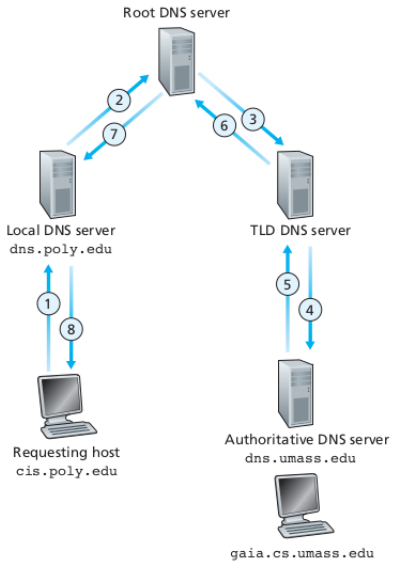
Resolution Types
i. Recursive Resolution
ii. Iterative Resolution
iii. Caching

6.12.5 DNS name resolution example
Suppose the host *cis.poly.edu* desires the IP address of *gaia.cs.umass.edu*. Also suppose that Polytechnic's local DNS server is called *dns.poly.edu* and that an authoritative DNS server for *gaia.cs.umass.edu* is called dns.umass.edu.

1. Figure 6.33a shows the *query* that is first sent to the local name server.
2. The local DNS server forwards the *query message* to a root DNS server.
3. The root DNS server takes note of the edu suffix and returns to the local DNS server a list of IP addresses for TLD servers responsible for edu.



(a) Interaction of the various DNS servers.



(b) Recursive queries in DNS.

Figure 6.33: DNS Queries

4. The local DNS server then resends the query message to one of these TLD servers.
5. The TLD server takes note of the *umass.edu* suffix and *responds* with the IP address of the authoritative DNS server.
6. Finally, the local DNS server resends the query message directly to *dns.umass.edu* responds with the IP address of *gaia.cs.umass.edu*.

- The example shown in Figure 6.33a makes use of both *recursive queries* and *iterative queries*. The query sent from *cis.poly.edu* to *dns.poly.edu* is a recursive query, since the query asks *dns.poly.edu* to obtain the mapping on its behalf.
- But the subsequent three queries are iterative since all of the replies are directly returned to *dns.poly.edu*.
- In theory, any DNS query can be iterative or recursive.
- For example, Figure 6.33b shows a DNS query chain for which all of the queries are recursive.
- **In practice**, the queries typically follow the pattern in Figure 6.33a: The query from the requesting host to the local DNS server is recursive, and the remaining queries are iterative.

6.12.6 DNS Caching

- A critically important feature of the DNS system.
- Improve the delay performance and to reduce the number of DNS messages ricocheting around the Internet.

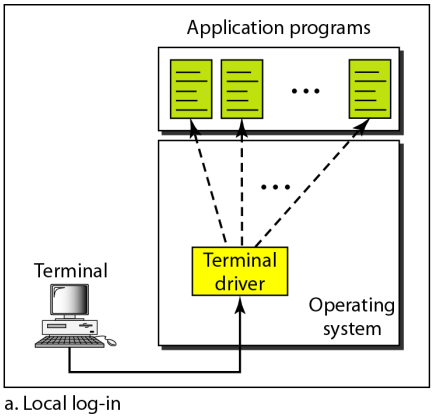
- Once (any) name server learns mapping, it caches mapping in local memory.
  -Cache entries timeout (disappear) after some time (TTL).
  - TLD servers typically cached in local name servers,
    -thus root name servers not often visited.
- Cached entries may be out-of-date (best effort name-to-address translation!)
  - if name host changes IP address, may not be known
  Internet-wide until all TTLs expire
- update/notify mechanisms proposed IETF standard
  - RFC 2136.

Example:
As an example, if Figure 6.33a, suppose that a host *apricot.poly.edu* queries dns.poly.edu for the IP address for the hostname *cnn.com*. Furthermore, suppose that a few hours later, another Polytechnic University host, say, *kiwi.poly.fr*, also queries dns.poly.edu with the same hostname. Because of caching, the local DNS server will be able to immediately return the IP address of *cnn.com* to this second requesting host without having to query

