Computer Networking with TCP/IP

Module 5
Application Layer: Domain Name System (DNS)

CLIENT-SERVER PARADIGM

The purpose of a network, or an internetwork, is to provide services to users: A user at a local site wants to receive a service from a computer at a remote site. One way to achieve this purpose is to run two programs. A local computer runs a program to request a service from a remote computer; the remote computer runs a program to give service to the requesting program. This means that two computers, connected by an internet, must each run a program, one to provide a service and one to request a service.

Server

A server is a program running on the remote machine providing service to the clients. When it starts, it opens the door for incoming requests from clients, but it never initiates a service until it is requested to do so.

A server program is an infinite program. When it starts, it runs infinitely unless a problem arises. It waits for incoming requests from clients. When a request arrives, it responds to the request, either iteratively or concurrently as we will see shortly.

Client

A client is a program running on the local machine requesting service from a server. A client program is finite, which means it is started by the user (or another application program) and terminates when the service is complete.

Concurrency in Clients

Clients can be run on a machine either iteratively or concurrently. Running clients iteratively means running them one by one;

Concurrency in Servers

An iterative server can process only one request at a time; it receives a request, processes it, and sends the response to the requestor before it handles another request.

Domain Name System (DNS)

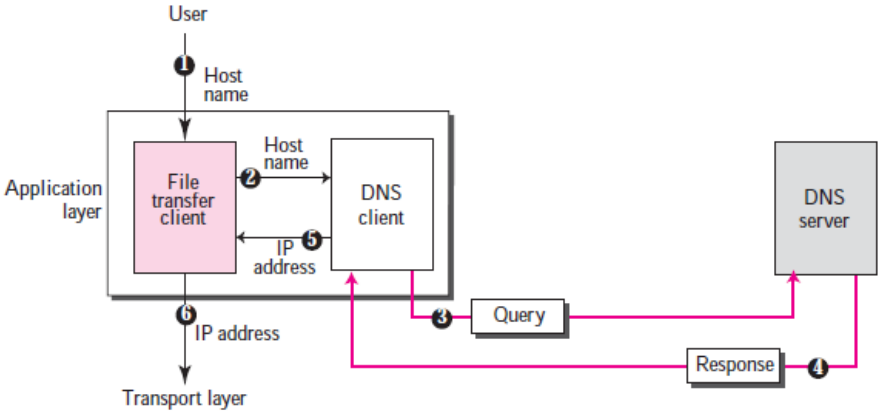
NEED FOR DNS

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.

One solution would be to store the entire host file in a single computer and allow access to this centralized information to every computer that needs mapping. But we know that this would create a huge amount of traffic on the Internet.

Another solution, the one used today, is to divide this huge amount of information into smaller parts and store each part on a different computer. In this method, the host that needs mapping can

contact the closest computer holding the needed information. This method is used by the Domain Name System (DNS).



A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as forouzan.com. However, the TCP/IP suite needs the IP address of the file transfer server to make the connection. The following six steps map the host name to an IP address.

1. The user passes the host name to the file transfer client.

2. The file transfer client passes the host name to the DNS client.

3. Each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.

4. The DNS server responds with the IP address of the desired file transfer server.

5. The DNS client passes the IP address to the file transfer server.

6. The file transfer client now uses the received IP address to access the file transfer server.

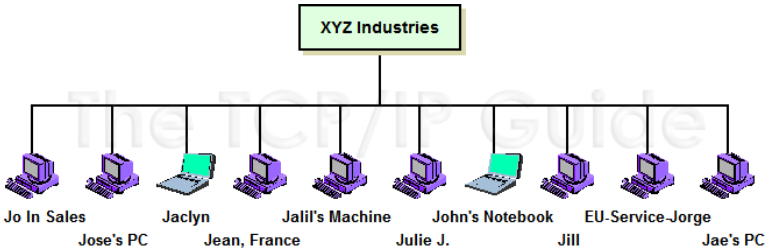
The purpose of accessing the Internet is to make a connection between the file transfer client and server, but before this can happen, another connection needs to be made between the DNS client and DNS server. In other words, we need two connections; the first is for mapping the name to an IP address; the second is for transferring files.

NAME SPACE

To be unambiguous, the names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. In other words, the names must be unique because the addresses are unique. A name space that maps each address to a unique name can be organized in two ways: **flat or hierarchical.**

Flat Name Space

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

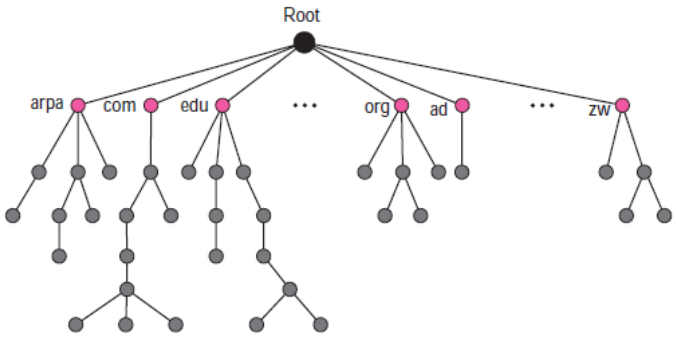


Hierarchical Name Space

In a hierarchical name space, each name is made of several parts. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility of the rest of the name can be given to the organization itself.

Domain Name Space

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.



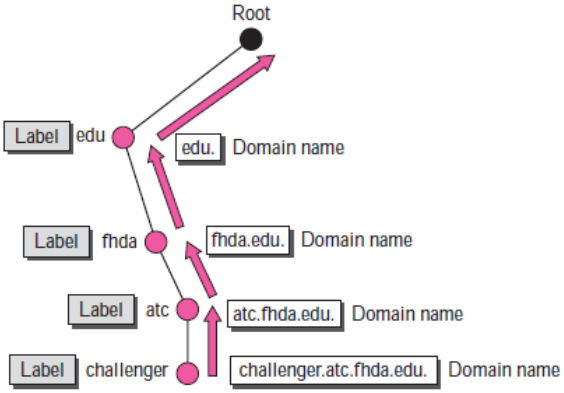
Label

Each node in the tree has a label, which is a string with a maximum of 63 characters.

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.

Figure 19.3 *Domain names and labels*



Fully Qualified Domain Name (FQDN) If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). An FQDN is a domain name that contains the full name of a host. It contains all labels, from the most specific to the most general, that uniquely define the name of the host.

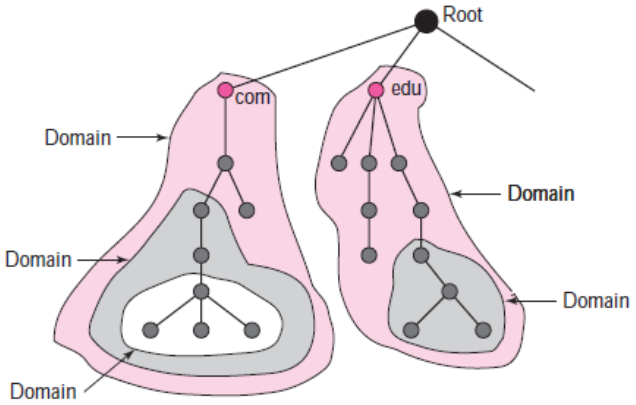
challenger.atc.fhda.edu.

Partially Qualified Domain Name (PQDN) If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client.

Challenger

Domain

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

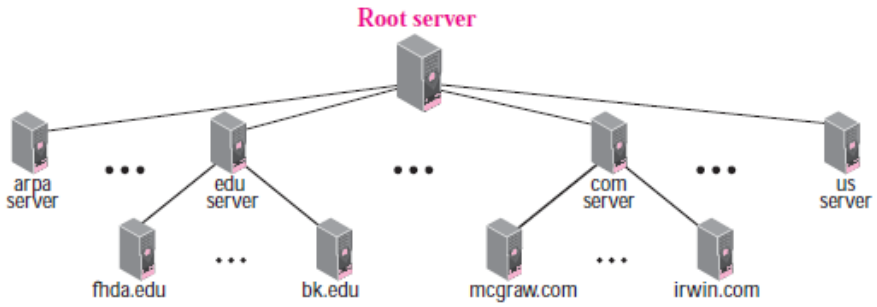


Distribution of Name Space

The information contained in the domain name space must be stored. However, it is very inefficient and also not reliable to have just one computer store such a huge amount of information.

Hierarchy of Name Servers

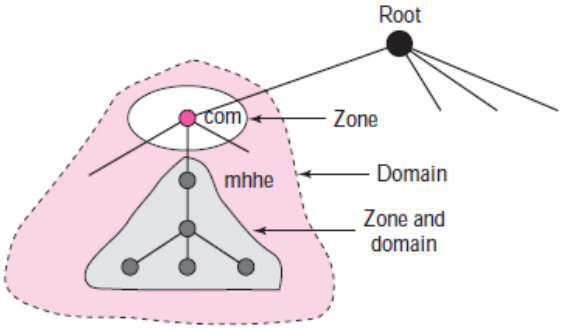
The solution to these problems is to distribute the information among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level.



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.



Root Server

A root server is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. There are several root servers, each covering the whole domain name space. The root servers are distributed all around the world.

Primary and Secondary Servers

DNS defines two types of servers: primary and secondary. **A primary server** is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.

DNS IN THE INTERNET

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) is divided into three different sections: generic domains, country domains, and the inverse domain.

Generic Domains

The generic domains define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database.

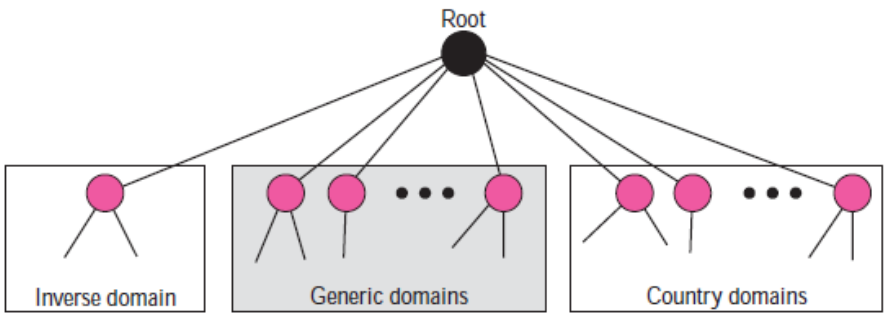
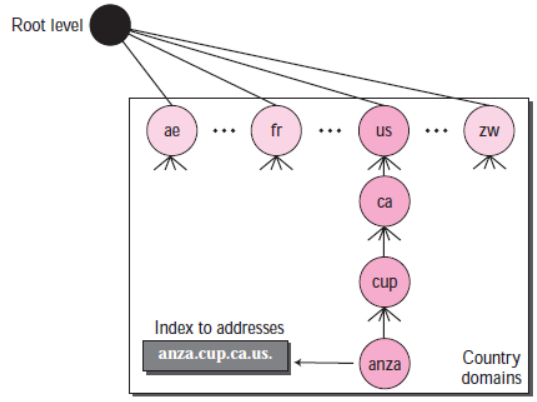


Table 19.1 *Generic domain labels*

Label	*Description*
aero	**Airlines and aerospace companies**
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other non-profit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

Country Domains

The country domains section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific, national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).



