DOMAIN NAME SYSTEM (DNS) :

What is DNS?

- The Domain Name System (DNS) is the phonebook of the Internet.
- Humans access information online through domain names, like nytimes.com or espn.com.
- Web browsers interact through Internet Protocol (IP) addresses.
- DNS translates domain names to IP addresses so browsers can load Internet resources.
- Each device connected to the Internet has a unique IP address which other machines use to find the device.
- DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1.

Goal of DNS Assign meaningful high level names to a large set of machines and handles the mapping of those names to a machine IP address.

- **DNS is distributed database system that resides on multiple machines on the internet and used to convert between names and addresses and provide email routing information.**
- **DNS provides client and server to communicate each other.**
- DNS is a **case sensitive** i.e **com or COM** means same thing.
- **Full domain name is a sequence of labels separated with dot (.)**
- **DNS name space is a hierarchical and it is** similar to unix file sytem.

➤ DNS NAME SPACE :

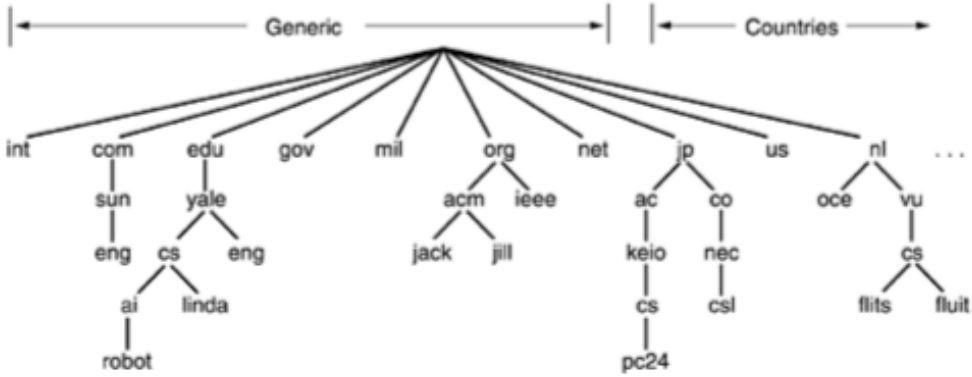
- DNS 2 types : **"Flat Name spaces" and "Hierarchical Name spaces".**
- The Name assigned to a machine carefully selected from a name space with complete control over the binding b/w the Names and IP Addresses.

- **Flat Name spaces :**
  - The original set of machines on the internet used flat name spaces.
  - These name spaces consisted of Sequence of characters with no further structure.
  - A Name is Assigned to address.
  - **Names are Convenient and Short.**
- **Hierarchical Name spaces:**
  - Hierarchical Name spaces provides a simple and flexible naming structure.
  - The Name space is portioned at the TOP level.

- Authority for names in each part ion are passed to each designated agent.
- The Names are designed in an inverted –tree structure with the root at the top.
- The tree can have only 128 levels.

The tree is divided into three domains: generic, country and reverse as shown in fig.

Figure 7-1. A portion of the Internet domain name space.

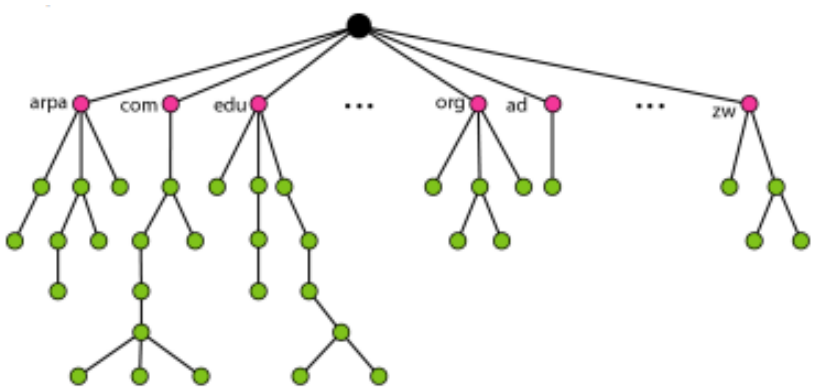


➤ DOMAIN NAME SYSTEM :

- Every domain, whether it is a single host or a top-level domain, can have a set of resource records associated with it.
- For a single host, the most common resource record is just its IP address.
- But many other kinds of resource records also exist.
- When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name. Thus, the primary function of DNS is to map domain names onto resource records.

A resource record is a five-tuple. Although they are encoded in binary for efficiency, in most expositions, resource records are presented as ASCII text, one line per resource record. The format we will use is as follows:

To have a hierarchical name space, a domain name space was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.

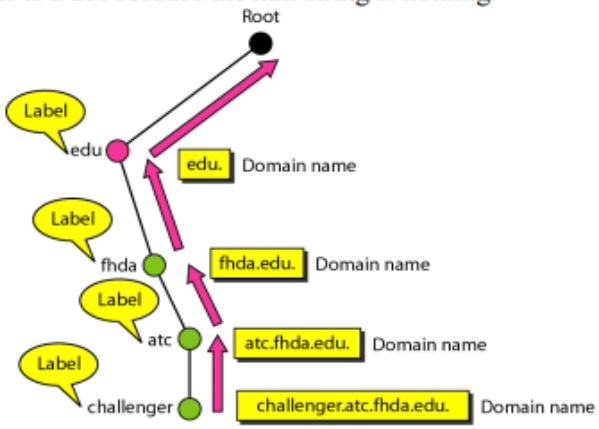


Label

Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

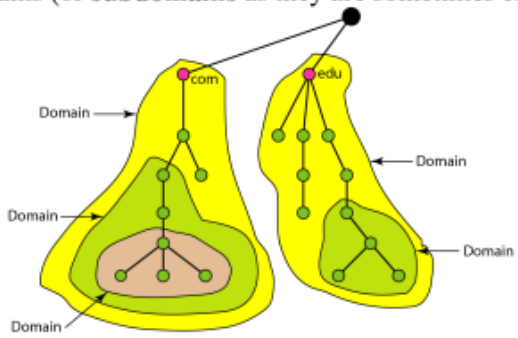
Domain Name

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.



Domain

A **domain** is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Figure 25.5 shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes called).



The first level of the generic domain convention allows seven possible three character labels describing organization type.

1. Com. commercial organization.

2. edu.: educational institution .

3. gov.: government institution.

4. int.: international organization.

5. mil.: military group.

6. net.: Network support center.

7. org. organizations other than listed above.

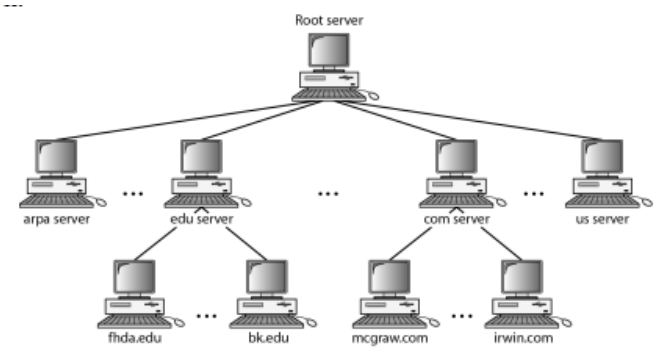
Each domain name corresponds to a particular IP address. To find the address, the resolution application begins searching with the first level. As a much is found, a pointer leads to the next level and finally to the associated IP address.

