


Computer Network and Network Design

Unit II

Application layer and Presentation layer - Part 1

Application Layer

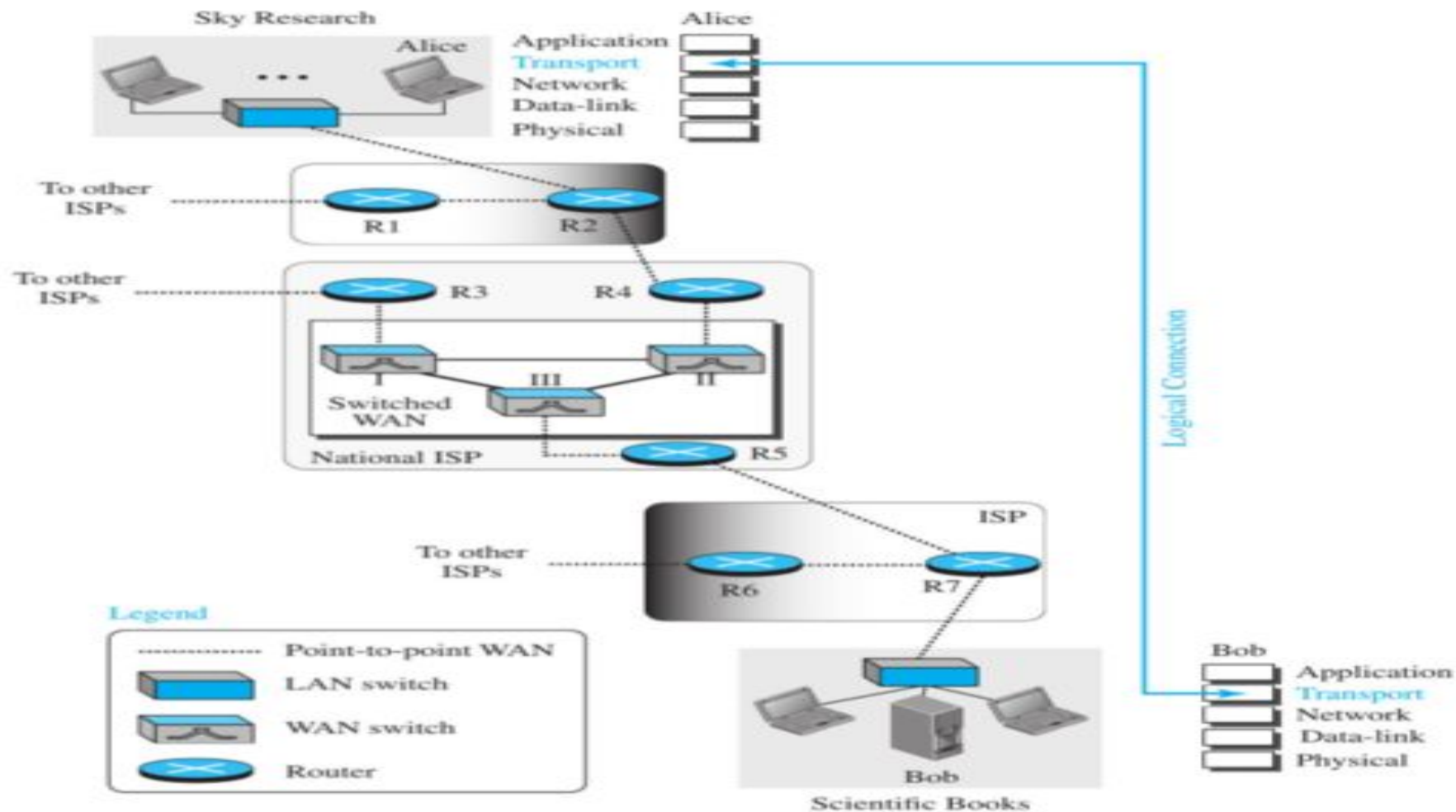
- Introduction
 - Providing Services,
 - Application layer Paradigms: Client- Server Paradigm, Peer to peer Paradigm
 - Application Programming Interface
 - Using Services of the Transport Layer
 - World Wide Web and HTTP
 - FTP
 - Electronic Mail
 - TELNET
 - Secure Shell (SSH)
 - Domain Name System (DNS)
- 

Application Layer:

INTRODUCTION

- The application layer provides services to the user.
- Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.
- Figure 2.1 shows the idea behind this logical connection.





INTRODUCTION

- We call the first host Alice and the second one Bob.
- The communication at the application layer is logical, not physical. Alice and Bob assume that there is a two-way logical channel between them through which they can send and receive messages.
- The actual communication, however, takes place through several devices (Alice, R2, R4, R5, R7, and Bob) and several physical channels as shown in the figure.




Providing Services

- The application layer is the only layer that provides services to the Internet user.
- The flexibility of the application layer, allows new application protocols to be easily added to the Internet, which has been occurring during the lifetime of the Internet.
- When the Internet was created, only a few application protocols were available to the users; but now new ones are being added constantly.
- **Standard and Nonstandard Protocols**

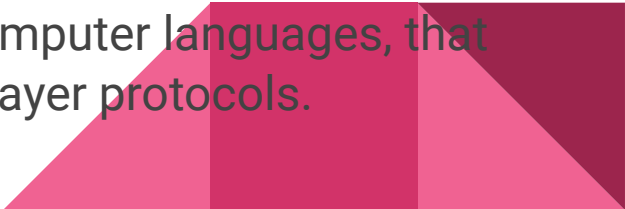
To be flexible, however, the application-layer protocols can be both standard and nonstandard.