Inverse Domain

The inverse domain is used to map an address to a name. This may happen, for example, when a server has received a request from a client to do a task. Although the server has a file that contains a list of authorized clients, only the IP address of the client (extracted from the received IP packet) is listed. The server asks its resolver to send a query to the DNS server to map an address to a name to determine if the client is on the authorized list.

This type of query is called an **inverse or pointer (PTR) query.** To handle a pointer query, the inverse domain is added to the domain name space with the first-level node called arpa (for historical reasons). The second level is also one single node named in-addr (for inverse address). The rest of the domain defines IP addresses.

Registrar

How are the new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged.

RESOLUTION

Mapping a name to an address or an address to a name is called name-address resolution.

Resolver

DNS is designed as a client-server application. 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.

After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

Mapping Names to Addresses

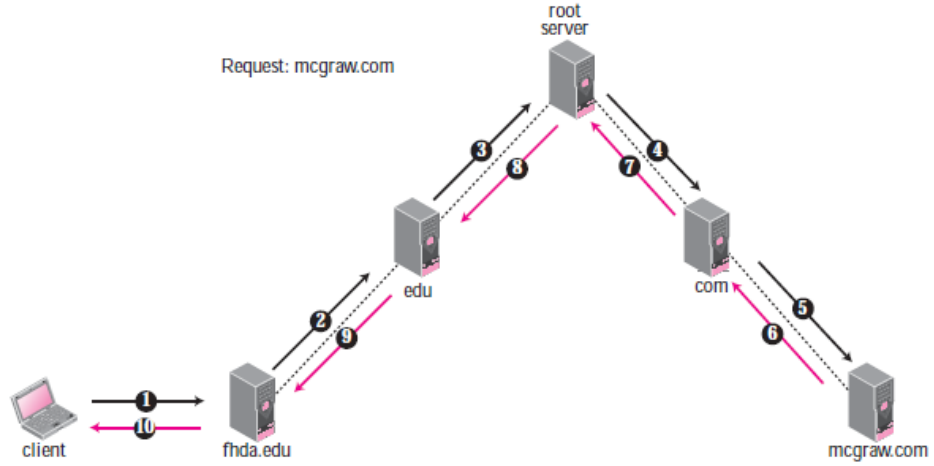
Most of the time, the resolver gives a domain name to the server and asks for the corresponding address. In this case, the server checks the generic domains or the country domains to find the mapping.

Mapping Addresses to Names

A client can send an IP address to a server to be mapped to a domain name.

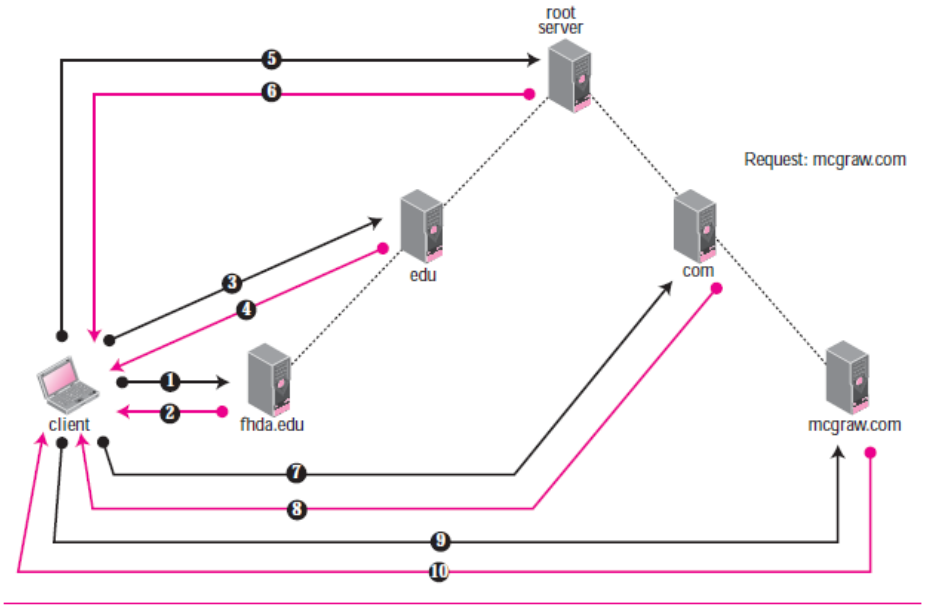
Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved, the response travels back until it finally reaches the requesting client.



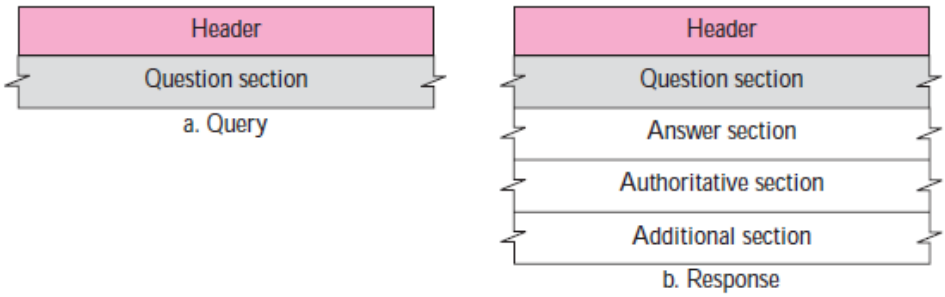
Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query. The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client. Now the client must repeat the query to the third server. This process is called iterative because the client repeats the same query to multiple servers.



DNS MESSAGES

DNS has two types of messages: query and response. Both types have the same format. The query message consists of a header and question records; the response message consists of a header, question records, answer records, authoritative records, and additional records.



Header

Both query and response messages have the same header format with some fields set to zero for the query messages. The header is 12 bytes.

Figure 19.15 *Header format*

Identification	Flags
Number of question records	Number of answer records (All 0s in query message)
Number of authoritative records (All 0s in query message)	Number of additional records (All 0s in query message)

Identification. This is a 16-bit field used by the client to match the response with the query. The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response.

Flags. This is a 16-bit field consisting of the subfields.

Figure 19.16 *Flags field*

a. **QR (query/response).** This is a 1-bit subfield that defines the type of message. If it is 0, the message is a query. If it is 1, the message is a response.
b. **OpCode**. This is a 4-bit subfield that defines the type of query or response (0 if standard, 1 if inverse, and 2 if a server status request).
c. **AA (authoritative answer).** This is a 1-bit subfield. When it is set (value of 1) it means that the name server is an authoritative server. It is used only in a response message.
d. **TC (truncated).** This is a 1-bit subfield. When it is set (value of 1), it means that the response was more than 512 bytes and truncated to 512. It is used when DNS uses the services of UDP (see Section 19.8 on Encapsulation).
e. **RD (recursion desired).** This is a 1-bit subfield. When it is set (value of 1) it means the client desires a recursive answer. It is set in the query message and repeated in the response message.
f. **RA (recursion available).** This is a 1-bit subfield. When it is set in the response, it means that a recursive response is available. It is set only in the response message.
g. **Reserved.** This is a 3-bit subfield set to 000.
h. **rCode.** This is a 4-bit field that shows the status of the error in the response.

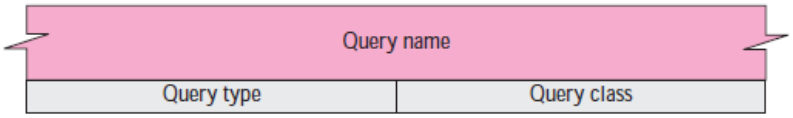
TYPES OF RECORDS

Two types of records are used in DNS. The question records are used in the question section of the query and response messages. The resource records are used in the answer, authoritative, and additional information sections of the response message.

Question Record

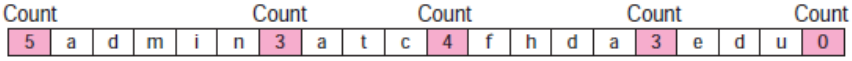
A question record is used by the client to get information from a server. This contains the domain name.

Figure 19.17 *Question record format*



Query name. This is a variable-length field containing a domain name.

Figure 19.18 *Query name format*



Query type. This is a 16-bit field defining the type of query.

Table 19.3 *Types*

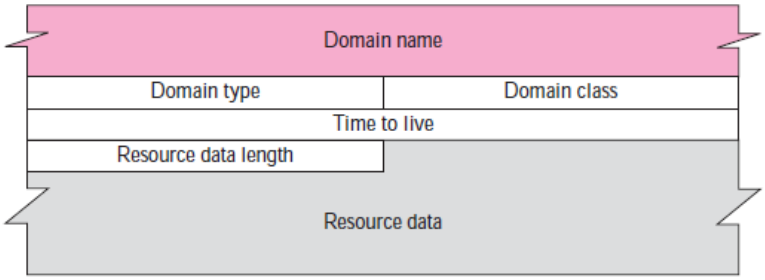
Type	*Mnemonic*	*Description*
1	A	**Address**. A 32-bit IPv4 address. It converts a domain name to an address.
2	NS	**Name server**. It identifies the authoritative servers for a zone.
5	CNAME	**Canonical name**. It defines an alias for the official name of a host.
6	SOA	**Start of authority**. It marks the beginning of a zone.
11	WKS	**Well-known services**. It defines the network services that a host provides.
12	PTR	**Pointer**. It is used to convert an IP address to a domain name.
13	HINFO	**Host information**. It defines the hardware and operating system.
15	MX	**Mail exchange**. It redirects mail to a mail server.
28	AAAA	**Address**. An IPv6 address (see Chapter 26).
252	AXFR	A request for the transfer of the entire zone.
255	ANY	A request for all records.

Query class. This is a 16-bit field defining the specific protocol using DNS.

Resource Record

Each domain name (each node on the tree) is associated with a record called the resource record. The server database consists of resource records. Resource records are also what is returned by the server to the client.

Figure 19.19 *Resource record format*



Domain name. This is a variable-length field containing the domain name. It is a duplication of the domain name in the question record.

Domain type. This field is the same as the query type field in the question record except the last two types are not allowed.

Domain class. This field is the same as the query class field in the question record.

Time-to-live. This is a 32-bit field that defines the number of seconds the answer is valid.

Resource data length. This is a 16-bit field defining the length of the resource data.

Resource data. This is a variable-length field containing the answer to the query (in the answer section) or the domain name of the authoritative server (in the authoritative section) or additional information (in the additional information section).

REGISTRARS

How are new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged.