DISTRIBUTION OF NAME SPACE :

The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not unreliable because any failure makes the data inaccessible.

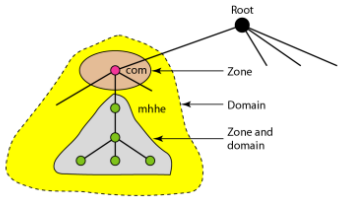
Hierarchy of Name Servers:

The solution to these problems is to distribute the information among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or a small domain.



Zone:

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a zone. We can define a zone as a contiguous part of the entire tree. If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the domain and the zone refer to the same thing. The server makes a database called a zone file and keeps all the information for every node under that domain.



Country Domain:

The country domain convention follows the same format as generic domain, but uses two character country abbreviation in place of three character organizational abbreviations at the first level shown in table. Second level labels can be organizational or they can be more specific national designations.

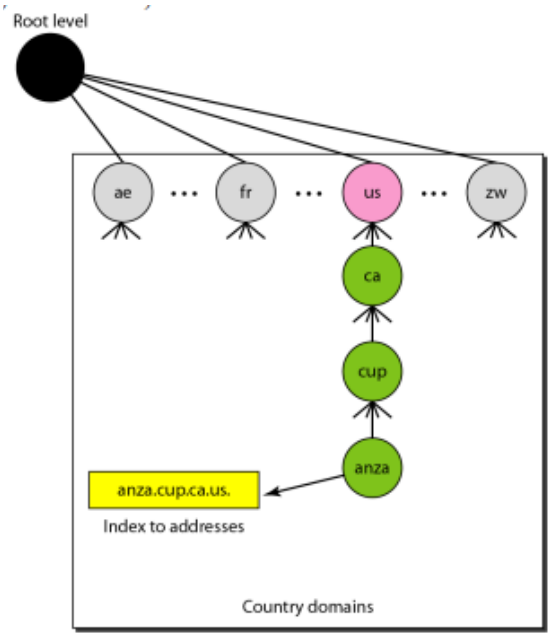


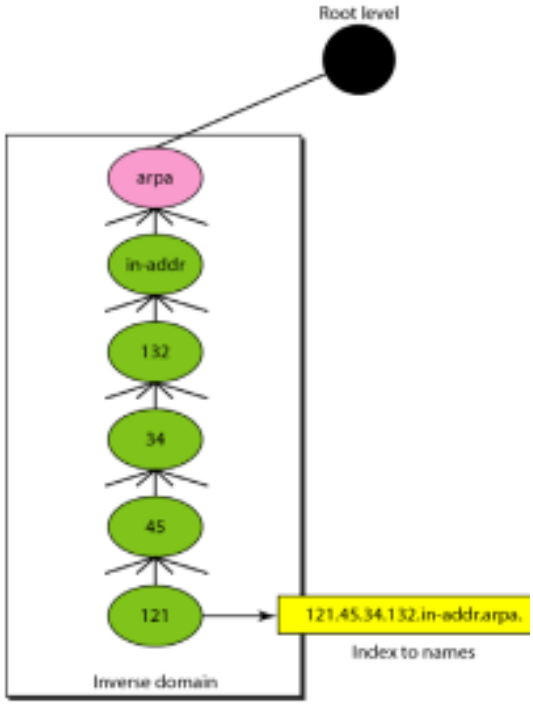
Table: SOME DOMAIN NAME SYSTEM COUNTRY CODE

Country Code	Country Name	Country Code	Country Name
AE	Arubeme rates	IN	India
AU	Australia	IT	Italy
BE	Belgium	JP	Japan
CA	Canada	KW	Kuwait
CH	Switzerland	NL	Netherlands
DE	Germany	NO	Norway
DK	Denmark	NZ	Newzeland
ES	Spain	SE	Sweden
FI	Finland	US	United States of America
GR	Greece		

Reverse Domain:

If we have the IP address and need the domain name, you can reverse domain the functions of DNS. The domain can be inserted onto the tree in two ways. For example ugc.control.edu could equally be listed under the country domain as cs.yale.ct.us.

To create a new domain, permission is required of the domain in which it will be included. For example, rgm group was started under aicte and is known as rgm.aicte.control.edu. It needs permission from which use manages aicte.control.edu. Naming follows organizational boundaries, not physical networks.



RESOURCE RECORDS :

Every domain in the DNS tree maintains a set of Resource Records, which are connected to it. For a leaf node i.e., single host, the most common resource record is its IP address. When a resolver gives a name to DNS, it gets back called as resource records associated with that name. The original function of a DNS is to map domain names on to the resource records. A resource record is a five tuple, in ASCII text they are represented as. The type of field tells what kind of record it is, some of the type records are listed in table 5.3.

S.No	Type	Meaning	Value
1.	SoA	Start of Authority	Parameter for this zone
2.	A	IP address of a host	32 bit integer
3.	Mx	Mail Exchange	Priority
4.	NS	Name Server	Name of the server for this domain
5.	CNAME	Canonical name	Domain Name
6.	PTR	Pointer	Alias for an IP address
7.	TXT	Text	Uninterpreted ASCII text

I. The SOA record provides name of the primary source of information about

(a) name servers zone

(b) e-mail address of its administration

(c) various flags and

(d) various time outs.

2. The record A, holds a 32 bit IP address of the host. If a host connects two or more networks, each case it has one type of a resource record per network connection.

3. The MX record specifies the name of domain prepared to accept e-mail for the specified domain. It allows the host that is not on the internet to receive e-mail from internet sites.

4. NS record specifies Name server.

5. CNAME record specifies allows the aliases to be created.

6. PTR is a regular DNS data type whose interpretation depends on the context on which it is found.

7. The TXT record allows domains to identify themselves in arbitrary way i.e., it is for user convenience.

NAME SERVERS:

- A single name server could contain the entire DNS database and respond to all queries about it.
- server would be so overloaded as to be useless. Furthermore, if it ever went down, the entire Internet would be crippled(severely damaged or malfunctioning.)
- To avoid the problems associated with having only a **single source of information, the DNS name space is divided into non overlapping zones.**
- One possible way to divide the name space of Fig. is shown in Fig. 7-4. Each zone contains some part of the tree and also contains name servers holding the information about that zone.
- Normally, a zone will have one primary name server, which gets its information from a file on its disk, and one or more secondary name servers, which get their information from the primary name server.
- To improve reliability, some servers for a zone can be located outside the zone.

Let us consider an example shown in fig connected with another domain. here a resolver on "ece.rgm.jntu.in" wants to know the IP address of the host "rgm.aicte.control.edu" can be explained in 8 steps.

Step 1: It sends a query to the local name server rgm.jntu.in.This query asks a record of type A and the class IN.