Standard Application-Layer Protocols

- There are several application-layer protocols that have been **standardized and documented by the Internet authority**, and we are using them in our daily interaction with the Internet.
 - Each standard protocol is a pair of computer programs that interact with the user and the transport layer to provide a specific service to the user.
 - The study of these protocols enables a network manager to easily solve the problems that may occur when using these protocols.
 - The deep understanding of how these protocols work will also give us some ideas about how to create new nonstandard protocols.
- 

Nonstandard Application-Layer Protocols

- A programmer can create a nonstandard application-layer program if they can write two programs that provide service to the user by interacting with the transport layer.
 - The creation of a nonstandard (proprietary) protocol that does not even need the approval of the Internet authorities if privately used.
 - A private company can create a new customized application protocol to communicate with all of its offices around the world using the service provided by the first four layers of the TCP/IP protocol suite without using any of the standard application programs.
 - What is needed is to write programs, in one of the computer languages, that use the available services provided by the transport-layer protocols.
- 

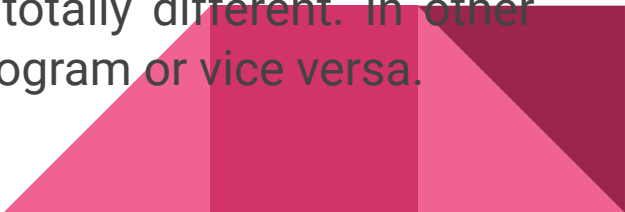
Application-Layer Paradigms

The client-server paradigm

The peer-to-peer paradigm



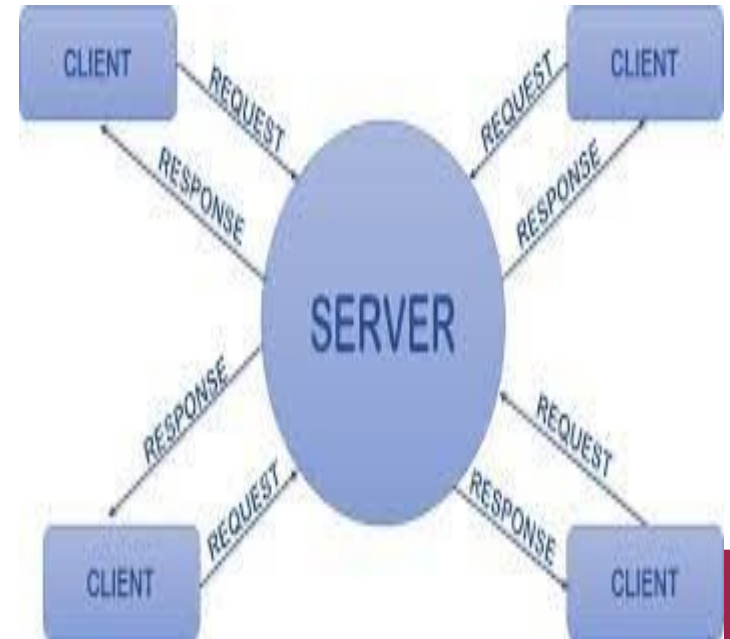
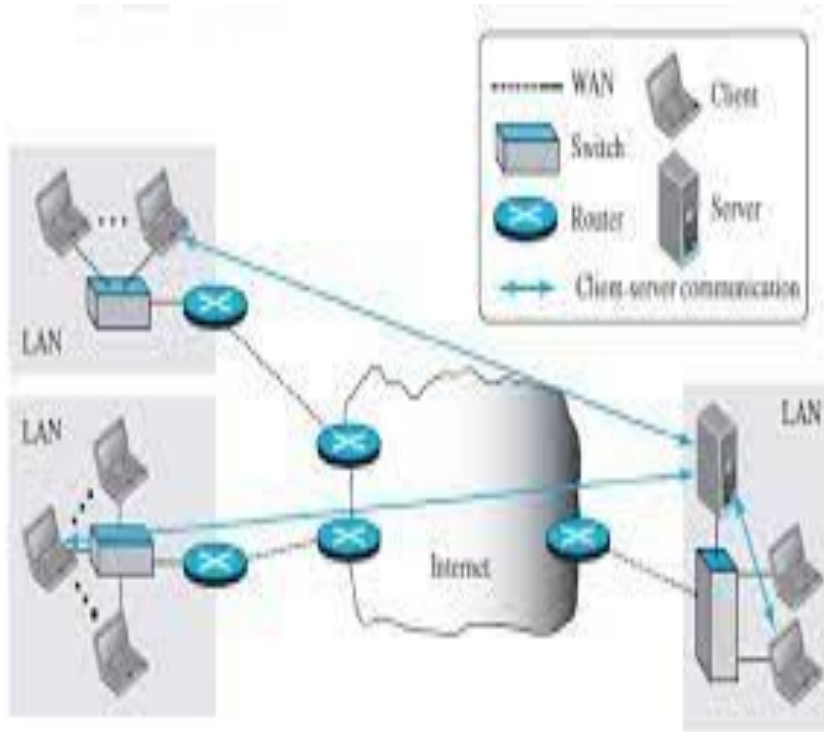
Traditional Paradigm: Client-Server