TELNET

TELNET is an abbreviation for **TErminaL NETwork**. It is the standard TCP/IP protocol for virtual terminal service as proposed by ISO. TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

TELNET is a general-purpose client-server application program.

Time-Sharing Environment

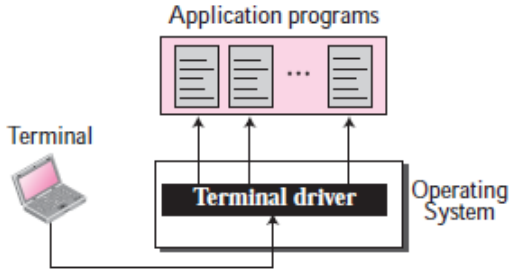
TELNET was designed at a time when most operating systems, such as UNIX, were operating in a time-sharing environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal.

In a time-sharing environment, all of the processing must be done by the central computer. When a user types a character on the keyboard, the character is usually sent to the computer and echoed to the monitor. Time-sharing creates an environment in which each user has the illusion of a dedicated computer.

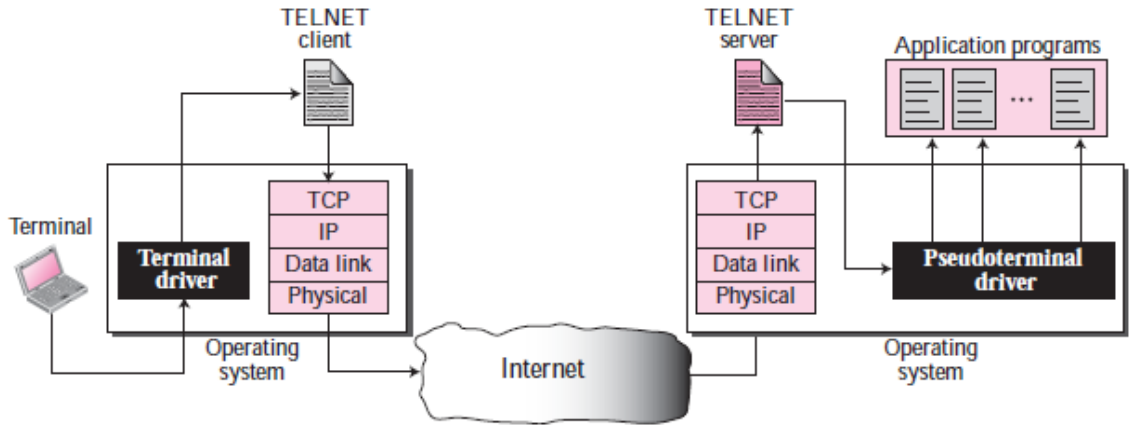
Login

In a time-sharing environment, users are part of the system with some right to access resources. Each authorized user has an identification and probably a password.

Local Login When a user logs into a local time-sharing system, it is called local login. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver.



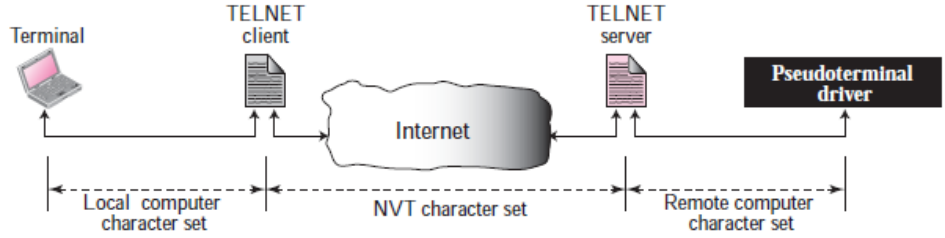
Remote Login When a user wants to access an application program or utility located on a remote machine, he or she performs remote login. Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them. The characters are sent 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 stack.



Network Virtual Terminal (NVT)

The mechanism to access a remote computer is complex. This is because every computer and its operating system accepts a special combination of characters as tokens.

Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network. The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer.



NVT Character Set

NVT uses two sets of characters, one for data and one for control. Both are 8-bit bytes.

a. Data Character b. Control Character

Data Characters For data, NVT normally uses what is called NVT ASCII. This is an 8-bit character set in which the seven lowest order bits are the same as US ASCII and the highest order bit is 0.

Control Characters To send control characters between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest order bit is set to 1.

Table 20.1 *Some NVT control characters*

Character	*Decimal*	*Binary*	*Meaning*
EOF	236	11101100	End of file
EOR	239	11101111	End of record
SE	240	11110000	Suboption end
NOP	241	11110001	No operation
DM	242	11110010	Data mark
BRK	243	11110011	Break
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase character
EL	248	11111000	Erase line
GA	249	11111001	Go ahead
SB	250	11111010	Suboption begin
WILL	251	11111011	Agreement to enable option
WONT	252	11111100	Refusal to enable option
DO	253	11111101	Approval to option request
DONT	254	11111110	Denial of option request
IAC	255	11111111	Interpret (the next character) as control

Embedding

TELNET uses only one TCP connection. The server uses the well-known port 23 and the client uses an ephemeral port. The same connection is used for sending both data and control characters. TELNET

accomplishes this by embedding the control characters in the data stream. TELNET accomplishes this by embedding the control characters in the data stream. However, to distinguish data from control characters, each sequence of control characters is preceded by a special control character called interpret as control (IAC).

Options

TELNET lets the client and server negotiate options before or during the use of the service. Options are extra features available to a user with a more sophisticated terminal. Users with simpler terminals can use default features. Some control characters discussed previously are used to define options.

Table 20.2 *Options*

Code	*Option*	*Meaning*
0	Binary	Interpret as 8-bit binary transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go-ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed
34	Line mode	Change to line mode

Binary. This option allows the receiver to interpret every 8-bit character received, except IAC, as binary data.

Echo. This option allows the server to echo data received from the client.

Suppress go-ahead. This option suppresses the go-ahead (GA) character.

Status. This option allows the user or the process running on the client machine to get the status of the options being enabled at the server site.

Timing mark. This option allows one party to issue a timing mark that indicates all previously received data has been processed.

Terminal type. This option allows the client to send its terminal type.

Terminal speed. This option allows the client to send its terminal speed.

Line mode. This option allows the client to switch to the line mode. We will discuss the line mode later.

Option Negotiation

To use any of the options mentioned in the previous section first requires option negotiation between the client and the server. Four control characters are used for this purpose;

Table 20.3 *NVT character set for option negotiation*

Character	*Code*	*Meaning 1*	*Meaning 2*	*Meaning 3*
WILL	251	Offering to enable	Accepting to enable	
WONT	252	Rejecting to enable	Offering to disable	Accepting to disable
DO	253	Approving to enable	Requesting to enable	
DONT	254	Disapproving to enable	Approving to disable	Requesting to disable

Enabling an Option

Some options can only be enabled by the server, some only by the client, and some by both. An option is enabled either through an offer or a request.

Offer to Enable A party can offer to enable an option if it has the right to do so. The offering can be approved or disapproved by the other party. The offering party sends the WILL command, which means "Will I enable the option?" The other party sends either the DO command, which means "Please do," or the DONT command, which means "Please don't."

Request to Enable A party can request from the other party the enabling of an option. The request can be accepted or refused by the other party. The requesting party sends the DO command, which means "Please do enable the option." The other party sends either the WILL command, which means "I will," or the WONT command, which means "I won't."

Disabling an Option

An option that has been enabled can be disabled by one of the parties. An option is disabled either through an offer or a request.

Offer to Disable A party can offer to disable an option. The other party must approve the offering; it cannot be disapproved. The offering party sends the WONT command, which means "I won't use this option any more." The answer must be the DONT command, which means "Don't use it anymore."

Offer to Disable A party can offer to disable an option. The other party must approve the offering; it cannot be disapproved. The offering party sends the WONT command, which means "I won't use this option any more." The answer must be the DONT command, which means "Don't use it anymore."

Symmetry

One interesting feature of TELNET is its symmetric option negotiation in which the client and server are given equal opportunity. This means that, at the beginning of connection, it is assumed that both sides are using a default TELNET implementation with no options enabled.

Suboption Negotiation

Some options require additional information. For example, to define the type or speed of a terminal, the negotiation includes a string or a number to define the type or speed. In either case, the two suboption characters indicated in Table 20.4 are needed for suboption negotiation.

Table 20.4 *NVT character set for suboption negotiation*

Character	*Decimal*	*Binary*	*Meaning*
SE	240	11110000	Suboption end
SB	250	11111010	Suboption begin

Controlling the Server

Some control characters can be used to control the remote server. When an application program is running on the local computer, special characters are used to interrupt (abort) the program (for example, Ctrl+c), or erase the last character typed (for example, delete key or backspace key), and so on. However, when a program is running on a remote computer, these control characters are sent to the remote machine. The user still types the same sequences, but they are changed to special characters and sent to the server.

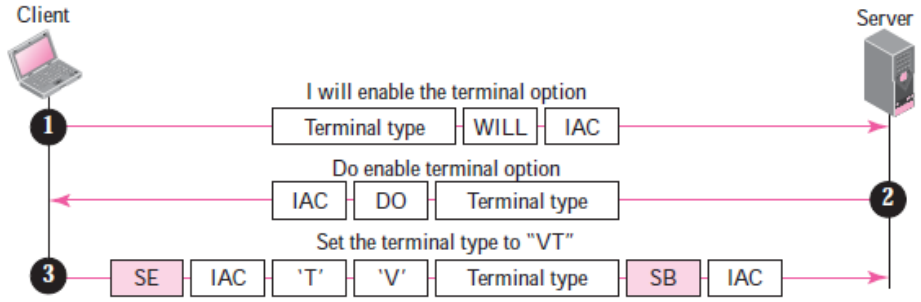


Table 20.5 *Characters used to control a program running on remote server*

Character	*Decimal*	*Binary*	*Meaning*
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase the last character
EL	248	11111000	Erase line

IP (interrupt process). When a program is being run locally, the user can interrupt (abort) the program if, for example, the program has gone into an infinite loop.

AO (abort output). This is the same as IP, but it allows the process to continue without creating output.

Modes of Operation

Most TELNET implementations operate in one of three modes: default mode, character mode, or line mode.

Default Mode

The default mode is used if no other modes are invoked through option negotiation. In this mode, the echoing is done by the client. The user types a character and the client echoes the character on the screen (or printer) but does not send it until a whole line is completed.

Character Mode

In the character mode, each character typed is sent by the client to the server. The server normally echoes the character back to be displayed on the client screen. In this mode the echoing of the character can be delayed if the transmission time is long (such as in a satellite connection).

Line Mode

A new mode has been proposed to compensate for the deficiencies of the default mode and the character mode. In this mode, called the line mode, line editing (echoing, character erasing, line erasing, and so on) is done by the client.

User Interface

The normal user does not use TELNET commands as defined above. Usually, the operating system (UNIX, for example) defines an interface with user-friendly commands.