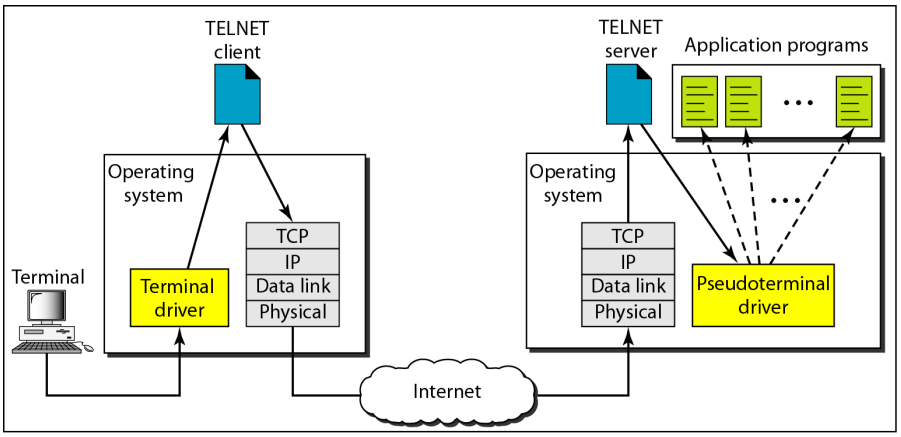
any other DNS servers. A local DNS server can also cache the IP addresses of TLD servers, thereby allowing the local DNS server to bypass the root DNS servers in a query chain (this often happens).

6.13 TELNET

- Abbreviation for TELecommunication NETwork.
- It is a client/server application program that allows a user to log on to a remote machine, giving the user access to the remote system.
- TELNET uses the **network virtual terminal** (NVT) system to encode the characters on the local system. On the server machine, NVT decodes the characters to a form acceptable to the remote machine.
- NVT uses the set of characters for data and a set of characters for control.
- TELNET uses only one TCP connection. The server uses the well known port 23 and client uses the an ephemeral port.
- The same connection is used to send data and control characters. Control characters are embedded in the data stream.



a. Local log-in



b. Remote log-in

Figure 6.34: Local and remote log in.

Local Log In

- When a user logs into a local timesharing system, it is called local log-in.
- When a user types at a terminal, the keystrokes are accepted by the terminal driver.
- The terminal driver passes the characters to the operating system.
- The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

Remote Log In

- When user access an application program or utility located on a remote machine, it is called remote log-in, in which telnet client and server comes in use.
- The users sends the keystrokes to the terminal driver, when the local operating system accepts the characters but does not interpret them. The characters are send to the TELNET client, which transforms the characters to

a universal character set called network virtual terminal (NVT) characters and delivers them to the local TCP/IP protocol stack.
- The commands or text, in NVT form, travel through the internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer.
- However, the character cannot directly passed to OS because remote OS are not designed to receive the characters from a TELNET server. The software called pseudo terminal driver is used which pretends that the characters are coming from the a terminal. The OS then passed the characters to the appropriate application.

Network Virutal Terminal
- TELNET uses the network virtual terminal (NVT) system to encode the characters on the local system. On the server machine, NVT decodes the characters to a form acceptable to the remote machine.
- NVT uses the set of characters for data and a set of characters for control.
- Control characters are embedded in the data stream and preceded by the interpret as control (IAC) control character.

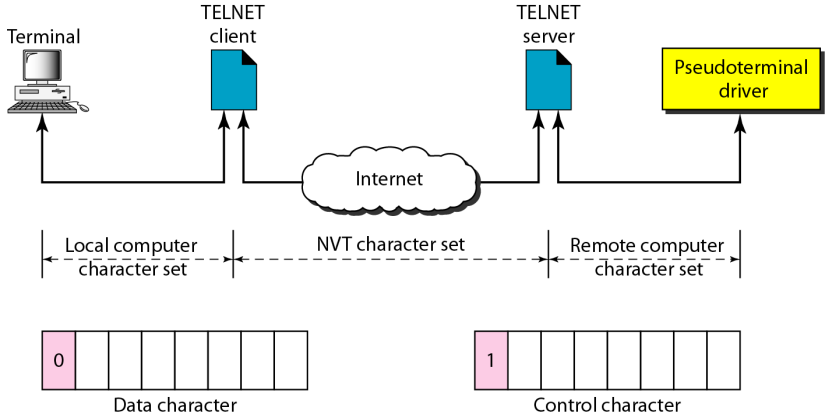


Figure 6.35: Network virtual terminal.

6.14 Peer-to-Peer Architecture

- Client-server architecture
  - Web, e-mail, DNS

- P2P architecture
  - Hosts, called peers, directly communicate with each other.
  - Hosts (e.g. desktops, laptops etc.) controlled by users.