- The traditional paradigm is called the client-server paradigm.
 - In this paradigm, the service provider is an application program, called the server process; it runs continuously, waiting for another application program, called the client process, to make a connection through the Internet and ask for service.
 - There are normally some server processes that can provide a specific type of service, but there are many clients that request service from any of these server processes.
 - The server process must be running all the time; the client process is started when the client needs to receive service.
 - Although the communication in the client-server paradigm is between two application programs, the role of each program is totally different. In other words, we cannot run a client program as a server program or vice versa.
- 

Traditional Paradigm: Client-Server

- One problem with this paradigm is that the concentration of the communication load is on the shoulder of the server, which means the server should be a powerful computer. Even a powerful computer may become overwhelmed if a large number of clients try to connect to the server at the same time.
- Another problem is that there should be a service provider willing to accept the cost and create a powerful server for a specific service, which means the service must always return some type of income for the server in order to encourage such an arrangement.
- Several traditional services are still using this paradigm, including the World Wide Web (WWW) and its vehicle HyperText Transfer Protocol (HTTP), file transfer protocol (FTP), secure shell (SSH), e-mail, and so on.

Traditional Paradigm: Client-Server

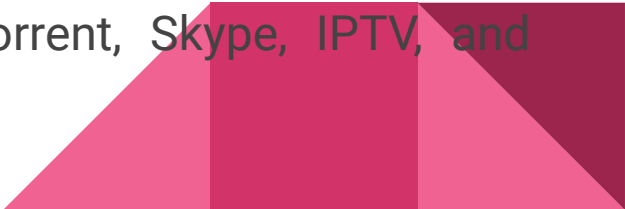


New Paradigm: Peer-to-Peer

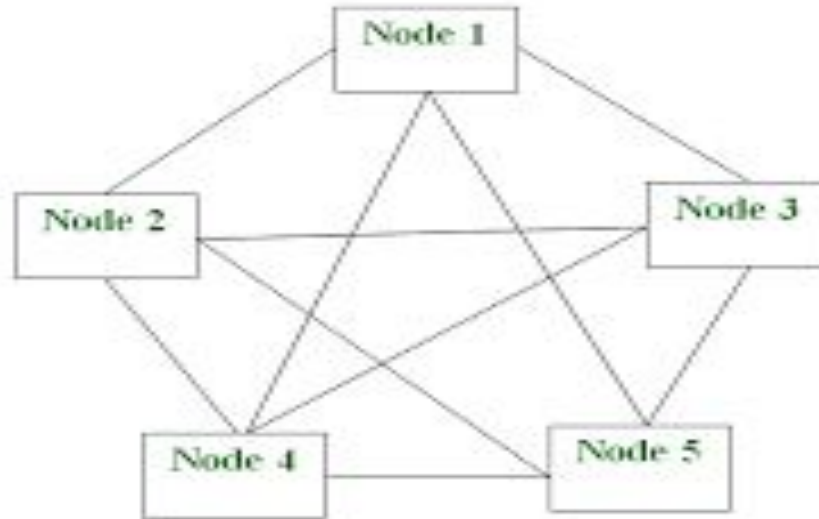
- A new paradigm, called the peer-to-peer paradigm (often abbreviated P2P paradigm) has emerged to respond to the needs of some new applications.
- In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect.
- The responsibility is shared between peers. A computer connected to the Internet can provide service at one time and receive service at another time.
- A computer can even provide and receive services at the same time.



New Paradigm: Peer-to-Peer

- For example, if an Internet user has a file available to share with other Internet users, there is no need for the file holder to become a server and run a server process all the time waiting for other users to connect and retrieve the file.
 - Although the peer-to-peer paradigm has been proved to be easily scalable and cost-effective in eliminating the need for expensive servers to be running and maintained all the time.
 - The main challenge has been security; it is more difficult to create secure communication between distributed services than between those controlled by some dedicated servers.
 - There are some new applications, such as BitTorrent, Skype, IPTV, and Internet telephony, that use this paradigm.
- 

New Paradigm: Peer-to-Peer



P2P Architecture

Application Programming Interface

- How can a client process communicate with a server process?
- A computer program is normally written in a computer language with a predefined set of instructions that tells the computer what to do. A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/ output access, and so on.
- If we need a process to be able to communicate with another process, we need a new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection.
- **A set of instructions of this kind is normally referred to as Application Programming Interface (API).**



Application Programming Interface

- An interface in programming is a set of instructions between two entities.
- In this case, one of the entities is the process at the application layer and the other is the operating system that encapsulates the first four layers of the TCP/IP protocol suite.
- In other words, a computer manufacturer needs to build the first four layers of the suite in the operating system and include an API.
- In this way, the processes running at the application layer are able to communicate with the operating system when sending and receiving messages through the Internet.
- Several APIs have been designed for communication. Three among them are common: socket interface, Transport Layer Interface (TLI), and STREAM.