Security Issue

TELNET suffers from security problems. Although TELNET requires a login name and password (when exchanging text), often this is not enough. A microcomputer connected to a broadcast LAN can easily

eavesdrop using snooper software and capture a login name and the corresponding password (even if it is encrypted).

SECURE SHELL (SSH)

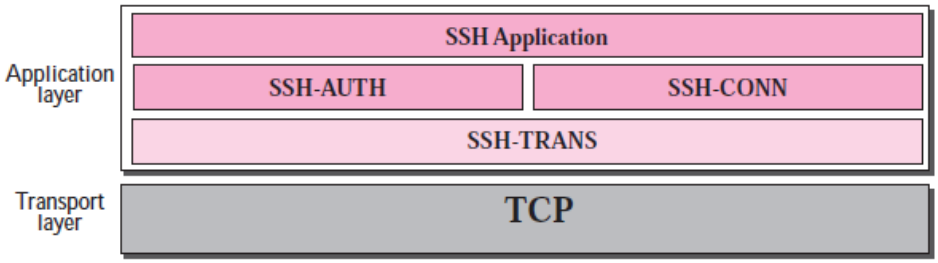
Another popular remote login application program is Secure Shell (SSH). SSH, like TELNET, uses TCP as the underlying transport protocol, but SSH is more secure and provides more services than TELNET.

Versions

There are two versions of SSH: SSH-1 and SSH-2, which are totally incompatible. The first version, SSH-1 is now deprecated because of security flaws in it.

Components

SSH is a proposed application-layer protocol with four components.



SSH Transport-Layer Protocol (SSH-TRANS)

Since TCP is not a secured transport layer protocol, SSH first uses a protocol that creates a secured channel on the top of TCP. This new layer is an independent protocol referred to as SSH-TRANS. When the software implementing this protocol is called, the client and server first use the TCP protocol to establish an insecure pro-connection. Then they exchange several security parameters to establish a secure channel on the top of the TCP.

Services provided by this protocol:

1. Privacy or confidentiality of the message exchanged.
2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder.
3. Server authentication, which means that the client is now sure that the server is the one that it claims to be.
4. Compression of the messages that improve the efficiency of the system and makes attack more difficult.

SSH Authentication Protocol (SSH-AUTH)

After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another software that can authenticate the client for the server.

SSH Connection Protocol (SSH-CONN)

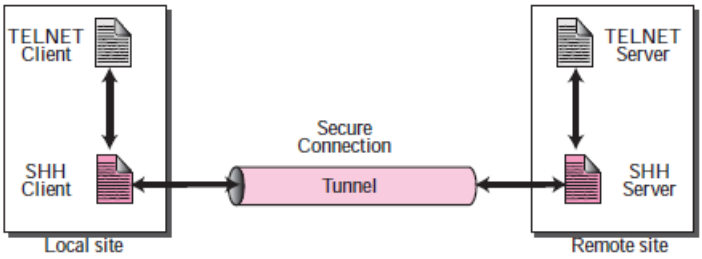
After the secured channel is established and both server and client are authenticated for each other, SHH can call a piece of software that implements the third protocol, SSHCONN.

SSH Applications

After the connection phase is completed, SSH allows several application programs to use the connection. Each application can create a logical channel as described above and then benefit from the secured connection.

Port Forwarding

One of the interesting services provided by the SSH protocol is to provide port forwarding. We can use the secured channels available in SSH to access an application program that does not provide security services. Application such as TELNET and SMTP can use the services of SSH using port forwarding mechanism. SSH port forwarding mechanism creates a tunnel through which the messages belonging to other protocol can travel. For this reason, this mechanism is sometimes referred to as **SSH tunneling.**



File Transfer Protocol(FTP)

File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another. Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first. For example, two systems may use different file name conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. All of these problems have been solved by FTP in a very simple and elegant approach.

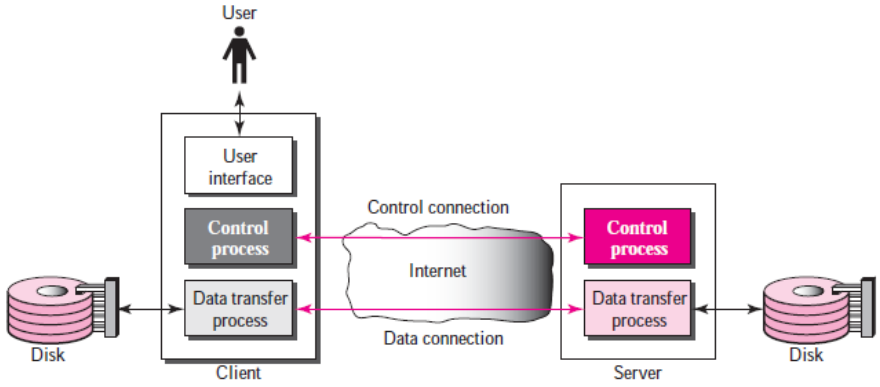
FTP differs from other client-server applications in that it establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. We need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred.

FTP uses the services of TCP. It needs two TCP connections. The well-known port 21 is used for the control connection and the well-known port 20 for the data connection.

The **control connection** remains connected during the entire interactive FTP session. The data connection is opened and then closed for each file transferred. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. In other words, when a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

Connections

The two FTP connections, control and data, use different strategies and different port numbers.



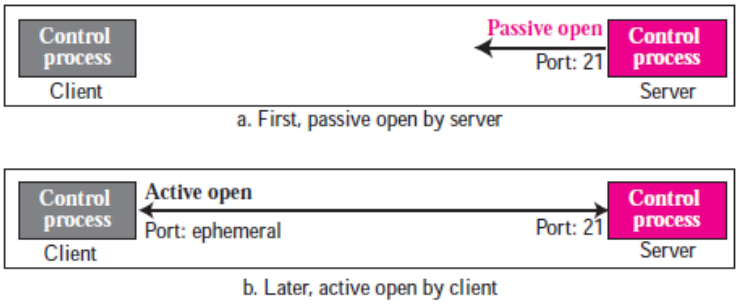
Control Connection

The control connection is created in the same way as other application programs. There are two steps:

1. The server issues a passive open on the well-known port 21 and waits for a client.

2. The client uses an ephemeral port and issues an active open.

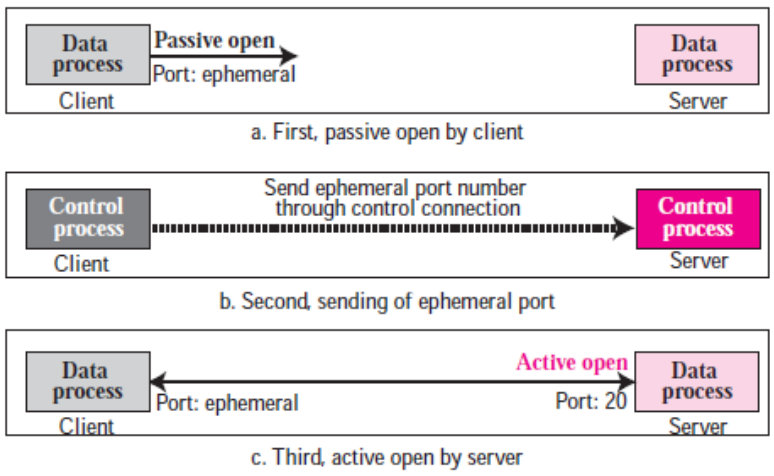
The connection remains open during the entire process.



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 what we have seen so far. The following shows how FTP creates a data connection:

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. The client sends this port number to the server using the PORT command (we will discuss this command shortly).
3. The server receives the port number and issues an active open using the wellknown port 20 and the received ephemeral port number.

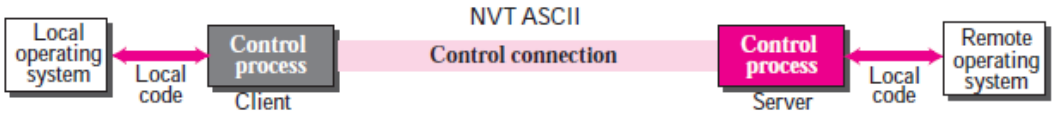


Communication

The FTP client and server, which run on different computers, must communicate with each other. These two computers may use different operating systems, different character sets, different file structures, and different file formats. FTP must make this heterogeneity compatible.

Communication over Control Connection

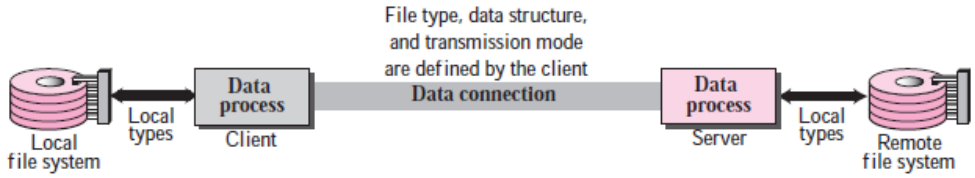
FTP uses the same approach as TELNET or SMTP to communicate across the control connection. It uses the NVT ASCII character set. Communication is achieved through commands and responses.



Communication over Data Connection

The purpose and implementation of the data connection are different from that of the control connection. 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 transmission mode.

Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication: file type, data structure, and transmission mode.



File Type FTP can transfer one of the following file types across the data connection:

ASCII file. This is the default format for transferring text files. Each character is encoded using NVT ASCII.

EBCDIC file. If one or both ends of the connection use EBCDIC encoding, the file can be transferred using EBCDIC encoding.

Image file. This is the default format for transferring binary files. The file is sent as continuous streams of bits without any interpretation or encoding.

Data Structure

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

File structure (default). The file has no structure. It is a continuous stream of bytes.

Record structure. The file is divided into records. This can be used only with text files.

Page structure. The file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially.

Transmission Mode

FTP can transfer a file across the data connection using one of the following three transmission modes:

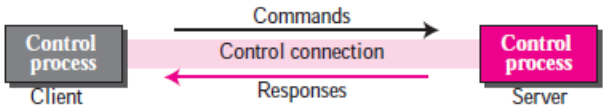
Stream mode. This is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes.

Block mode. Data can be delivered from FTP to TCP in blocks.

Compressed mode. If the file is big, the data can be compressed. The compression method normally used is run-length encoding.

Command Processing

FTP uses the control connection to establish a communication between the client control process and the server control process. During this communication, the commands are sent from the client to the server and the responses are sent from the server to the client.



Commands

Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument. We can roughly divide the commands into six groups: access commands, file management commands, data formatting commands, port defining commands, file transferring commands, and miscellaneous commands.

Access commands. These commands let the user access the remote system.

Command	*Argument(s)*	*Description*
USER	User id	User information
PASS	User password	Password
ACCT	Account to be charged	Account information
REIN		Reinitialize
QUIT		Log out of the system
ABOR		Abort the previous command

File management commands. These commands let the user access the file system on the remote computer.