- P2P applications
  - P2P File Distribution
  - Distributed Hash Table (DHTs)

6.14.1 P2P File Distribution: Bit Torrent

- Originally developed by Bram Cohen.
- One of the most popular P2P file distribution protocol.

- In bit torrent, the collection of all peer participating in the distribution of a particular file is called a *torrent*.
- Peers in a torrent download equal size chunks (typically 256 Kb) of the file from one another.
- When a peer first joins a torrent, it has no chunks. Over time it accumulates more and more chunks. While it downloads chunks it also uploads chunks to other peers. Once a peer has acquired the entire file, it may leave the torrent, or remain in the torrent and continue to upload chunks to other peers.

- Each torrent has an infrastructure node called *tracker*.
- When a peer joins a torrent, it registers itself with the tracker and periodically informs the trackers that it is still in the torrent.
- In this manner, the tracker keeps track of peers that are participating in the torrent.

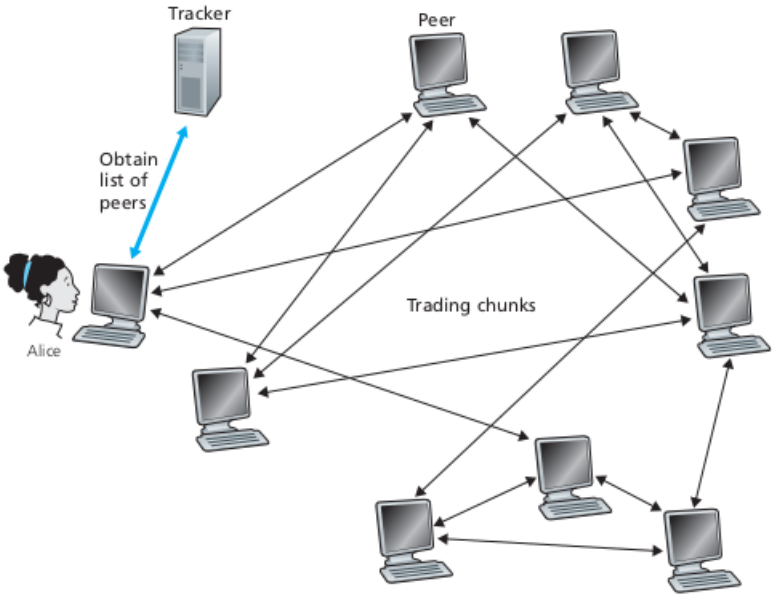


Figure 6.36: File distribution with Bit Torrent.

6.14.2 Bit Torrent: requesting, sending file chunks

- As shown in Figure 6.36, when a new peer, Alice, joins the torrent, the tracker randomly selects a subset of peers (for concreteness, say 50) from the set of participating peers, and sends the IP addresses of these 50 peers to Alice.
- Possessing this list of peers, Alice attempts to establish concurrent TCP connections with all the peers on this list. Let's call all the peers with which Alice succeeds in establishing a TCP connection "neighboring peers." (In Figure 6.36, Alice is shown to have only three neighboring peers. Normally, she would have many more.)
- As time evolves, some of these peers may leave and other peers (outside the initial 50) may attempt to establish TCP connections with Alice. So a peer's neighboring peers will fluctuate over time.

- At any given time, each peer will have a subset of chunks from the file, with different peers having different subsets. Periodically, Alice will ask each of her neighboring peers (over the TCP connections) for the list of the chunks they have. If Alice has **L** different neighbors, she will obtain **L** lists of chunks. With this knowledge, Alice will issue requests (again over the TCP connections) for chunks she currently does not have.
- Alice uses a technique called **rarest first** to request the chunks of file.

- To determine which requests she responds to, BitTorrent uses a clever **trading algorithm**. The basic idea is that Alice gives priority to the neighbors that are currently supplying her data at the highest rate. Specifically, for each of her neighbors, Alice continually measures the rate at which she receives bits and determines the four peers that are feeding her bits at the highest rate. She then reciprocates by sending chunks to these same four peers. Every 10 seconds, she recalculates the rates and pos- sibly modifies the set of four peers. In BitTorrent lingo, these four peers are said to be unchoked. Importantly, every 30 seconds, she also picks one additional neighbor at random and sends it chunks. Let's call the randomly chosen peer Bob. In BitTorrent lingo, Bob is said to be optimistically unchoked. Because Alice is sending data to Bob, she may become one of Bob's top four uploaders, in which case Bob would start to send data to Alice. If the rate at which Bob sends data to Alice is high enough, Bob could then, in turn, become one of Alice's top four uploaders. In other words, every 30 seconds, Alice will randomly choose a new trading partner and initiate trad- ing with that partner. All other neighboring peers besides these five peers (four "top" peers and one probing peer) are "choked," that is, they do not receive any chunks from Alice.