Application Programming Interface

- Socket interface started in the early 1980s at UC Berkeley as part of a UNIX environment.
- The socket interface is a set of instructions that provide communication between the application layer and the operating system.
- It is a set of instructions that can be used by a process to communicate with another process.
- The idea of sockets allows us to use the set of all instructions already designed in a programming language for other sources and sinks.
- For example, in most computer languages, like C, C++, or Java, we have several instructions that can read and write data to other sources and sinks such as a keyboard (a source), a monitor (a sink), or a file (source and sink).
- We can use the same instructions to read from or write to sockets. In other words, we are adding only new sources and sinks to the programming language without changing the way we send data or receive data.

Figure 2.4 *Position of the socket interface*

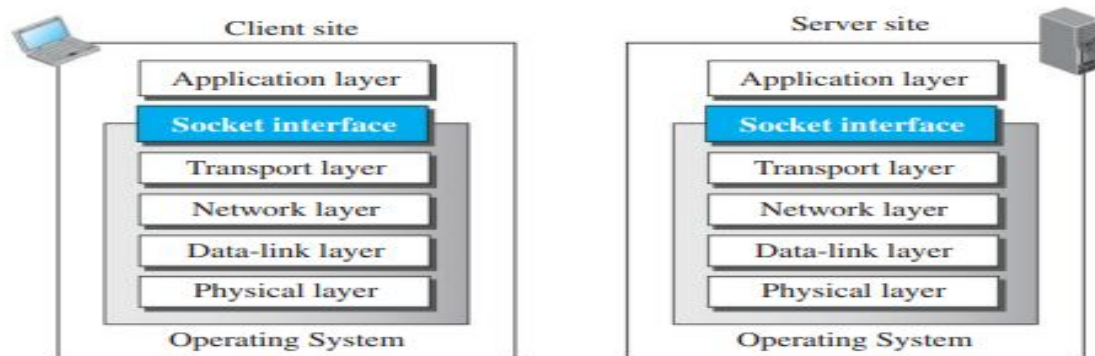
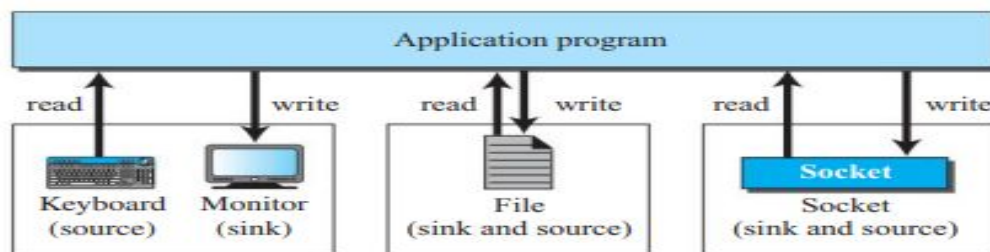
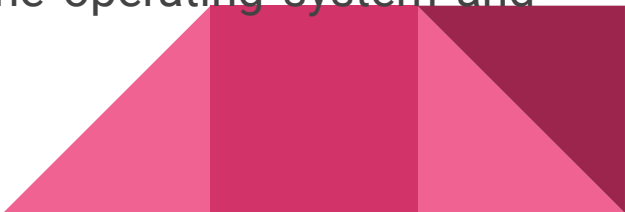


Figure 2.5 *Sockets used the same way as other sources and sinks*

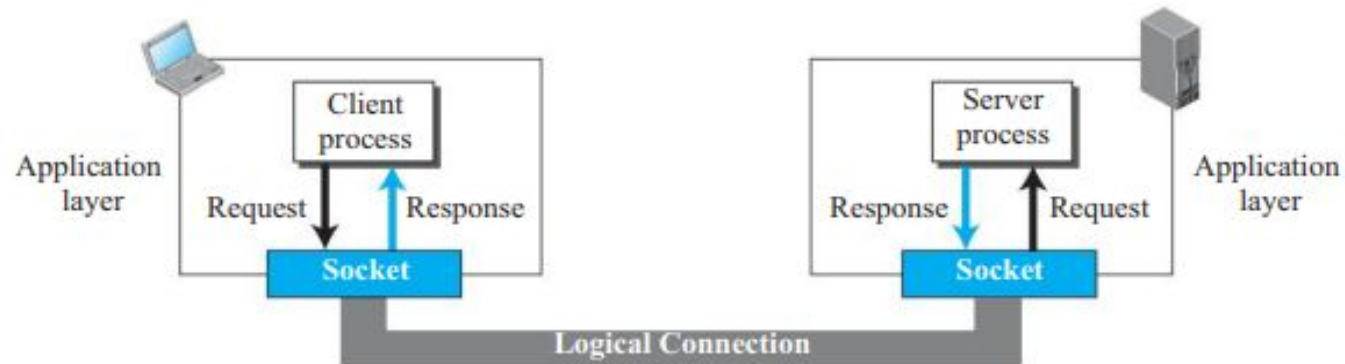


Sockets :