Command	*Argument(s)*	*Description*
CWD	Directory name	Change to another directory
CDUP		Change to parent directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
NLIST	Directory name	List subdirectories or files without attributes
MKD	Directory name	Create a new directory
PWD		Display name of current directory
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify a file to be renamed
RNTO	File name (new)	Rename the file
SMNT	File system name	Mount a file system

Data formatting commands. These commands let the user define the data structure, file type, and transmission mode.

Command	*Argument(s)*	*Description*
TYPE	A (ASCII), E (EBCDIC), I (Image), N (Nonprint), or T (TELNET)	Define file type
STRU	F (File), R (Record), or P (Page)	Define organization of data
MODE	S (Stream), B (Block), or C (Compressed)	Define transmission mode

Port defining commands. These commands define the port number for the data connection on the client site. There are two methods to do this. In the **first method**, using the PORT command, the client can choose an ephemeral port number and send it to the server using a passive open. The server uses that port number and creates an active open.

In the **second method**, using the PASV command, the client just asks the server to first choose a port number. The server does a passive open on that port and sends the port number in the response The client issues an active open using that port number.

Command	*Argument(s)*	*Description*
PORT	6-digit identifier	Client chooses a port
PASV		Server chooses a port

File transfer commands. These commands actually let the user transfer files.

Command	*Argument(s)*	*Description*
RETR	File name(s)	Retrieve files; file(s) are transferred from server to client
STOR	File name(s)	Store files; file(s) are transferred from client to server
APPE	File name(s)	Similar to STOR, but if file exists, data must be appended to it
STOU	File name(s)	Same as STOR, but file name will be unique in the directory
ALLO	File name(s)	Allocate storage space for files at the server
REST	File name(s)	Position file marker at a specified data point
STAT	File name(s)	Return status of files

Miscellaneous commands. These commands deliver information to the FTP user at the client site.

Command	*Argument(s)*	*Description*
HELP		Ask information about the server
NOOP		Check if server is alive
SITE	Commands	Specify the site-specific commands
SYST		Ask about operating system used by the server

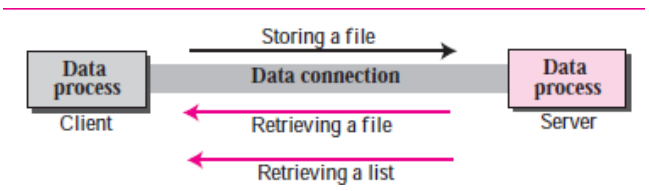
Responses

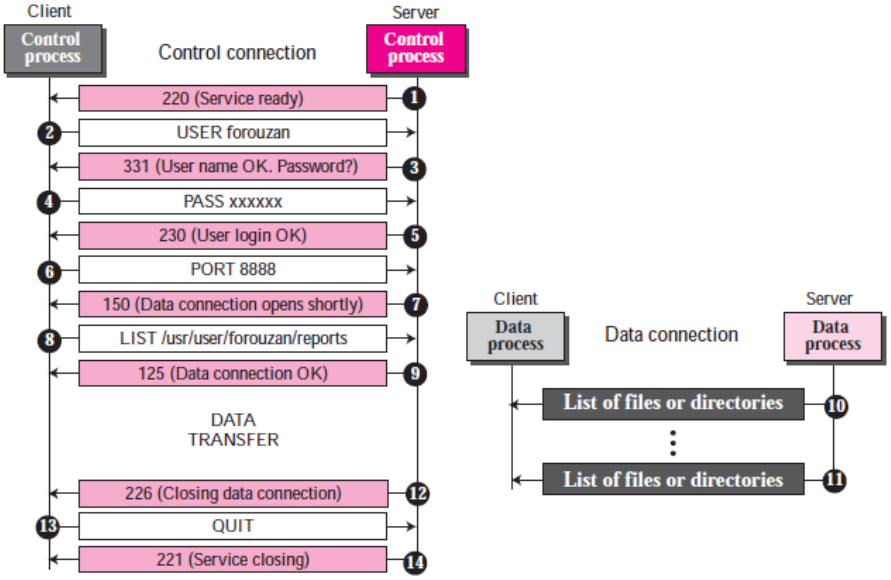
Every FTP command generates at least one response. A response has two parts: a three digit number followed by text. The numeric part defines the code; the text part defines needed parameters or extra explanations.

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.

1. A file is to be copied from the server to the client (download). This is called retrieving a file. It is done under the supervision of the RETR command.
2. A file is to be copied from the client to the server (upload). This is called storing a file. It is done under the supervision of the STOR command.
3. A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command.





1. After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.
2. The client sends the USER command.
3. The server responds with 331 (user name is OK, password is required).
4. The client sends the PASS command.
5. The server responds with 230 (user login is OK).
6. The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.
7. The server does not open the connection at this time, but it prepares itself for issuing an active open on the data connection between port 20 (server side) and the ephemeral port received from the client. It sends response 150 (data connection will open shortly).
8. The client sends the LIST message.
9. Now the server responds with 125 and opens the data connection.
10. The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.
11. The client now has two choices. It can use the QUIT command to request the closing of the control connection, or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.
12. After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control connection.

Anonymous FTP

To use FTP, a user needs an account (user name) and a password on the remote server. Some sites have a set of files available for public access. To access these files, a user does not need to have an account or password. Instead, the user can use anonymous as the user name and guest as the password.

User access to the system is very limited. Some sites allow anonymous users only a subset of commands. For example, most sites allow the user to copy some files, but do not allow navigation through the directories.

Security for FTP

The FTP protocol was designed when the security was not a big issue. Although FTP requires a password, the password is sent in plaintext (unencrypted), which means it can be intercepted and used by an attacker. The data transfer connection also transfers data in plaintext, which is insecure. between the FTP application layer and the TCP layer. In this case FTP is called SSL-FTP.

The sftp Program

Another way to transfer files using a secure channel is to use another independent protocol called sftp (secure file transfer protocol). This is actually a program in Unix called sftp that is part of the SSH protocol. When SSH has established a secure connection between an SSH client and an SSH server, one of the application programs that can use this connection (multiplexing) is sftp. In other words, sftp is part of the application component of the SSH. The sftp program is an interactive program that can work like FTP and uses a set of interface commands to transfer files between the SSH client and SSH server.

TFTP

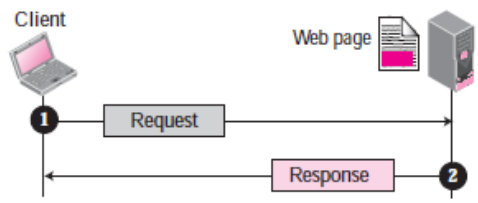
There are occasions when we need to simply copy a file without the need for all of the features of the FTP protocol. For example, when a diskless workstation or a router is booted, we need to download the bootstrap and configuration files. Here we do not need all of the sophistication provided in FTP. We just need a protocol that quickly copies the files. Trivial File Transfer Protocol (TFTP) is designed for these types of file transfer. It is so simple that the software package can fit into the read-only memory of a diskless workstation. It can be used at bootstrap time. The reason that it fits on ROM is that it requires only basic IP and UDP. However, there is no security for TFTP. TFTP can read or write a file for the client. Reading means copying a file from the server site to the client site. Writing means copying a file from the client site to the server site.

TFTP uses the services of UDP on the well-known port 69.

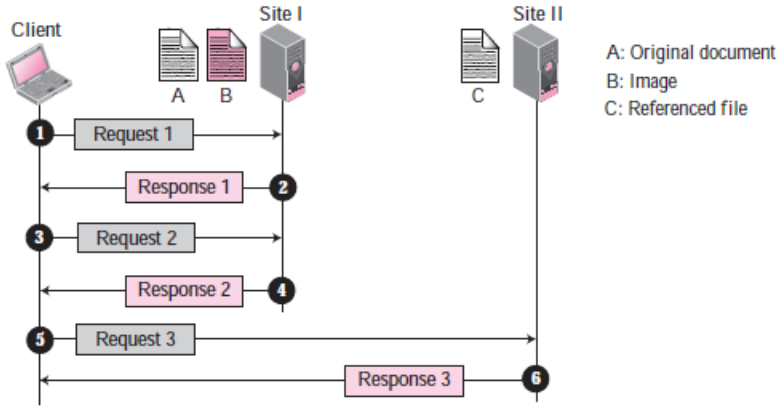
World Wide Web

ARCHITECTURE

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 sites. Each site holds one or more documents, referred to as **Web pages**. Each Web page, however, can contain some links to other Web pages in the same or other sites. In other words, a Web page can be simple or composite. **A simple Web page** has no link to other Web pages; a **composite Web page** has one or more links to other Web pages. Each Web page is a file with a name and address.



Example 1. Since the pictures are not stored as separate files, the whole document is a simple Web page.



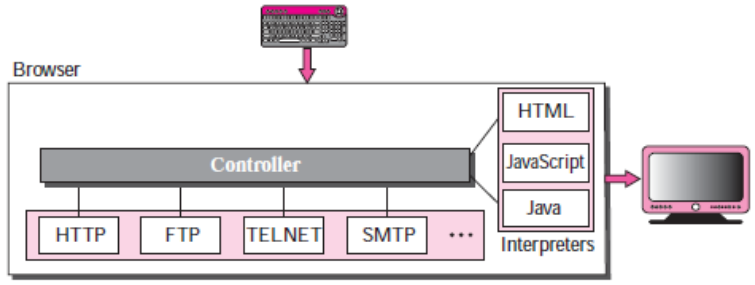
Example 2 We need to retrieve a scientific document that contains one reference to another text file and one reference to a large image.

Hypertext and Hypermedia

Hypertext means creating documents that refer to other documents. In a hypertext document, a part of text can be defined as a link to another document. When a hypertext is viewed with a browser, the link can be clicked to retrieve the other document. **Hypermedia** is a term applied to document that contains links to other textual document or documents containing graphics, video, or audio.

Web Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all of them use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters.



Web Server

The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk. A server can also become more efficient through multithreading or multiprocessing. In this case, a server can answer more than one request at a time.

Uniform Resource Locator (URL)

A client that wants to access a Web page needs the file name and the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource

locator (URL) is a standard locator 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 application program used to retrieve the document.

The host is the domain name of the computer on which the information is located.

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.

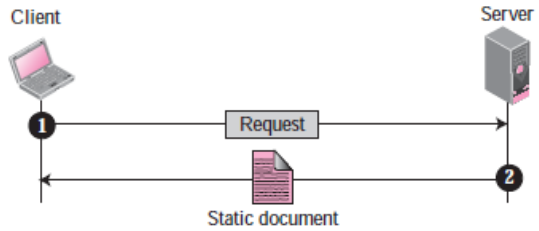
Path is the pathname of the file where the information is located.