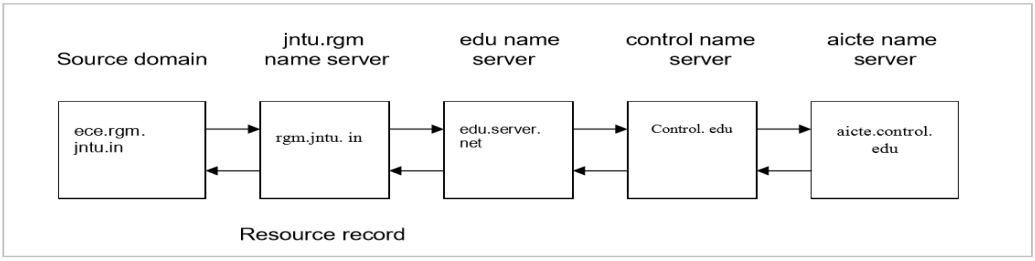
Step 2: If the local name server had no such domain and knows nothing about it, it may ask a few other near by name servers if none of them know, it sends a UDP packet to the server for "edu" given in its database (see fig) edu.server.net.

Step 3: It forwards the request to the name server control.edu.

Step 4: And in turn this forwards the request aicte.control.edu, which has authoritative resource records. This is the request from client to a server, the resource record requested will work its way back in

step 5 to step 8.Once these records get back to rgm.jntu.in name server, they will be entered into a cache/memory.

However this information is not authoritative, since changes made at aicte.control.edu will not be propagated to all the memories in the world. For this reason cache should not live too long, so time-to-live field is used in each resource record. It tells the name server how long to cache records



WORKING OF A RESOLVER FOR A DOMAIN IN 8 STEPS

World Wide Web (WWW):

- The World Wide Web is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet.
- WWW is a way of exchange information between computers over internet.
- WWW is a collection of files stored on thousands of servers all over the world.
- These files represent documents , pictures, videos, sounds, programs, interactive environments.
- www or **web** it consist of world wide collection of web documents.
- **Web site** is collection of web pages and associated items.
- **Web server** is a computer that delivers requested web pages to your computer.

The WWW today is a distributed client/server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called web sites.

Architectural Overview:

- Web consists of a vast, worldwide collection of documents or **Web pages**, often just called pages for short.
- Each page may contain links to other pages anywhere in the world.
- Users can follow a link by clicking on it, which then takes them to the page pointed to.
- This process can be repeated indefinitely.
- The idea of having one page point to another, now called **hypertext.**
- Pages are viewed with a program called a **browser**, of which Internet Explorer and Netscape Navigator are two

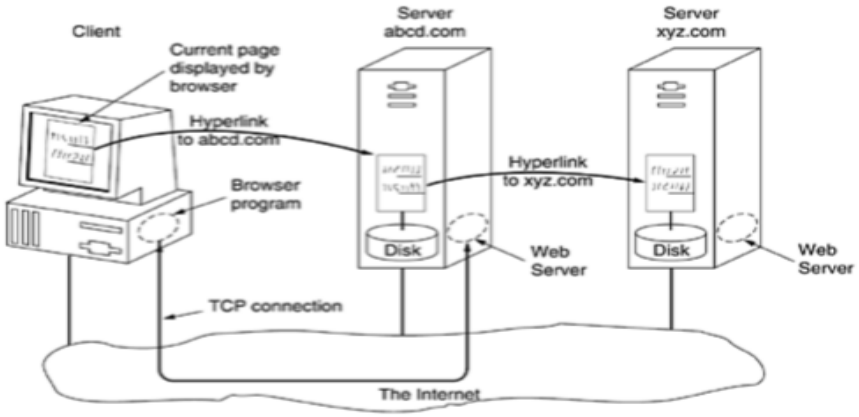
popular ones.

- many Web pages, this one starts with a title, contains some information, and ends with the e-mail address of the page's maintainer.

(a) A Web page. (b) The page reached by clicking on Department of Animal Psychology.

The basic model of how the Web works is shown in Fig. 7-19. Here the browser is displaying a Web page on the client machine. When the user clicks on a line of text that is linked to a page on the abcd.com server, the browser follows the hyperlink by sending a message to the abcd.com server asking it for the page. When the page arrives, it is displayed. If this page contains a hyperlink to a page on the xyz.com server that is clicked on, the browser then sends a request to that machine for the page, and so on indefinitely.

Figure 7-19. The parts of the Web model.



The Client Side :

Let us now examine the client side of Fig. 7-19 in more detail. In essence, a browser is a program that can display a Web page and catch mouse clicks to items on the displayed page. When an item is selected, the browser follows the hyperlink and fetches the page selected. Therefore, the embedded hyperlink needs a way to name any other page on the Web. Pages are named using URLs (Uniform Resource Locators). A typical URL is

http://www.abcd.com/products.html

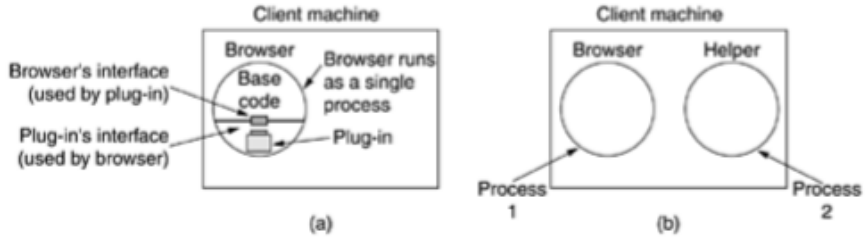
When a user clicks on a hyperlink, the browser carries out a series of steps in order to fetch the page pointed to. Suppose that a user is browsing the Web and finds a link on Internet telephony that points to ITU's home page, which is http://www.itu.org/home/index.html. Let us trace the steps that occur when this link is selected.

1. The browser determines the URL (by seeing what was selected).
2. The browser asks DNS for the IP address of www.itu.org.
3. DNS replies with 156.106.192.32.
4. The browser makes a TCP connection to port 80 on 156.106.192.32.
5. It then sends over a request asking for file /home/index.html.
6. The www.itu.org server sends the file /home/index.html.
7. The TCP connection is released.
8. The browser displays all the text in /home/index.html.
9. The browser fetches and displays all images in this file.

There are two possibilities: plug-ins and helper applications. A plug-in is a code module that the browser fetches from a special directory on the disk and installs as an extension to itself, as illustrated in Fig. 7-20(a). Because plug-ins

run inside the browser, they have access to the current page and can modify its appearance. After the plug-in has done its job (usually after the user has moved to a different Web page), the plug-in is removed from the browser's memory.

Figure 7-20. (a) A browser plug-in. (b) A helper application.



The Server Side :

So much for the client side. Now let us take a look at the server side. As we saw above, when the user types in a URL or clicks on a line of hypertext, the browser parses the URL and interprets the part between http:// and the next slash as a DNS name to look up. Armed with the IP address of the server, the browser establishes a TCP connection to port 80 on that server. Then it sends over a command containing the rest of the URL, which is the name of a file on that server. The server then returns the file for the browser to display.