- Although a socket is supposed to behave like a terminal or a file, it is not a physical entity like them; it is an abstraction.
 - It is a data structure that is created and used by the application program.
 - We can say that, as far as the application layer is concerned, communication between a client process and server process is communication between two sockets, created at two ends, as shown in Figure 2.6.
 - The client thinks that the socket is the entity that receives the request and gives the response; the server thinks that the socket is the one that has a request and needs the response.
 - If we create two sockets, one at each end, and define the source and destination addresses correctly, we can use the available instructions to send and receive data. The rest is the responsibility of the operating system and the embedded TCP/IP protocol.
- 

Sockets :


Figure 2.6 *Use of sockets in process-to-process communication*



Socket Addresses

- The interaction between a client and a server is two-way communication.
- In a two-way communication, we need a pair of addresses: local (sender) and remote (receiver).
- The local address in one direction is the remote address in the other direction and vice versa.
- Since communication in the client-server paradigm is between two sockets, we need a pair of socket addresses for communication: a local socket address and a remote socket address.
- However, we need to define a socket address in terms of identifiers used in the TCP/IP protocol suite.
- A socket address should first define the computer on which a client or a server is running.
-

Socket Addresses

- A computer in the Internet is uniquely defined by its IP address, a 32-bit integer in the current Internet version.
 - However, several client or server processes may be running at the same time on a computer, which means that we need another identifier to define the specific client or server involved in the communication.
 - An application program can be defined by a port number, a 16-bit integer.
 - This means that a socket address should be a combination of an IP address and a port number as shown in Figure 2.7.
 - Since a socket defines the end-point of the communication, we can say that a socket is identified by a pair of socket addresses, a local and a remote.
- 

Socket Addresses

Figure 2.7 *A socket address*



Finding Socket Addresses

How can a client or a server find a pair of socket addresses for communication?
The situation is different for each site.

Server Site

Client Site



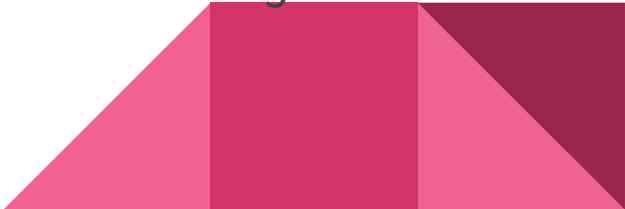
Finding Socket Addresses

Server Site

- The server needs a local (server) and a remote (client) socket address for communication.
- **Local Socket Address** The local (server) socket address is provided by the operating system. The operating system knows the IP address of the computer on which the server process is running.
- The port number of a server process, however, needs to be assigned. If the server process is a standard one defined by the Internet authority, a port number is already assigned to it.
- For example, the assigned port number for a Hypertext Transfer Protocol (HTTP) is the integer 80, which cannot be used by any other process.
- If the server process is not standard, the designer of the server process can choose a port number, in the range defined by the Internet authority, and assign it to the process.
- When a server starts running, it knows the local socket address.

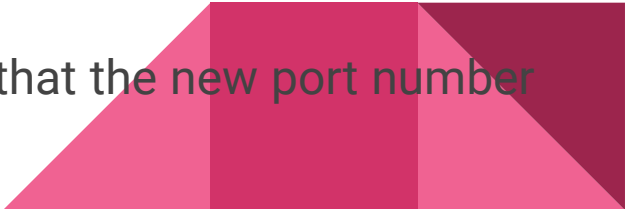


Finding Socket Addresses

- **Remote Socket Address** The remote socket address for a server is the socket address of the client that makes the connection.
 - Since the server can serve many clients, it does not know beforehand the remote socket address for communication.
 - The server can find this socket address when a client tries to connect to the server.
 - The client socket address, which is contained in the request packet sent to the server, becomes the remote socket address that is used for responding to the client.
 - In other words, although the local socket address for a server is fixed and used during its lifetime, the remote socket address is changed in each interaction with a different client.
- 

Finding Socket Addresses

Client Site

- The client also needs a local (client) and a remote (server) socket address for communication.
 - **Local Socket Address** The local (client) socket address is also provided by the operating system.
 - The operating system knows the IP address of the computer on which the client is running.
 - The port number, however, is a 16-bit temporary integer that is assigned to a client process each time the process needs to start the communication.
 - The port number, however, needs to be assigned from a set of integers defined by the Internet authority and called the ephemeral (temporary) port numbers.
 - The operating system, however, needs to guarantee that the new port number is not used by any other running client process.
- 

Finding Socket Addresses

Remote Socket Address

- When a client process starts, it should know the socket address of the server it wants to connect to.
- We will have two situations in this case.
- Sometimes, the user who starts the client process knows both the server port number and IP address of the computer on which the server is running.
- This usually occurs in situations when we have written client and server applications and we want to test them.
- In this situation, the programmer can provide these two pieces of information when it runs the client program.



Remote Socket Address

- Although each standard application has a well-known port number, most of the time, we do not know the IP address.
- This happens in situations such as when we need to contact a Web page, send an e-mail to a friend, copy a file from a remote site, and so on.
- In these situations, the server has a name, an identifier that uniquely defines the server process.
- Examples of these identifiers are URLs, such as `www.xxx.yyy`, or e-mail addresses, such as `xxxx@yyyy.com`.
- The client process should now change this identifier (name) to the corresponding server socket address.
- The client process normally knows the port number because it should be a well-known port number, but the IP address can be obtained using another clientserver application called the Domain Name System (DNS).