WEB DOCUMENTS

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active. The category is based on the time the contents of the document are determined.

Static Documents

Static documents are fixed-content documents that are created and stored in a server. The client can get a copy of the document only. In other words, the contents of the file are determined when the file is created, not when it is used. Of course, the contents in the server can be changed, but the user cannot change them. When a client accesses the document, a copy of the document is sent.



Static documents are prepared using one of the several languages: Hypertext Markup Language (HTML), Extensible Markup Language (XML), Extensible Style Language (XSL), and Extended Hypertext Markup Language (XHTML).

Dynamic Documents

A dynamic document is created by a Web server whenever a browser requests the document. When a request arrives, the Web server runs an application program or a script that creates the dynamic document. The server returns the output of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another.

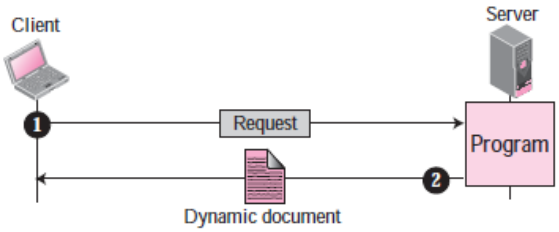
Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) is a technology that creates and handles dynamic documents. CGI is a set of standards that defines how a dynamic document is written, how data are input to the program, and how the output result is used.

CGI is not a new language; instead, it allows programmers to use any of several languages.

The term common in CGI indicates that the standard defines a set of rules that is common to any language or platform. The term gateway here means that a CGI program can be used to access other

resources such as databases, graphic packages, and so on. The term interface here means that there is a set of predefined terms, variables, calls, and so on that can be used in any CGI program.



Input In traditional programming, when a program is executed, parameters can be passed to the program. Parameter passing allows the programmer to write a generic program that can be used in different situations.

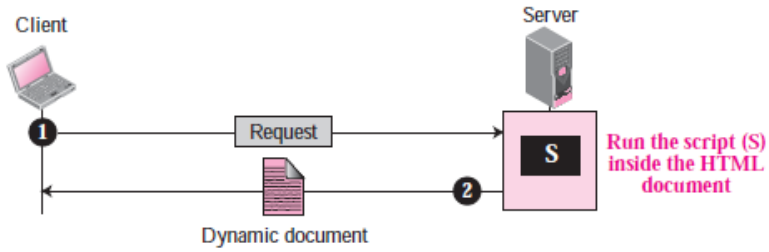
The input from a browser to a server is sent using a form. If the information in a form is small (such as a word), it can be appended to the URL after a question mark.

Output The whole idea of CGI is to execute a CGI program at the server site and send the output to the client (browser). The output is usually plain text or a text with HTML structures; however, the output can be a variety of other things.

Scripting Technologies for Dynamic Documents

The problem with CGI technology is the inefficiency that results if part of the dynamic document that is to be created is fixed and not changing from request to request.

For example, assume that we need to retrieve a list of spare parts, their availability, and prices for a specific car brand. Although the availability and prices vary from time to time, the name, description, and the picture of the parts are fixed. If we use CGI, the program must create an entire document each time a request is made. The solution is to create a file containing the fixed part of the document using HTML and embed a script, a source code, that can be run by the server to provide the varying availability and price section.



A few technologies have been involved in creating dynamic documents using scripts. Among the most common are Hypertext Preprocessor (PHP); Java Server Pages (JSP), which uses the Java language for scripting; Active Server Pages (ASP), a Microsoft product, which uses Visual Basic language for scripting; and ColdFusion, which embeds SQL database queries in the HTML document.

Active Documents

For many applications, we need a program or a script to be run at the client site. These are called active documents. For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes place. When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client (browser) site.

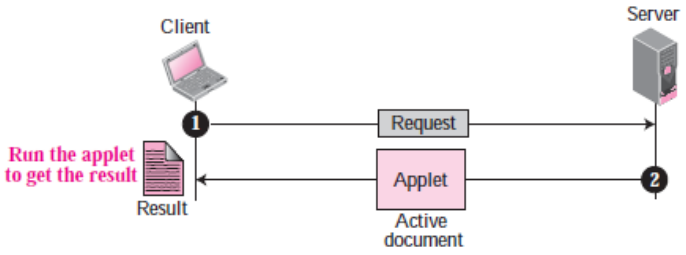
Java Applets

One way to create an active document is to use Java applets. Java is a combination of a high-level programming language, a run-time environment, and a class library that allows a programmer to write an active document (an applet) and a browser to run it.

An applet is a program written in Java on the server. It is compiled and ready to be run. The document is in bytecode (binary) format. The client process (browser) creates an instance of this applet and runs it.

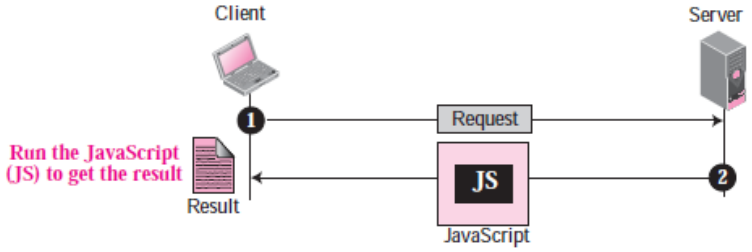
A Java applet can be run by the browser in two ways. In the **first method**, the browser can directly request the Java applet program in the URL and receive the applet in binary form.

In the **second method**, the browser can retrieve and run an HTML file that has embedded the address of the applet as a tag.



JavaScript

The idea of scripts in dynamic documents can also be used for active documents. If the active part of the document is small, it can be written in a scripting language; then it can be interpreted and run by the client at the same time. The script is in source code (text) and not in binary form. The scripting technology used in this case is usually JavaScript. JavaScript, which bears a small resemblance to Java, is a very high level scripting language developed for this purpose.



Active documents are sometimes referred to as client-site dynamic documents.

HTTP

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions like a combination of FTP and SMTP. It is similar to FTP because it transfers files and uses the services of TCP. However, it is much simpler than FTP because it uses only one TCP connection. There is no separate control connection; only data are transferred between the client and the server.

HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. Unlike SMTP, the HTTP messages are not destined to be read by humans; they are read and interpreted by the HTTP server and HTTP client (browser).

HTTP uses the services of TCP on well-known port 80.

HTTP Transaction

Although HTTP uses the services of TCP, HTTP itself is a stateless protocol, which means that the server does not keep information about the client. The client initializes the transaction by sending a request. The server replies by sending a response.

Request Message

A request message consists of a request line, a header, and sometimes a body.

Request Line The first line in a request message is called a request line. There are three fields in this line separated by some character delimiter The fields are called methods, URL, and Version. These three should be separated by a space character. At the end two characters, a carriage return followed by a line feed, terminate the line. The method field defines the request type.

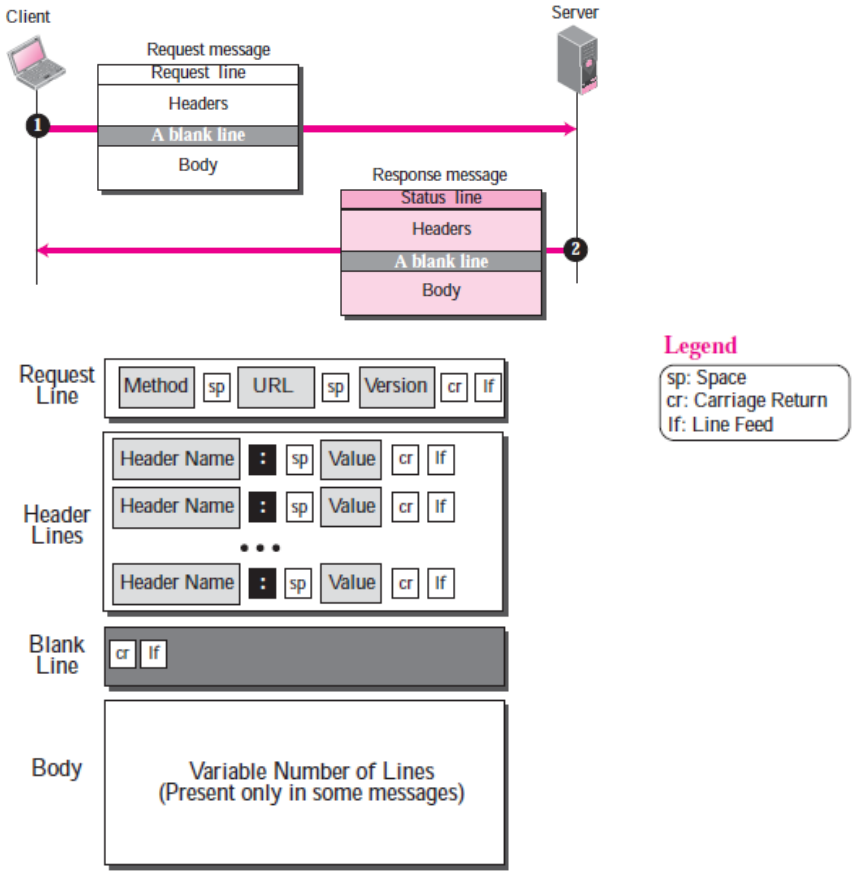


Table 22.1 *Methods*

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
DELETE	Remove the Web page
OPTIONs	Enquires about available options

Header Lines In Request Message After the request line, we can have zero or more request header lines. Each header line sends additional information from the client to the server.

Body In Request Message The body can be present in a request message. Usually, it contains the comment to be sent.

Response Message

A response message consists of a status line, header lines, a blank line and sometimes a body.

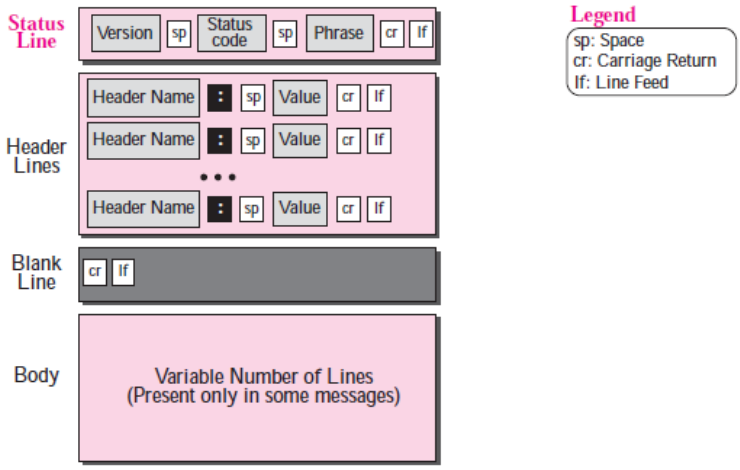
Status Line The first line in a response message is called the status line. There are three fields in this line separated by spaces and terminated by a carriage return and line feed. The first field defines the version of HTTP protocol, currently 1.1. The status code field defines the status of the request. 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. The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site. Finally, the codes in the 500 range indicate an error at the server site.