To a first approximation, a Web server is similar to the server of Fig. 6-6. That server, like a real Web server, is given the name of a file to look up and return. In both cases, the steps that the server performs in its main loop are:

1. Accept a TCP connection from a client (a browser).

2. Get the name of the file requested.

3. Get the file (from disk).

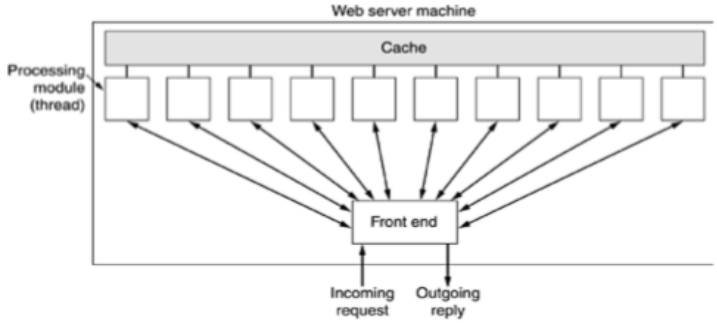
4. Return the file to the client. 5. Release the TCP connection.

Fig. 7-21. The k + 1 threads all belong to the same process so the processing modules all have access to the cache within the process' address space. When a request comes in, the front end accepts it and builds a short record describing it. It then hands

the record to one of the processing modules. In another possible design, the front end is eliminated and each processing module tries to acquire its own requests, but a locking protocol is then required to prevent conflicts.

Figure 7-21. A multithreaded Web server with a front end and processing modules.

Figure 7-21. A multithreaded Web server with a front end and processing modules.



URLs—Uniform Resource Locators :

Web pages may contain ***pointers to other Web pages****. Now it is time to see in a bit more detail how these pointers are implemented. When the Web was first created, it was immediately apparent that having one page point to another Web page required mechanisms for naming and locating pages.*

In particular, three questions had to be answered before a selected page could be displayed:

1. *What is the page called?*
2. *Where is the page located?*
3. *How can the page be accessed?*

Each page *is assigned a* ***URL*** *(Uniform Resource Locator) that effectively serves as the page's* ***worldwide name****.*

URLs have 3 parts*: the protocol (also known as the scheme), the DNS name of the machine on which the page is located, and a local name uniquely indicating the specific page (usually just a file name on the machine where it resides).*

http://www.cs.vu.nl/video/index-en.html

A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path

The *protocol* is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

The URL can optionally contain the port number of the server. If the *port* is included, it is inserted between the host and the path, and it is separated from the host by a colon.

Path is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

Statelessness and Cookies:

- The web basically stateless. There is no concept of login session.
- The browser sends a request to a server and gets back a file .
- Then the server forgets that it has ever seen that particular client.
- When the client request a webpage , the server can supply additional information along with the request page.
- This information may include a cookie which is a small file .
- browser store offered cookies in a cookies dictionaries on the client hard disk unless the user has dis able cookies.
- Cookies are Just files or strings not executable programs.

Cookies may contain 5 fields.

1. Domain
2. Path
3. Content
4. Expires
5. Secure

Static Web Documents

The basis of the Web is transferring Web pages from server to client. In the simplest form, Web pages are static,

that is, are just files sitting on some server waiting to be retrieved. In this context, even a video is a static Web page because it is just a file. In this section we will look at static Web pages in detail. In the next one, we will examine dynamic content.

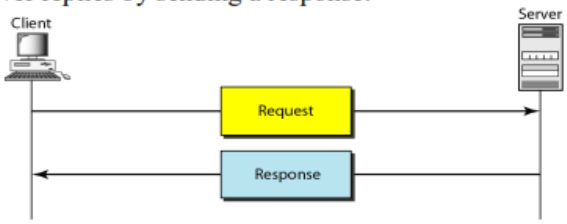
HTML—The Hyper Text Markup Language :

Web pages are currently written in a language called HTML (HyperText Markup Language). HTML allows users to produce Web pages that include text, graphics, and pointers to other Web pages. HTML is a markup language, a language for describing how documents are to be formatted. The term "markup" comes from the old days when copyeditors actually marked up documents to tell the printer—in those days, a human being—which fonts to use, and so on. Markup languages thus contain explicit commands for formatting.

For example, in HTML, means start boldface mode, and means leave boldface mode. The advantage of a markup language over one with no explicit markup is that writing a browser for it is straightforward: the browser simply has to understand the markup commands. TeX and troff are other well-known examples of markup languages.

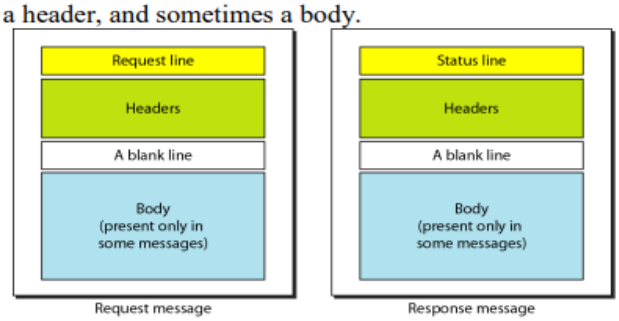
5.5.1 HTTP Transaction

Below figure illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response.

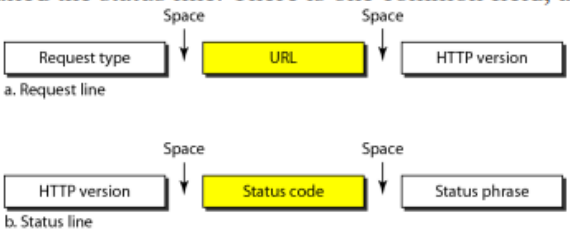


5.5.2 HTTP Messages

The formats of the request and response messages are similar; both are shown in below figure. A request message consists of a request line, a header, and sometimes a body. A response message consists of a status line, a header, and sometimes a body.



Request and Status Lines. The first line in a request message is called a request line; the first line in the response message is called the status line. There is one common field, as shown in below figure.



Request type. This field is used in the request message. In version 1.1 of HTTP, several request types are defined. The request type is categorized into *methods* as defined in below table.

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Inquires about available options

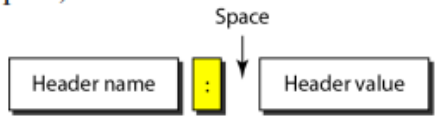
URL. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet..

Version. The most current version of HTTP is 1.1.

Status code. This field is used in the response message. The status code field is similar to those in the FTP and the SMTP protocols. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request.

Status phrase. This field is used in the response message. It explains the status code in text form.

Header The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value.



A header line belongs to one of four categories: **general header, request header, response header, and entity header.**

General header The general header gives general information about the message and can be present in both a request and a response. Below table lists some general headers with their descriptions.