Using Services of the Transport Layer

- A pair of processes provide services to the users of the Internet, human or programs.
- A pair of processes, however, need to use the services provided by the transport layer for
- communication because there is no physical communication at the application layer.
- There are three common transport layer protocols in the TCP/IP suite: UDP, TCP, and SCTP.
- Most standard applications have been designed to use the services of one of these protocols. When we write a new application, we can decide which protocol we want to use.
- The choice of the transport layer protocol seriously affects the capability of the application processes.



UDP Protocol

- UDP provides connectionless, unreliable, datagram service.
- Connectionless service means that there is no logical connection between the two ends exchanging messages.
- Each message is an independent entity encapsulated in a packet called a datagram.
- UDP does not see any relation (connection) between consequent datagrams coming from the same source and going to the same destination.
- UDP is not a reliable protocol. Although it may check that the data is not corrupted during the transmission, it does not ask the sender to resend the corrupted or lost datagram.
- For some applications, UDP has an advantage: it is message-oriented. It gives boundaries to the messages exchanged.
- An application program may be designed to use UDP if it is sending small messages and the simplicity and speed is more important for the application than reliability.
- For example, some management and multimedia applications fit in this category.

TCP Protocol

- TCP provides connection-oriented, reliable, byte-stream service.
- TCP requires that two ends first create a logical connection between themselves by exchanging some connection-establishment packets. This phase, which is sometimes called handshaking, establishes some parameters between the two ends including the size of the data packets to be exchanged, the size of buffers to be used for holding the chunks of data until the whole message arrives, and so on.
- After the handshaking process, the two ends can send chunks of data in segments in each direction. By numbering the bytes exchanged, the continuity of the bytes can be checked. For example, if some bytes are lost or corrupted, the receiver can request the resending of those bytes, which makes TCP a reliable protocol.
- TCP also can provide flow control and congestion control,.
- One problem with the TCP protocol is that it is not message-oriented; it does not put boundaries on the messages exchanged.
- Most of the standard applications that need to send long messages and require reliability may benefit from the service of the TCP.




SCTP Protocol

- SCTP provides a service which is a combination of the two other protocols.
- Like TCP, SCTP provides a connection-oriented, reliable service, but it is not byte-stream oriented.
- It is a message-oriented protocol like UDP. In addition, SCTP can provide multistream service by providing multiple network-layer connections.
- SCTP is normally suitable for any application that needs reliability and at the same
- time needs to remain connected, even if a failure occurs in one network-layer
- connection.

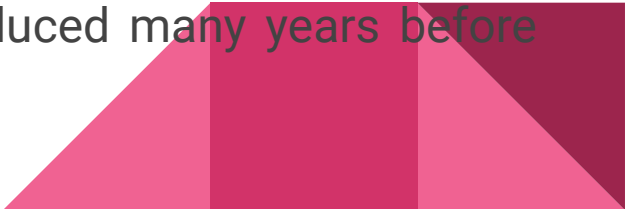


STANDARD CLIENT-SERVER APPLICATIONS

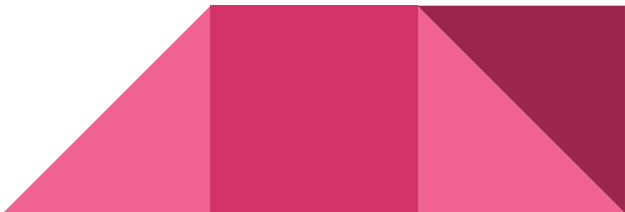
- We have selected six standard application programs in this section.
 - We start with **HTTP** and the **World Wide Web** because they are used by almost all Internet users.
 - We then introduce **file transfer** and **electronic mail** applications which have high traffic loads on the Internet.
 - Next, we explain remote logging and how it can be achieved using the **TELNET** and **SSH** protocols.
 - Finally, we discuss **DNS**, which is used by all application programs to map the application layer identifier to the corresponding host IP address.
 - Some other applications, such as Dynamic Host Configuration Protocol (**DHCP**) or Simple Network Management Protocol (**SNMP**).
- 

World Wide Web and HTTP

World Wide Web

- The Web today is a repository of information in which the documents, called **Web pages**, are distributed all over the world and related documents are linked together.
 - The popularity and growth of the Web can be related to two terms in the above statement: **distributed and linked**.
 - **Distribution** allows the growth of the Web. Each web server in the world can add a new web page to the repository and announce it to all Internet users without overloading a few servers.
 - **Linking** allows one web page to refer to another web page stored in another server somewhere else in the world. The linking of web pages was achieved using a concept called **hypertext**, which was introduced many years before the advent of the Internet.
- 