Header Lines In Response Message After the status line, we can have zero or more response header lines. Each header line sends additional information from the server to the client.

Table 22.3 *Status Codes and Status Phrases*

Status Code	*Status Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request received, continue.
101	Switching	The server is complying to switch protocols.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.
	Redirection	
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not modified.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable.

Body The body contains the document to be sent from the server to the client. The body is present unless the response is an error message.



Conditional Request

A client can add a condition in its request. In this case, the server will send the requested Web page if the condition is met or inform the client otherwise. One of the most common conditions imposed by the client is the time and date the Web page is modified. The client can send the header line If-Modified-Since to the request to tell the server that it needs the page if it is modified after a certain point of time.

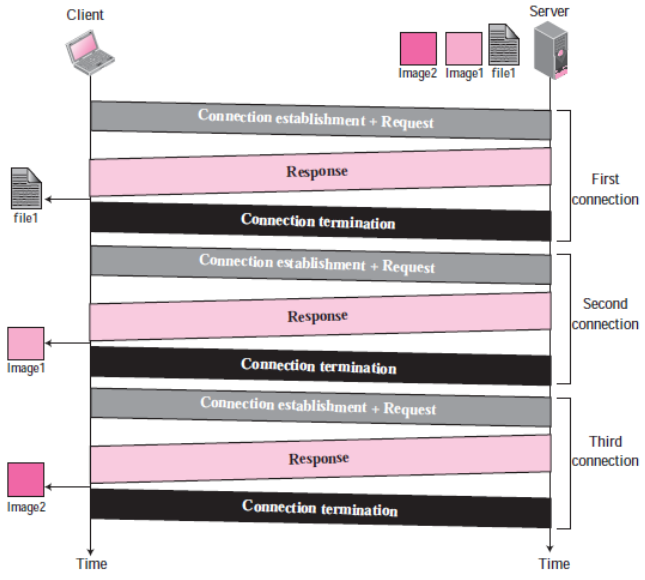
Persistence

HTTP, prior to version 1.1, specified a nonpersistent connection, while a persistent connection is the default in version 1.1.

Nonpersistent Connection

In a nonpersistent connection, one TCP connection is made for each request/response. The following lists the steps in this strategy:

1. The client opens a TCP connection and sends a request.
2. The server sends the response and closes the connection.
3. The client reads the data until it encounters an end-of-file marker; it then closes the connection.

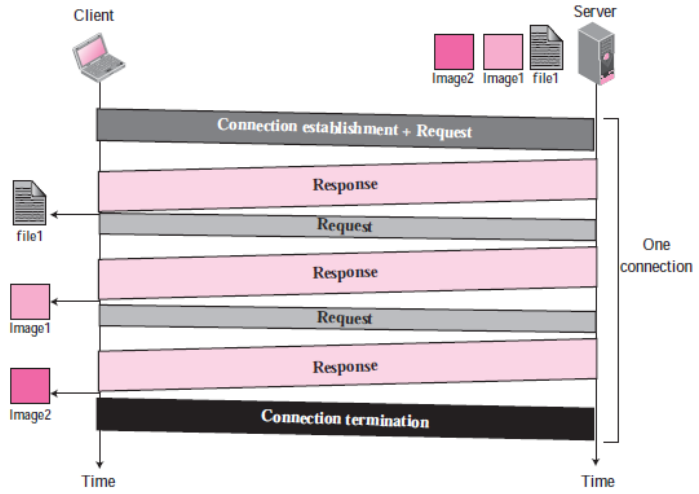


In this strategy, if a file contains link to N different pictures in different files (all located on the same server), the connection must be opened and closed N + 1 times. The nonpersistent strategy imposes high overhead on the server because the server needs N + 1 different buffers and requires a slow start procedure each time a connection is opened.

Eg: The client needs to access a file that contains two links to images. The text file and images are located on the same server.

Persistent Connection

HTTP version 1.1 specifies a persistent connection by default. In a persistent connection, the server leaves the connection open for more requests after sending a response. The server can close the connection at the request of a client or if a time-out has been reached. The sender usually sends the length of the data with each response. However, there are some occasions when the sender does not know the length of the data. This is the case when a document is created dynamically or actively. In these cases, the server informs the client that the length is not known and closes the connection after sending the data so the client knows that the end of the data has been reached.



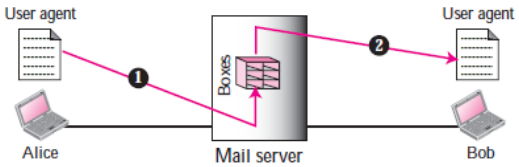
Electronic Mail

ARCHITECTURE

To explain the architecture of e-mail, we give four scenarios.

First Scenario

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same mail server; they are directly connected to a shared mail server. 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 needs to send a message to Bob, she 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.

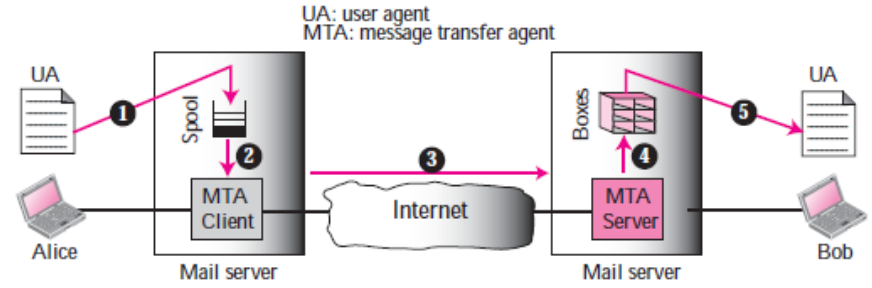


When the sender and the receiver of an e-mail are on the same mail server, we need only two user agents.

Second Scenario

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

Alice needs to use a user agent program to send her message to the mail server at her own site. The mail server at her site uses a queue (spool) 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 of the time because it does not know when a client will ask for a connection. The client, on the other hand, can be triggered by the system when there is a message in the queue to be sent.

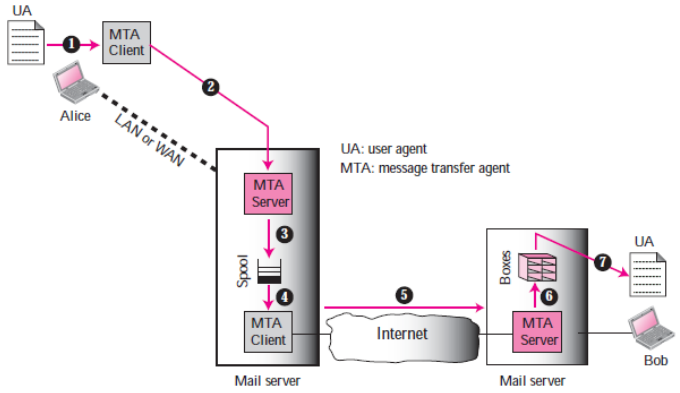


When the sender and the receiver of an e-mail are on different mail servers, we need two UAs and a pair of MTAs (client and server).

Third Scenario

Bob, as in the second scenario, is directly connected to his mail server. Alice, however, is separated from her mail server. Alice is either connected to the mail server 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 turn, 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.



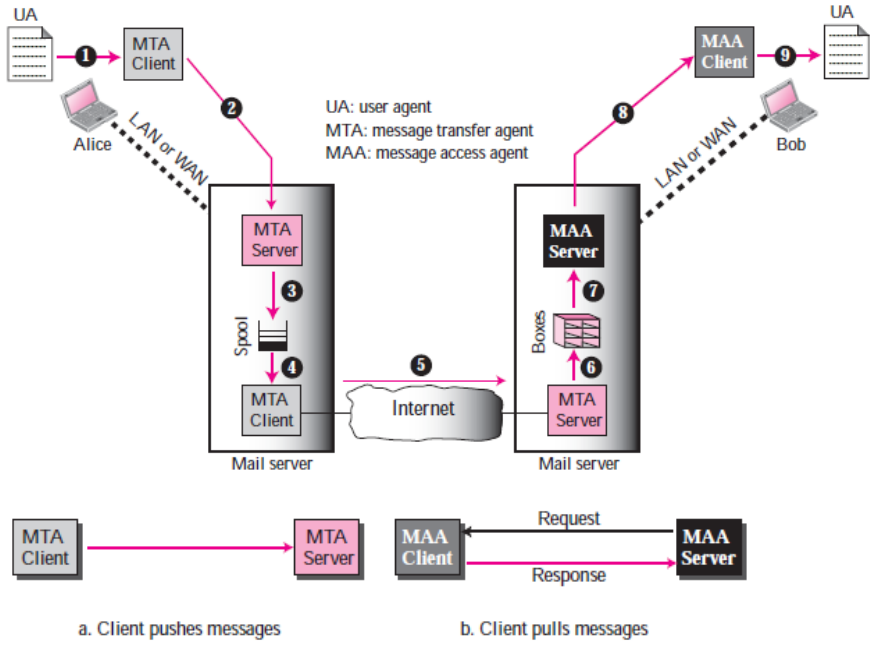
When the sender is connected to the mail server via a LAN or a WAN, we need two UAs and two pairs of MTAs (client and server).

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.

First, Bob cannot bypass the mail server and use the MTA server directly. To use the MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive.

Second, note that Bob needs another pair of client-server programs: message access programs. This is because an MTA client-server program is a push program: the client pushes the message to the server. Bob needs a pull program. The client needs to pull the message from the server.



When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server). This is the most common situation today.

USER AGENT

The first component of an electronic mail system is the user agent (UA). It provides service to the user to make the process of sending and receiving a message easier.

Services Provided by a User Agent

A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers.

User Agent Types

There are two types of user agents: command-driven and GUI-based. Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. A command-driven user agent normally accepts a one character command from the keyboard to perform its task.

Some examples of command-driven user agents are mail, pine, and elm.

Modern user agents are GUI-based. They contain graphical user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse.

Some examples of GUI-based user agents are Eudora, Outlook, and Netscape.

Sending Mail

To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an envelope and a message.

Envelope

The envelope usually contains the sender address, the receiver address, and other information.

Message

The message contains the header and the body. The header of the message defines the sender, the receiver, the subject of the message, and some other information. The body of the message contains the actual information to be read by the recipient.

Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the UA informs the user with a notice. If the user is ready to read the mail, a list is displayed in which each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address, the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts: a local part and a domain name, separated by an @ sign.

Local Part

The local part defines the name of a special file, called the user mailbox, where all of the mail received for a user is stored for retrieval by the message access agent.