Header	*Description*
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

Figure 7-26. (a) The HTML for a sample Web page. (b) The formatted page.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a>
   <li> <a href="http://widget.com/products/little"> Little widgets </a>
</ul>
<h2> Telephone numbers </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS
   <li> By fax: 1-415-765-4321
</ul>
</body>
</html>
```

(b)

Figure 7-27. A selection of common HTML tags. Some can have additional parameters.

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
...	Set ... in boldface
...	Set ... in italics
...	Center ... on the page horizontally
...	Brackets an unordered (bulleted) list
...	Brackets a numbered list
...	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
<img src="...">	Displays an image here
 ... 	Defines a hyperlink

XHTML—The eXtended HyperText Markup Language

HTML keeps evolving to meet new demands. Many people in the industry feel that in the future, the majority of Web-enabled devices will not be PCs, but wireless, handheld PDA-type devices. These devices have limited memory for large browsers full of heuristics that try to somehow deal with syntactically incorrect Web pages. Thus, the next step after HTML 4 is a language that is Very Picky. It is called XHTML (eXtended HyperText Markup Language) rather than HTML 5 because it is essentially HTML 4 reformulated in XML. By this we mean that tags such as <h1> have no intrinsic meaning. To get the HTML 4 effect, a definition is needed in the XSL file. XHTML is the new Web standard and should be used for all new Web pages to achieve maximum portability across platforms and browsers.

There are six major differences and a variety of minor differences between XHTML and HTML 4, Let us now go over the major differences. First, XHTML pages and browsers must strictly conform to the standard. No more shoddy Web pages. This property was inherited from XML.

Second, all tags and attributes must be in lower case. Tags like <HTML> are not valid in XHTML. The use of tags like <html> is now mandatory. Similarly, <img SRC="pic001.jpg"> is also forbidden because it contains an upper-case attribute.

Third, closing tags are required, even for </p>. For tags that have no natural closing tag, such as
, <hr>, and <img>, a slash must precede the closing ">," for example

Fourth, attributes must be contained within quotation marks. For example,

EMAIL (Electronic mail):

Electronic mail, or email, is a very popular application in computer networks such as the Internet. Email appeared in the early 1970s and allows users to exchange text based messages. Initially, it was mainly used to exchange short messages.

One of the most popular Internet services is electronic mail (e-mail). At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

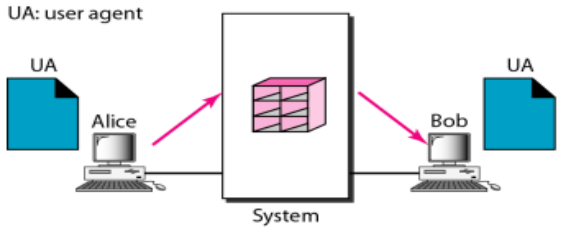
Electronic mail or E-mail as it is popularly called, is a system that allows a person or a group to electronically communicate with each other through a network. Presently people can now receive and send e-mail to: □ nearly any country in the world. □ one of millions of computer users. □ many users at once. □ computer programs. The first e-map systems consisted of file transfer protocols, with the convention that the first line of each message contained the recipient address.

Architecture

1. First Scenario:

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same system; they are directly connected to a shared system. The administrator has created one mailbox for each user where the received messages are stored. A mailbox is part of a local hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice, a user, needs to send a message to Bob, another user, Alice runs a user agent (UA) program to prepare the message and store it in Bob's mailbox. The message has the sender and recipient mailbox addresses (names of files). Bob can retrieve and read the contents of his mailbox at his convenience, using a user agent.

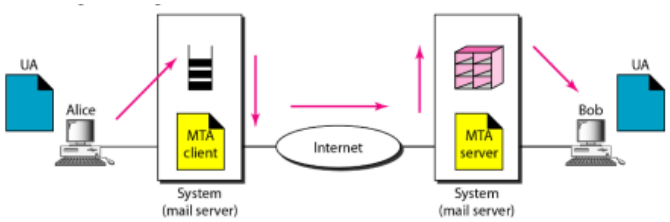
agent.



Second Scenario

In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different systems. The message needs to be sent over the Internet. Here we need user agents (UAs) and message transfer agents (MTAs).

UA: user agent
MTA: message transfer agent

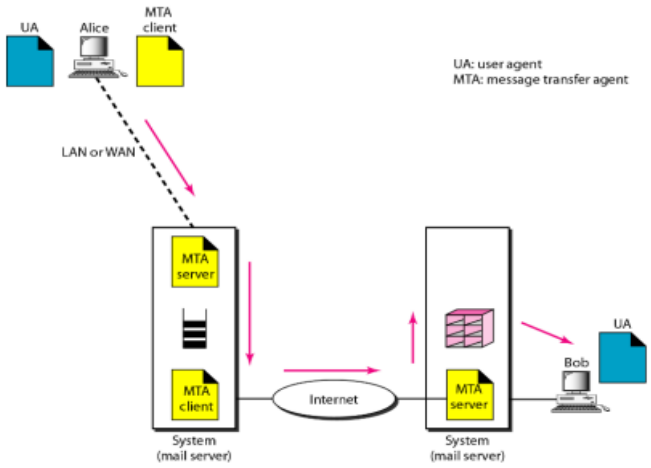


Alice needs to use a user agent program to send her message to the system at her own site. The system (sometimes called the mail server) at her site uses a queue to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site. The message, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one 'client and one server. Like most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection. The client, on the other hand, can be alerted by the system when there is a message in the queue to be sent.

---

3. *Third Scenario*

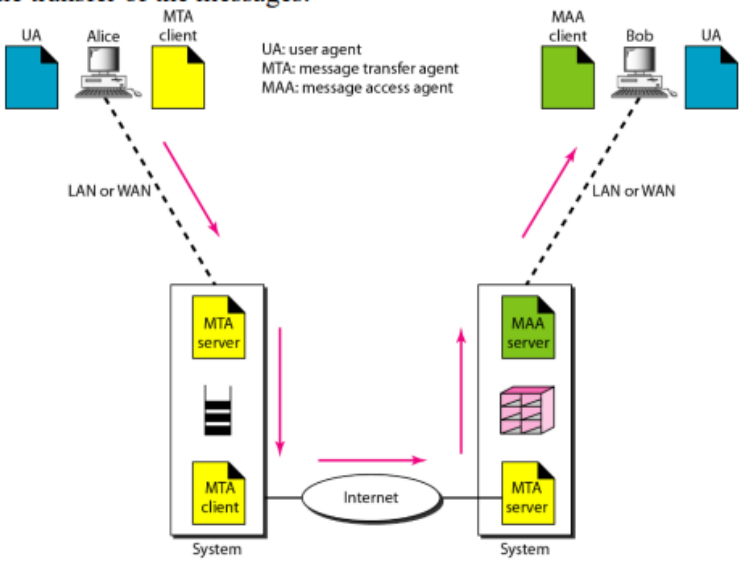
In the third scenario, Bob, as in the second scenario, is directly connected to his system. Alice, however, is separated from her system. Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization that uses one mail server for handling e-mails-all users need to send their messages to this mail server.



Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in tum, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox. At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client/server programs.

Fourth Scenario

In the fourth and most common scenario, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob needs to retrieve it. Here, we need another set of client/server agents, which we call message access agents (MAAs). Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages.



There are two important points here. First, Bob cannot bypass the mail server and use the MTA server directly. To use MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a-WAN, he must keep the connection up all the time. Neither of these situations is feasible today.

Architecture and Services :

- *e-mail systems can do and how they are organized.*
- *They normally consist* ***of two subsystems****: the* ***user agents****, which allow people to* ***read and send e-mail****, and the message*
- ***transfer agents****, which* ***move the messages from the source to the destination.***