- BitTorrent has a number of interesting mechanisms that are not discussed here, including pieces (mini-chunks), pipelining, random first selection, endgame mode, and anti-snubbing.

6.14.3 Distributed Hash Tables (DHTs)

- A DHT is a simple database in a P2P network.
- A centralized version of this simple database, which will simply contain (key, value) pairs. For example, the keys could be social security numbers and the values could be the corresponding human names; in this case, an example key-value pair is (156-45-7081, Johnny Wu).

- When query with key(s), return the value corresponding to the key(s).
- A client-server architecture stores all the (key, value) pairs in one central server.
- P2P system store the (key, value) pairs over millions of peers. Such distributed database is referred to as a **distributed hash table** (DHT).
- A specific example DHT service in the context of P2P file sharing, a key is the content name and the value is the IP address of a peer that has a copy of the content.

- One appraoch of building a DHT is, each peer is assigned an identifier in the range $[0, 2^n - 1]$, for some value of n.
Each key is also assigned a integer in the range $[0, 2^n - 1]$ using hash function.
- A hash function is a many-to-one function for which two different inputs can have the same output (same integer), but the likelihood of the having the same output is extremely small.
- Given that each peer has an integer identifier and that each key is also an integer in the same range, a natural approach is to assign each (key, value) pair to the peer whose identifier is the *closest* to the key.
-For example: Suppose n = 4 so that all the peer and key identifiers are in the range [0, 15]. Further suppose that there are eight peers in the system with identifiers 1, 3, 4, 5, 8, 10, 12, and 15. Finally, suppose we want to store the (key, value) pair (11, Johnny Wu) in one of the eight peers. But in which peer? Using our closest convention, since peer 12 is the closest successor for key 11, we therefore store the pair (11, Johnny Wu) in the peer 12.

- To insert (key, value) in the system ,it requires each peer to keep track of all other peers in the DHT—which is completely impractical for a large-scale system with millions of peers.

6.14.3.1 Circular DHT

In this circular arrangement, each peer only keeps track of its immediate successor and immediate predecessor (modulo 2 n). An example of such a circle is shown in Figure 6.37(a). In this example, n is again 4 and there are the same eight peers from the previous example. Each peer is only aware of its immediate successor and predecessor; for example, peer 5 knows the IP address and identifier for peers 8 and 4 but does not necessarily know anything about any other peers that may be in the DHT.
- This circular arrangement of the peers is a special case of an **overlay network**.

- In an overlay network, the peers form an abstract logical network which resides above the “underlay” computer network consisting of physical links, routers, and hosts.
- The links in an overlay network are not physical links, but are simply virtual liaisons between pairs of peers.

- Using the circular overlay in Figure 6.37(a), now suppose that peer 3 wants to determine which peer in the DHT is responsible for key 11. Using the circular overlay, the origin peer (peer 3) creates a message saying “Who is responsible for key 11?” and sends this message clockwise around the circle. Whenever a peer receives such a message, because it knows the identifier of its successor and predecessor, it can determine whether it is responsible for (that is, closest to) the key in question.
- If a peer is not responsible for the key, it simply sends the message to its successor.

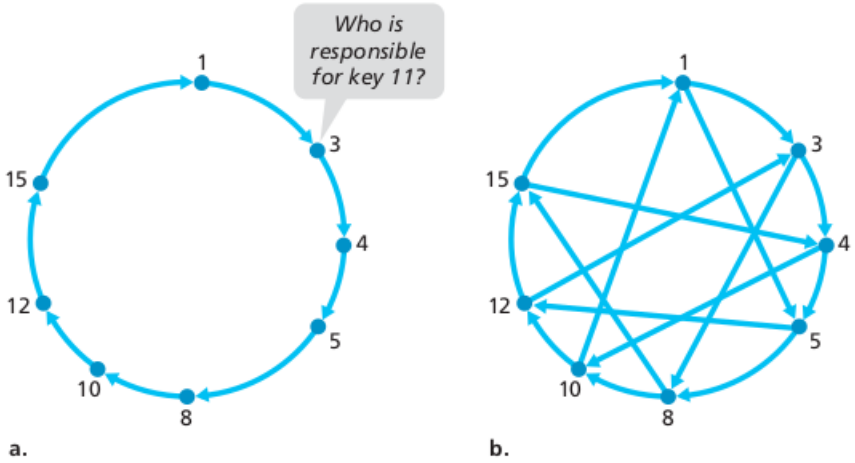


Figure 6.37: (a) A circular DHT. Peer 3 wants to determine who is responsible for key 11. (b) A circular DHT with shortcuts.