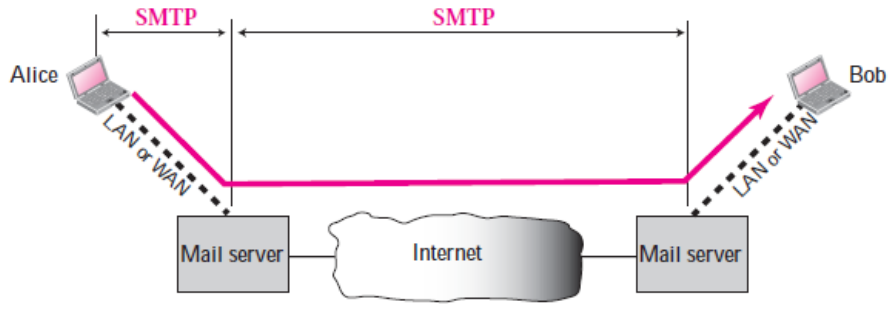
Domain Name

The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; they are sometimes called mail servers or exchangers. The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name (for example, the name of the organization).

MESSAGE TRANSFER AGENT: SMTP

The actual mail transfer is done through message transfer agents (MTAs). To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called Simple Mail Transfer

Protocol (SMTP). As we said before, two pairs of MTA client server programs are used in the most common situation (fourth scenario).

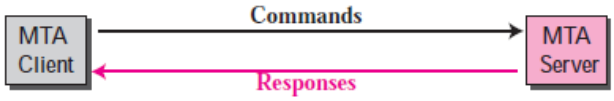


SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver.

SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We will discuss the mechanism of mail transfer by SMTP in the remainder of the section.

Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.



Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.

Commands

Commands are sent from the client to the server. The format of a command is shown below:

It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands listed in and described in more detail below.

Table 23.1 *Commands*

Keyword	*Argument(s)*	*Keyword*	*Argument(s)*
HELO	Sender's host name	NOOP	
MAIL FROM	Sender of the message	TURN	
RCPT TO	Intended recipient	EXPN	Mailing list
DATA	Body of the mail	HELP	Command name
QUIT		SEND FROM	Intended recipient
RSET		SMOL FROM	Intended recipient
VRFY	Name of recipient	SMAL FROM	Intended recipient

HELO. This command is used by the client to identify itself. The argument is the domain name of the client host. The format is HELO: challenger.atc.fhda.edu.

MAIL FROM. This command is used by the client to identify the sender of the message. The argument is the e-mail address of the sender.

MAIL FROM: forouzan@challenger.atc.fhda.edu

RCPT TO. This command is used by the client to identify the intended recipient of the message. The argument is the e-mail address of the recipient.

RCPT TO: betsy@mcgraw-hill.com

Table 23.2 *Responses*

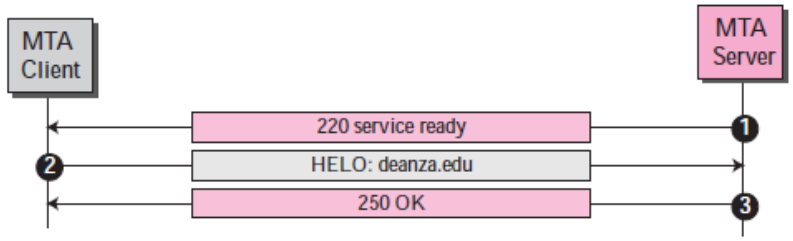
Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Mail Transfer Phases

The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.

Connection Establishment

After a client has made a TCP connection to the well-known port 25, the SMTP server starts the connection phase. This phase involves the following three steps,



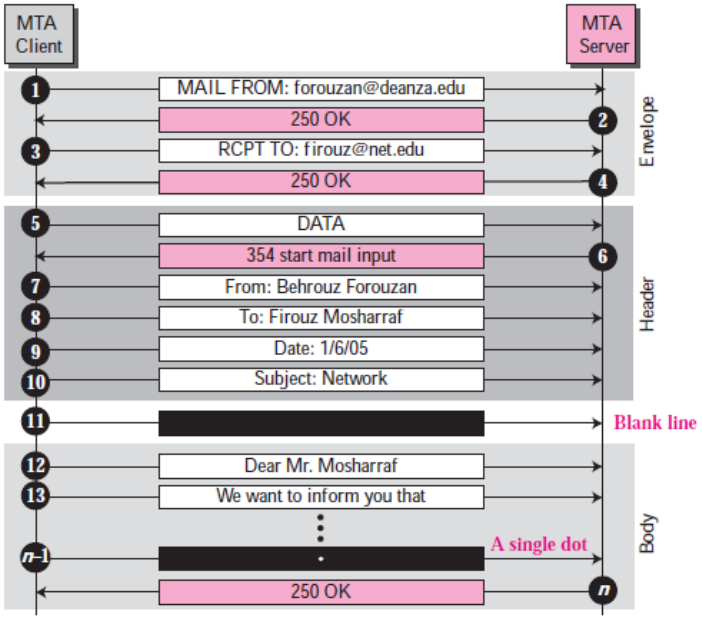
1. The server sends code 220 (service ready) to tell the client that it is ready to receive mail. If the server is not ready, it sends code 421 (service not available).
2. The client sends the HELO message to identify itself using its domain name address

3. The server responds with code 250 (request command completed) or some other code depending on the situation.

Message Transfer

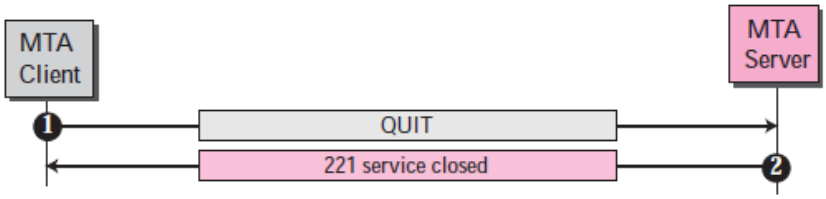
After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged. This phase involves eight steps.

1. The client sends the MAIL FROM message to introduce the sender of the message. It includes the mail address of the sender (mailbox and the domain name).
2. The server responds with code 250 or some other appropriate code.
3. The client sends the RCPT TO (recipient) message, which includes the mail address of the recipient.
4. The server responds with code 250 or some other appropriate code.
5. The client sends the DATA message to initialize the message transfer.
6. The server responds with code 354 (start mail input) or some other appropriate message.
7. The client sends the contents of the message in consecutive lines. Each line is terminated by a two-character end-of-line token (carriage return and line feed).
8. The server responds with code 250 (OK) or some other appropriate code.



Connection Termination

After the message is transferred successfully, the client terminates the connection.

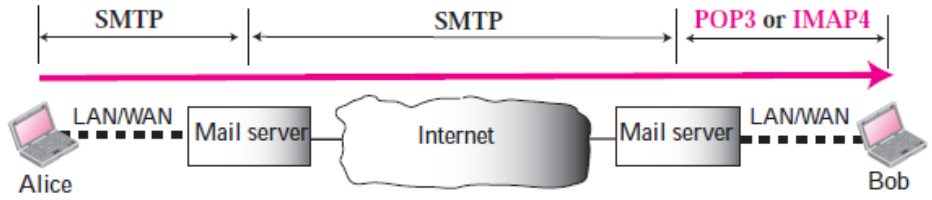


MESSAGE ACCESS AGENT: POP AND IMAP

The first and the second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a push protocol; it pushes the message from the client to the server. In other words, the direction of the bulk data (messages) is from the client to the server. On the other

hand, the third stage needs a pull protocol; the client must pull messages from the server. The direction of the bulk data is from the server to the client. The third stage uses a message access agent.

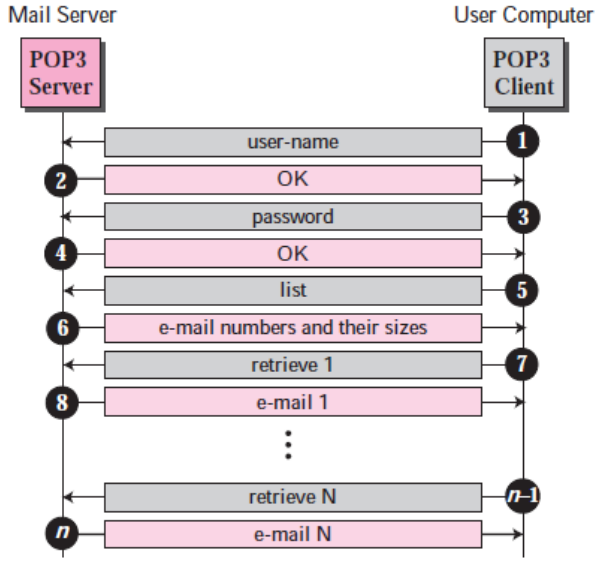
Currently two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4).



POP3

Post Office Protocol, version 3 (POP3) is simple and limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

Mail access starts with the client when the user needs to download its e-mail from the mailbox on the mail server. 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 messages, one by one.



POP3 has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval.

IMAP4

Another mail access protocol is Internet Mail Access Protocol, version 4 (IMAP4). IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.

POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides the following extra functions:

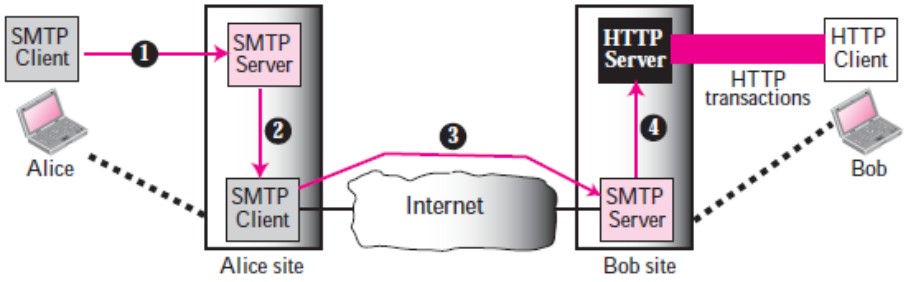
- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- 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.

WEB-BASED MAIL

E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Three common sites are Hotmail, Yahoo, and Google.

Case I

In the first case, Alice, the sender, uses a traditional mail server; Bob, the receiver, has an account on a Web-based server. Mail transfer from Alice's browser to her mail server is done through SMTP. The message from the receiving server to Bob's browser is done through HTTP. In other words, instead of using POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a request HTTP message to the website. The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in name and password match, the list of e-mails is transferred from the Web server to Bob's browser in HTML format. Now Bob can browse through his received e-mails and then, using more HTTP transactions, can get his e-mails one by one.



Case II

In the second case, both Alice and Bob use Web servers, but not necessarily the same server. Alice sends the message to the Web server using HTTP transactions. Alice sends an HTTP request message to her Web server using the name and address of Bob's mailbox as the URL. The server at the Alice site passes the message to the SMTP client and sends it to the server at the Bob site using SMTP protocol. Bob receives the message using HTTP transactions. However, the message from the server at the Alice site to the server at the Bob site still takes place using SMTP protocol.