  The user agents are local programs that provide a command based, menu-based, or graphical method for interacting with the e-mail system.

e-mail systems support Five basic functions:

Composition *refers to the process of creating messages and answers. Although any text editor can be used for the body of the message, the system itself can provide assistance with addressing and the numerous header fields attached to each message.*

Transfer *refers to moving messages from the originator to the recipient. In large part, this requires establishing a connection to the destination or some intermediate machine, outputting the message, and releasing the connection. The e-mail system should do this automatically, without bothering the user.*

Reporting *has to do with telling the originator what happened to the message. Was it delivered? Was it rejected? Was it lost?*

Displaying incoming messages *is needed so people can read their e-mail. Sometimes conversion is required or a special viewer must be invoked.*

Disposition *is the final step and concerns what the recipient does with the message after receiving it.*

Sending e-mail: *To send an email message the user must provide*

(a) message

(b) destination address and

(c) priority or security levels (options).

Message can be produced with a free standing text editor, a word processing program or by using a text editor built into the user agents. The format of an e-mail message is similar to that of a conventional letter.

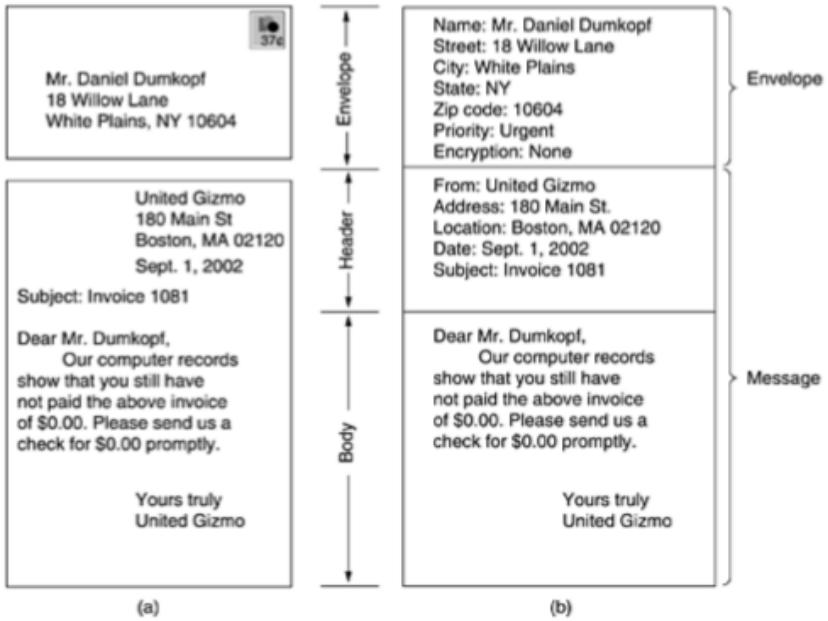
User Agent :

- *E-mail systems have two basic parts, as we have seen: the* ***user agents*** *and the* ***message transfer agents.***
- *In this section we will look at the* ***user agents.***
- *e-mail systems can do and how they are organized.*
- *They normally consist* ***of two subsystems****: the* ***user agents****, which allow people to read and send e-mail, and the message*

- **transfer agents**, which move the messages from the source to the destination. shortcuts.

Sending E-mail :

- To **send an e-mail message, a user must provide the message, the destination address, and possibly some other parameters.**
- The message can be produced with a free-standing text editor, a word processing program, or possibly with a specialized text editor built into the user agent.
- **The destination address must** be in a format that the user agent can deal with.
- **Many user agents expect addresses of the form user@dns-address.** Since we have **studied DNS.**



Envelopes and messages. (a) Paper mail. (b) Electronic mail.

Reading E-mail :

- when a user agent is started up, it looks at the user's **mailbox for incoming e-mail before displaying anything on the screen.**
- Then it may announce the number of messages in the mailbox or display a one-line summary of each one and wait for a command.

Figure 7-8. An example display of the contents of a mailbox.

#	Flags	Bytes	Sender	Subject
1	K	1030	asw	Changes to MINIX
2	KA	6348	trudy	Not all Trudys are nasty
3	K F	4519	Amy N. Wong	Request for information
4		1236	bal	Bioinformatics
5		104110	kaashoek	Material on peer-to-peer
6		1223	Frank	Re: Will you review a grant proposal
7		3110	guido	Our paper has been accepted
8		1204	dmr	Re: My student's visit

Message Formats:

Header	Meaning
To:	E-mail address(es) of primary recipient(s)
Cc:	E-mail address(es) of secondary recipient(s)
Bcc:	E-mail address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	E-mail address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

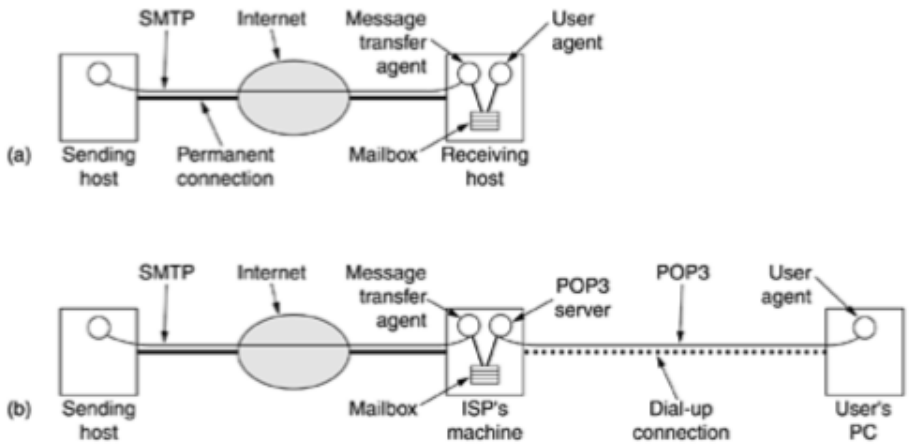
Message Transfer :

- *The simplest way to do this is to* ***establish a transport connection from the source machine to the destination machine and then just transfer the message.***

SMTP—The Simple Mail Transfer Protocol

- *Within the Internet, e-mail is delivered by having the source machine establish a TCP connection to port 25 of the destination machine. Listening to this port is an e-mail daemon that speaks SMTP (Simple Mail Transfer Protocol).*

Figure 7-15. (a) Sending and reading mail when the receiver has a permanent Internet connection and the user agent runs on the same machine as the message transfer agent. (b) Reading e-mail when the receiver has a dial-up connection to an ISP.



User Agent : *A user agent is normally a program that accepts a variety of commands for composing, receiving and replying to messages as well as manipulating mail boxes.*

Sending E-mail : *To send an e-mail a user must provide the message, the destination address and some other parameters. The message can be produced in any text editor (or) the one built in user agent. The destination address must be in the format that the user agent can deal with i.e., either DNS address (or) X.400 address. Most e-mail systems support mailing list, so that a user can send the same message to a list of people with a single command.*

Reading E-mail : *When a user agent is started up, it will look at the user's mailbox for incoming e-mail before displaying anything on the screen. It then announces the number of messages in the mail box(or) a one line summary of each one. In a sophisticated system the user can specify the fields to be displayed by providing the display format.*

Eg: 1. Message numbers

2. Flag etc.

Message format: *Message consist of a primitive envelope, some number of header field, blank line followed by message body. In normal usage, the user agent builds a message and passes it. To the message transfer agent which then uses some of the header fields to construct the actual envelope.*

Principal header include:

To : DNS address of primary recipient.

CC : DNS address of secondary recipient. In terms of delivery there is no distinction between primary and secondary (carbon copies).

BCC : Similar to CC, allows people to send copies to third parties without primary and secondary knowing it. **From** : Who wrote the message.

Sender : The one who sent the message.

Received : Added by each message transfers agent along the way used for finding bugs in routing system. Return path : Added by final message transfer agent intended to tell how to get back to the sender etc. Explain how e-mail works?

SMTP: Simple mail transfer protocol:

E-mail is delivered by having the source machine establish a TCP connection to destination. Listening to this port is an E-mail daemon that speaks SMTP. This daemon accepts incoming connections and copies messages from them to appropriate mail boxes. If the message cannot be delivered, an error message is given. After establishing a TCP connection, the sending machine operates as a client and waits for receiving entity to talk first. The server starts by giving its identity and informing whether (or) not it is prepared to receive mail. If it is not, the client releases the connection. If the server is ready, the client announces whom the E-mail is coming from and whom it is going to. If the recipient exists, the server gives a go-ahead to send the message. Then the client sends the message and the server acknowledges it. When the E-mail has been exchanged then the connection is released.

E-mail Gateways : SMTP does not work, when both sender and receiver are not on internet. In order to overcome this difficulty E-mail gateways are used. Here the sender establishes a TCP connection to the gateway and then uses SMTP to transfer the message. The daemon on the gateway then puts the message in a buffer of messages destined for host2. Late TPU (similar to TCP) is established with host2 and the message is transmitted.

Final Delivery :

Post Office Protocol (POP) Used to fetch e-mail from a remote mail box, has commands for user to logon, logout, fetch and delete messages. If fetches the mail and stores it in local system.

Interactive mail access protocol (IMAP) This protocol is used by a person having multiple systems (office, residence, car, etc). Here the Email server maintains a central repository that can be accessed from any machine. IMAP does not copy E-mail as POP .

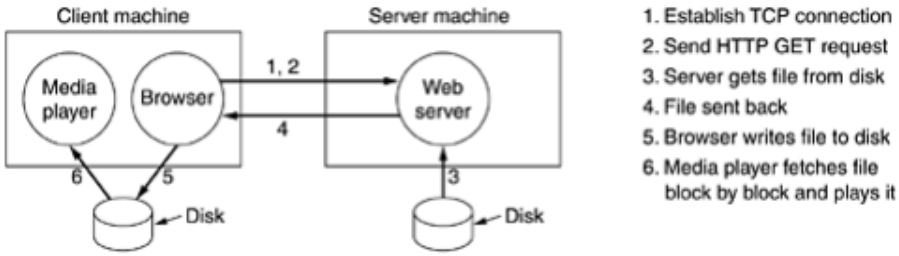
Streaming AUDIO and VIDEO:

7.4.3 Streaming Audio

Let us now move from the technology of digital audio to three of its network applications. Our first one is streaming audio, that is, listening to sound over the Internet. This is also called music on demand. In the next two, we will look at Internet radio and voice over IP, respectively.

The Internet is full of music Web sites, many of which list song titles that users can click on to play the songs. Some of these are free sites (e.g., new bands looking for publicity); others require payment in return for music, although these often offer some free samples as well (e.g., the first 15 seconds of a song). The most straightforward way to make the music play is illustrated in Fig. 7-59.

Figure 7-59. A straightforward way to implement clickable music on a Web page.



The process starts when the user clicks on a song. Then the browser goes into action. Step 1 is for it to establish a TCP connection to the Web server to which the song is hyperlinked. Step 2 is to send over a ***GET*** request in HTTP to request the song. Next (steps 3 and 4), the server fetches the song (which is just a file in MP3 or some other format) from the disk and sends it back to the browser. If the file is larger than the server's memory, it may fetch and send the music a block at a time.

Using the MIME type, for example, ***audio/mp3***, (or the file extension), the browser looks up how it is supposed to display the file. Normally, there will be a helper application such as RealOne Player, Windows Media Player, or Winamp, associated with this type of file. Since the usual way for the browser to communicate with a helper is to write the content to a scratch file, it will save the entire music file as a scratch file on the disk (step 5) first. Then it will start the media player and pass it the name of the scratch file. In step 6, the media player starts fetching and playing the music, block by block.

In principle, this approach is completely correct and will play the music. The only trouble is that the entire song must be transmitted over the network before the music starts. If the song is 4 MB (a typical size for an MP3 song) and the modem is 56 kbps, the user will be greeted by almost 10 minutes of silence while the song is being downloaded. Not all music lovers like this idea. Especially since the next song will also start with 10 minutes of download time, and the one after that as well.

To get around this problem without changing how the browser works, music sites have come up with the following scheme. The file linked to the song title is not the actual music file. Instead, it is what is called a **metafile**, a very short file just naming the music. A typical metafile might be only one line of ASCII text and look like this:

In most cases, the server named in the metafile is not the same as the Web server. In fact, it is generally not even an HTTP server, but a specialized media server. In this example, the media server uses **RTSP** (**Real Time Streaming Protocol**), as indicated by the scheme name ***rtsp***. It is described in RFC 2326.

The media player has four major jobs to do:

1. Manage the user interface.
2. Handle transmission errors.
3. Decompress the music.
4. Eliminate jitter.

Most media players nowadays have a glitzy user interface, sometimes simulating a stereo unit, with buttons, knobs, sliders, and visual displays. Often there are interchangeable front panels, called **skins**, that the user can drop onto the player. The media player has to manage all this and interact with the user.

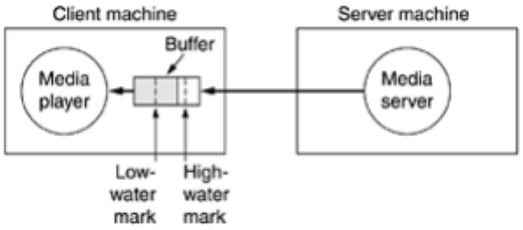
Its second job is dealing with errors. Real-time music transmission rarely uses TCP because an error and retransmission might introduce an unacceptably long gap in the music. Instead, the actual transmission is usually done with a protocol like RTP, which we studied in Chap. 6. Like most real-time protocols, RTP is layered on top of UDP, so packets may be lost. It is up to the player to deal with this.

Figure 7-60. When packets carry alternate samples, the loss of a packet reduces the temporal resolution rather than creating a gap in time.

The media player's third job is decompressing the music. Although this task is computationally intensive, it is fairly straightforward.

The fourth job is to eliminate jitter, the bane of all real-time systems. All streaming audio systems start by buffering about 10–15 sec worth of music before starting to play, as shown in Fig. 7-61. Ideally, the server will continue to fill the buffer at the exact rate it is being drained by the media player, but in reality this may not happen, so feedback in the loop may be helpful.