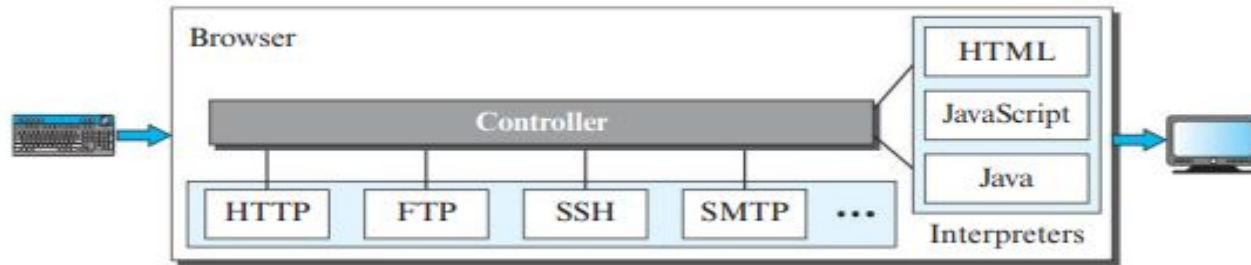
Architecture

- 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 links 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.
- 


Web Client (Browser)

- A variety of vendors offer commercial browsers that interpret and display a web page, and all of them use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocols, and interpreters. (see Figure 2.9).

Figure 2.9 Browser



Web Client (Browser)

- The **controller** receives input from the keyboard or the mouse and uses the client programs to access the document.
 - After the document has been accessed, the controller uses one of the **interpreters** to display the document on the screen.
 - The **client protocol** can be one of the protocols described later, such as HTTP or FTP.
 - The interpreter can be HTML, Java, or JavaScript, depending on the type of document.
 - Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.
- 

Web Server

- The web page is stored at the server.
- Each time a 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. Some popular web servers include Apache and Microsoft Internet Information Server.



Uniform Resource Locator (URL)

- A web page, as a file, needs to have a unique identifier to distinguish it from other web pages.
- To define a web page, we need three identifiers: **host, port, and path**.
- However, before defining the web page, we need to tell the browser what client server application we want to use, which is called the **protocol**.
- This means we need four identifiers to define the web page. The first is the type of vehicle to be used to fetch the web page; the last three make up the combination that defines the destination object (web page).



Uniform Resource Locator (URL)

Protocol. The first identifier is the abbreviation for the client-server program that we need in order to access the web page. Although most of the time the protocol is HTTP (HyperText Transfer Protocol), and also use other protocols such as FTP (File Transfer Protocol).

Host. The host identifier can be the IP address of the server or the unique name given to the server. IP addresses can be defined in dotted decimal notations, (such as 64.23.56.17);

The name is normally the domain name that uniquely defines the host, such as forouzan.com.

Port. The port, a 16-bit integer, is normally predefined for the client-server application.

For example, if the HTTP protocol is used for accessing the web page, the well-known port number is 80. However, if a different port is used, the number can be explicitly given.

Path. The path identifies the location and the name of the file in the underlying operating system. The format of this identifier normally depends on the operating system.

In UNIX, a path is a set of directory names followed by the file name, all separated by a slash.



Uniform Resource Locator (URL)

- For example, /top/next/last/myfile is a path that uniquely defines a file named myfile, stored in the directory last, which itself is part of the directory next, which itself is under the directory top. In other words, the path lists the directories from the top to the bottom, followed by the file name.
- To combine these four pieces together, the uniform resource locator (URL) has been designed; it uses three different separators between the four pieces as shown below:
- protocol://host/path Used most of the time
- protocol://host:port/path Used when port number is needed



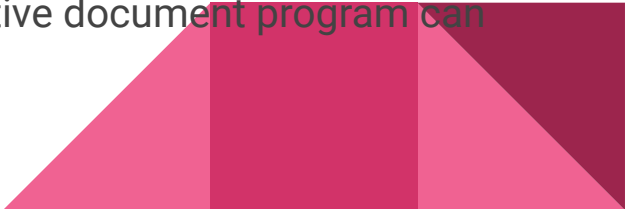
Web Documents

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Static. A static web document resides in a file that it is associated with a web server. The author of a static document determines the contents at the time the document is written. Because the contents do not change, each request for a static document results in exactly the same response.

Dynamic. A dynamic web document does not exist in a predefined form. When a request arrives the web server runs an application program that creates the document. The server returns the output of the program 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 can vary from one request to another.

Active An active web document consists of a computer program that the server sends to the browser and that the browser must run locally. When it runs, the active document program can interact with the user and change the display continuously.




HyperText Transfer Protocol (HTTP)

- The HyperText Transfer Protocol (HTTP) is a protocol that is used to define how the client-server programs can be written to retrieve web pages from the Web.
- An HTTP client sends a request; an HTTP server returns a response.
- The server uses the port number 80; the client uses a temporary port number.
- HTTP uses the services of TCP, which, is a connection-oriented and reliable protocol.
- This means that, before any transaction between the client and the server can take place, a connection needs to be established between them. After the transaction, the connection should be terminated.



Nonpersistent versus Persistent Connections

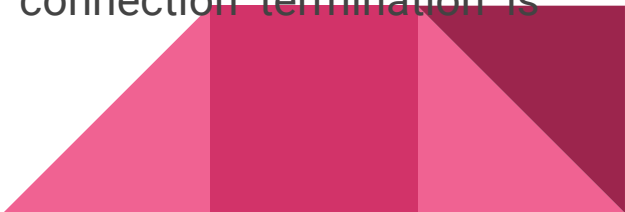
- The hypertext concept embedded in web page documents may require several requests and responses.
 - If the web pages, objects to be retrieved, are located on different servers, we do not have any other choice than to create a new TCP connection for retrieving each object.
 - However, if some of the objects are located on the same server, we have two choices:
 - to retrieve each object using a new TCP connection or to make a TCP connection and retrieve them all.
 - The first method is referred to as **nonpersistent connections**, the second as **persistent connections**.
- 

Nonpersistent Connections

- 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.