6.15 Network Traffic Analysis and Monitoring

- Process of capturing, decoding, and analyzing network traffic
  - Why is the network slow?
  - What is the network traffic pattern?
  - How is the traffic being shared between nodes?

- Network administrator are constantly striving to maintain smooth operation of their networks.
  - Need to monitor traffic movement,
  - Need to monitor performance throughput the network, and
  - Verify that security breeches do not occur within the network.

6.15.1 Network Analyzer

- A combination of hardware and software tools that can detect, decode, and manipulate traffic on the network.
  - Passive monitoring (detection) – difficult to detect
  - Active (attack)

- Available both free and commercially
- Mainly software-based.
- Also known as sniffer.
  - A program that monitors that data traveling through the network passively.

Who uses network analyzer?

- System administrators
  - Understand system problems and performance

- Malicious individuals (intruders)
  - Capture clear text data
  - Passively collect data on vulnerable protocols
    - ◇ FTP, POP3, IMAP, SMTP, HTTP, etc.
    - ◇ Capture VoIP data
  - Traffic pattern discovery
  - Actively break into the network (backdoor techniques)

6.16 Network Management

- Network management is defined as monitoring, testing, configuring, and troubleshooting network components to meet a set of requirements defined by an organization.
- These requirements include the smooth, efficient operation of the network that provides the predefined quality of service for users. To accomplish this task, a network management system uses hardware, software and humans.

- The International organization for Standardization defines five areas of network management.

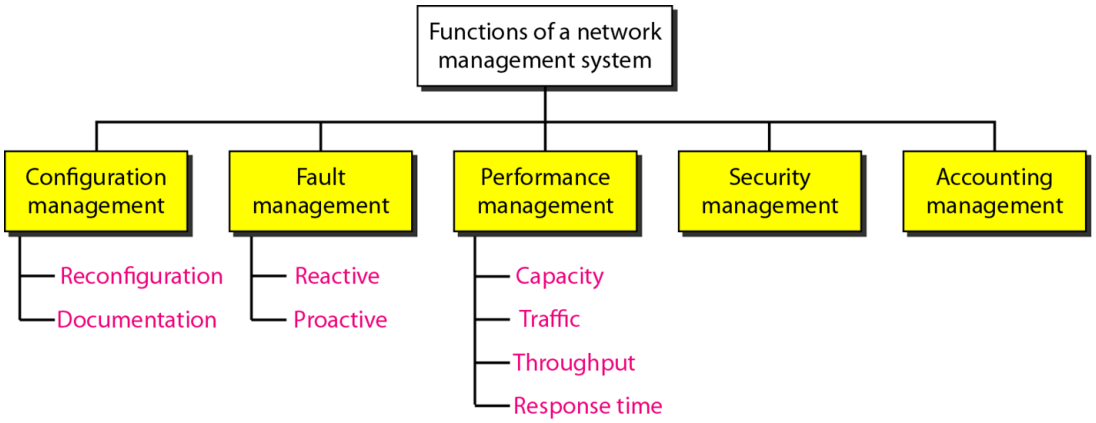


Figure 6.38: Areas of network management.

6.16.1 SNMP

- The Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using the TCP/IP protocol suite.
- It provides a set of fundamental operations for monitoring and maintaining an internet.
- SNMP uses the concept of manager and agent. A manager is usually a host, that controls and monitors a set of agents, usually routers or servers.
- It is an application layer protocol that monitor devices made by different manufactures and installed on different networks. That is, it frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology.
- It can be used in heterogeneous internet.

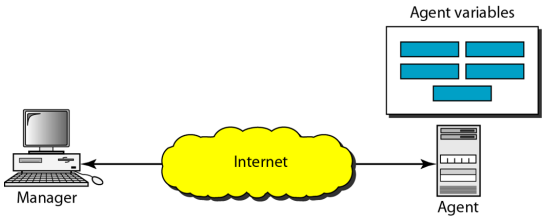


Figure 6.39: SNMP concept.