Figure 7-61. The media player buffers input from the media server and plays from the buffer rather than directly from the network.



Two approaches can be used to keep the buffer filled. With a **pull server**,as long as there is room in the buffer for another block, the media player just keeps sending requests for an additional block to the server. Its goal is to keep the buffer as full as possible.

mark. Then the media player tells it to pause. Since data will continue to pour in until the server has gotten the pause request, the distance between the high-water mark and the end of the buffer has to be greater than the bandwidth-delay product of the network. After the server has stopped, the buffer will begin to empty. When it hits the low-water mark, the media player tells the media server to start again. The low-water mark has to be positioned so that buffer underrun does not occur.

To operate a push server, the media player needs a remote control for it. This is what RTSP provides. It is defined in RFC 2326 and provides the mechanism for the player to control the server. It does not provide for the data stream, which is usually RTP. The main commands provided for by RTSP are listed in Fig. 7-62.

Figure 7-62. RTSP commands from the player to the server.

Command	Server action
DESCRIBE	List media parameters
SETUP	Establish a logical channel between the player and the server
PLAY	Start sending data to the client
RECORD	Start accepting data from the client
PAUSE	Temporarily stop sending data
TEARDOWN	Release the logical channel

Straming video:

7.4.6 Introduction to Video

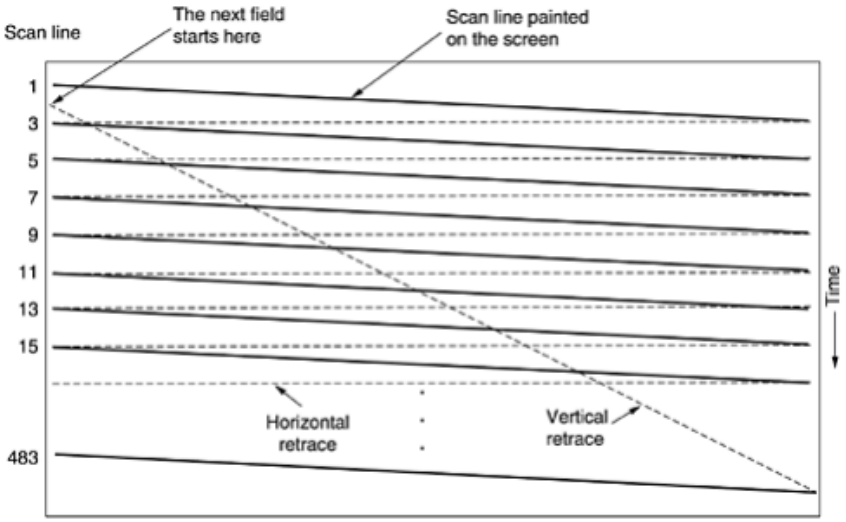
We have discussed the ear at length now; time to move on to the eye (no, this section is not followed by one on the nose). The human eye has the property that when an image appears on the retina, the image is retained for some number of milliseconds before decaying. If a sequence of images is drawn line by line at 50 images/sec, the eye does not notice that it is looking at discrete images. All video (i.e., television) systems exploit this principle to produce moving pictures.

Analog Systems

To understand video, it is best to start with simple, old-fashioned black-and-white television. To represent the two-dimensional image in front of it as a one-dimensional voltage as a function of time, the camera scans an electron beam rapidly across the image and slowly down it, recording the light intensity as it goes. At the end of the scan, called a **frame**, the beam

retraces. This intensity as a function of time is broadcast, and receivers repeat the scanning process to reconstruct the image. The scanning pattern used by both the camera and the receiver is shown in Fig. 7-70. (As an aside, CCD cameras integrate rather than scan, but some cameras and all monitors do scan.)

Figure 7-70. The scanning pattern used for NTSC video and television.



The exact scanning parameters vary from country to country. The system used in North and South America and Japan has 525 scan lines, a horizontal-to-vertical aspect ratio of 4:3, and 30 frames/sec. The European system has 625 scan lines, the same aspect ratio of 4:3, and 25 frames/sec. In both systems, the top few and bottom few lines are not displayed (to approximate a rectangular image on the original round CRTs). Only 483 of the 525 NTSC scan lines (and 576 of the 625 PAL/SECAM scan lines) are displayed. The beam is turned off during the vertical retrace, so many stations (especially in Europe) use this time to broadcast TeleText (text pages containing news, weather, sports, stock prices, etc.).

To allow color transmissions to be viewed on black-and-white receivers, all three systems linearly combine the RGB signals into a **luminance** (brightness) signal and two **chrominance** (color) signals, although they all use different coefficients for constructing these signals from the RGB signals. Oddly enough, the eye is much more sensitive to the luminance signal than to the chrominance signals, so the latter need not be transmitted as accurately. Consequently, the luminance signal can be broadcast at the same frequency as the old black-and-white signal, so it can be received on black-and-white television sets. The two chrominance signals are broadcast in narrow bands at higher frequencies. Some television sets have controls labeled brightness, hue, and saturation (or brightness, tint, and color) for controlling these three signals separately. Understanding luminance and chrominance is necessary for understanding how video compression works.

In the past few years, there has been considerable interest in **HDTV** (**High Definition TeleVision**), which produces sharper images by roughly doubling the number of scan lines. The United States, Europe, and Japan have all developed HDTV systems, all different and all mutually incompatible. Did you expect otherwise? The basic principles of HDTV in terms of scanning, luminance, chrominance, and so on, are similar to the existing systems. However, all three formats have a common aspect ratio of 16:9 instead of 4:3 to match them better to the format used for movies (which are recorded on 35 mm film, which has an aspect ratio of 3:2).

Digital Systems

The simplest representation of digital video is a sequence of frames, each consisting of a rectangular grid of picture elements, or **pixels**. Each pixel can be a single bit, to represent either black or white. The quality of such a system is similar to what you get by sending a color photograph by fax—awful. (Try it if you can; otherwise photocopy a color photograph on a copying machine that does not rasterize.)

The next step up is to use 8 bits per pixel to represent 256 gray levels. This scheme gives high-quality black-and-white video. For color video, good systems use 8 bits for each of the RGB colors, although nearly all systems mix these into composite video for transmission. While using 24 bits per pixel limits the number of colors to about 16 million, the human eye cannot even distinguish this many colors, let alone more. Digital color images are produced using three scanning beams, one per color. The geometry is the same as for the analog system of Fig. 7-70 except that the continuous scan lines are now replaced by neat rows of discrete