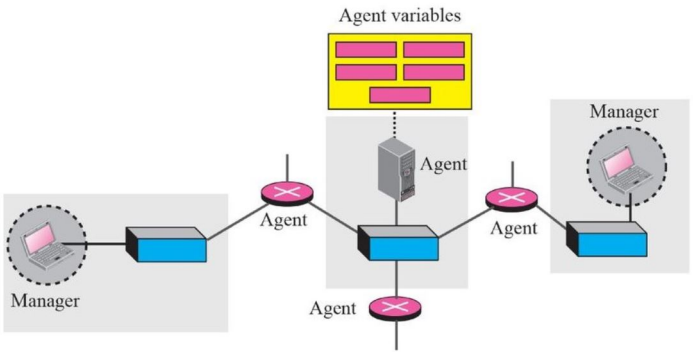


Figure 6.40: SNMP concept.

Mangers and Agents
- A management station, called a manger, is a host the run the SNMP client program. A managed station, called an agent, is a router(or a host) that runs the SNMP server program. Management is achieved through simple interaction between manager and agent.

- The agent keeps the performance information in a database. The manger has access to the values in the database. For example, a router can store in appropriate variable the number of packets received and forwarded. The manger can fetch and compare the values of these two variable to see if the router is congested or not.
- The manger can also make the router perform certain actions. For example, force router to reboot.
- Agents can contribute the management process by sending a warning message (trap) to the manger if it notices something unusual.

- In other words, management with SNMP is based on three basic ideas:

1. A manger checks an agent by requesting information that reflects the behavior of the agent.
2. A manger forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by warning the manger of an unusual situation.

- SNMP has the ability to help you manage your network by:

- Provide read/write abilities: for example you could use it to reset password remotely, or reconfigure IP addresses.
- Collect information on how much bandwidth is being used.
- Collect error reports into a log, useful for troubleshooting and identifying trends.
- Email an alert when your server is low on disk space. Monitor your servers' CPU an Memory use, alert when threshold are exceeded.
- Send SMS-text message when a device fails.

Components of network management on the Internet
Management on internet is done through the cooperation of three protocols: SNMP, SMI (Structure of Management Information) and MIB (Management Information Base).

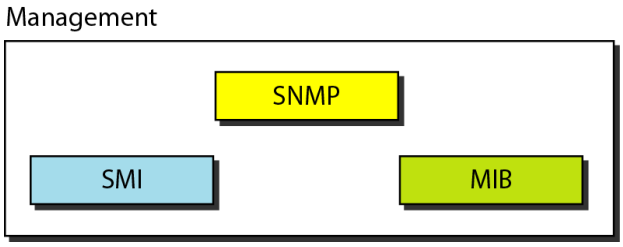


Figure 6.41: Components of network management on the internet.

Roles of SNMP
- Defines the format of the packet to be sent from manager to an agent and vice-versa.
- Interprets the results and creates statistics.
- Responsible for reading and changing the object values (status).

Role of SMI
- To use SNMP, we need rules for naming objects.
- SMI defines the general rules for naming objects, defining object types (including range and length), and showing how to encode objects and values.
- SMI is a protocol that defines these universal rules.
- However, SMI does not define the number of objects an entity should manage or name the objects to be managed or define the association between the objects and their values.

Roles of MBI
- For the entity to be managed, MBI define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object.
- This protocol creates a set of objects defined for each entity in a manner similar to that of a database.

6.16.2 MRTG
- Most common network traffic measurement tool.
- MRTG is a tool to monitor the traffic load on network-links.
- Monitors bits in and out of a network device (e.g. switch port, router port, NIC Card).
- MRTG uses simple SNMP queries on a regular interval to generate graphs.
- MRTG generates HTML pages containing PNG images which provide an almost live visual representation of this traffic.
- MRTG measures bandwidth.
- MRTG software can be used not only to measure network traffic on interfaces, but also build graphs of anything that has an equivalent SNMP MIB –like CPU load, Disk availability, temperature etc..

6.16.3 PRTG (Paessler Router Traffic Grapher)
- An easy to use Windows-based software for monitoring network and bandwidth usage as well as various other network parameters like memory and CP utilization.
- Provides system administrators with live readings and periodical usage trends of leased lines, routers, firewalls, servers, and many other network devices.

Features:
- Supports data acquisition via SNMP, packet sniffing, or Netflow
- Classifies network traffic by IP address, protocol, and other parameters.
- Easy installation and use on Windows 2000/XP/2003.
- Capable of monitoring up to several thousands sensors.
  - `https://www.paessler.com/prtg`

6.16.4 Wireshark
- Supports commands-line and GUI interfaces.
  - TSHARK offers command line interface.
- For detail information and to download the program refer to:
  - `www.wireshark.org`