Persistent Connections

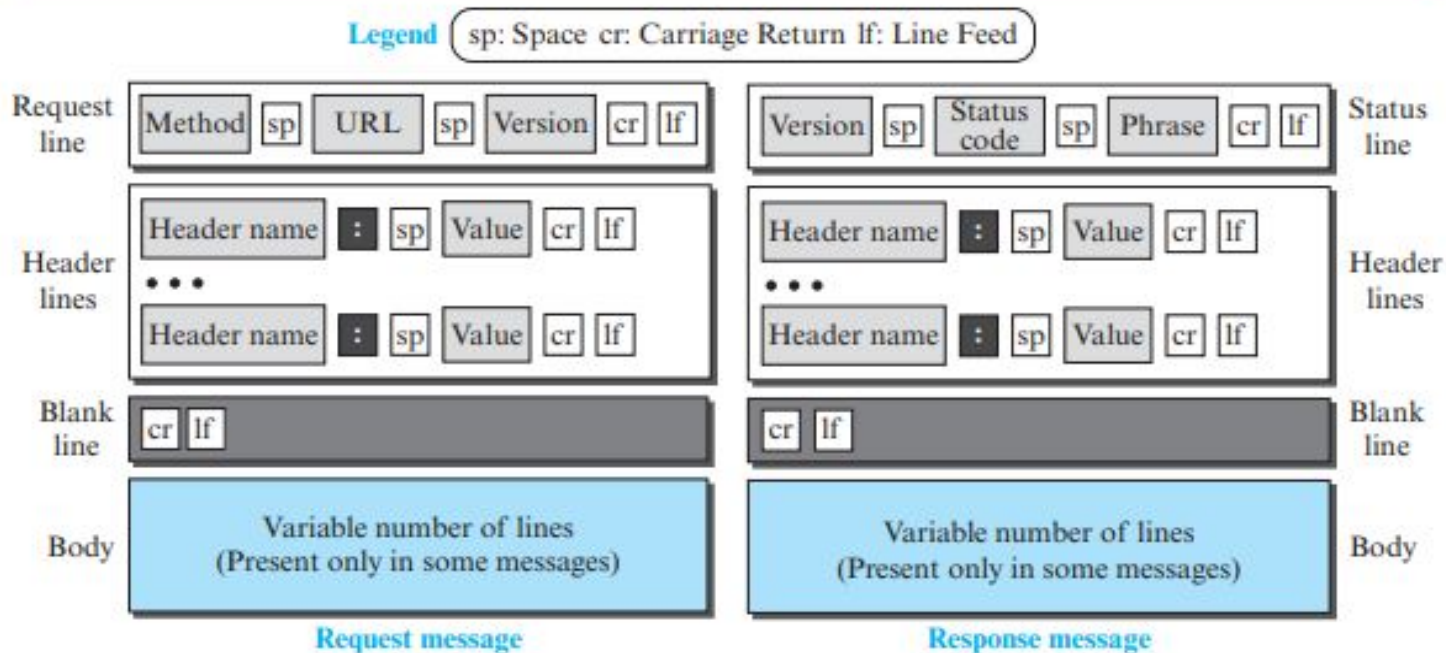
- HTTP version 1.1 specifies a persistent connection by default.
 - In a persistent connection, the server leaves the connection open for more requests after sending a response.
 - The server can close the connection at the request of a client or if a time-out has been reached.
 - The sender usually 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.
 - Time and resources are saved using persistent connections.
 - Only one set of buffers and variables needs to be set for the connection at each site.
 - The round trip time for connection establishment and connection termination is saved.
- 

Message Formats

- The HTTP protocol defines the format of the request and response messages, as shown in Figure 2.12.
- We have put the two formats next to each other for comparison.
- Each message is made of four sections.
- The first section in the request message is called the request line; the first section in the response message is called the status line.
- The other three sections have the same names in the request and response messages.
- However, the similarities between these sections are only in the names; they may have different contents.



Figure 2.12 *Formats of the request and response messages*



Request Message

- The first line in a request message is called a **request line**.
- There are three fields in this line separated by one space and terminated by two characters (carriage return and line feed) as shown in Figure 2.12.
- The fields are called **method**, **URL**, and **version**.
- The **method** field defines the request types.
- In version 1.1 of HTTP, several methods are defined, as shown in Table 2.1.
- The second field, **URL**, it defines the address and name of the corresponding web page.
- The third field, **version**, gives the version of the protocol; the most current version of HTTP is 1.1.



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.
- For example, the client can request to send the document in specific format.
- Each header line has a header name, a colon, a space, and a header value.
- Header names are given in table and value field defines the values associated with each header name.
- **The body** contains the document. Usually a comment or the file to be published on the website etc.



Request Message

Table 2.1 *Methods*

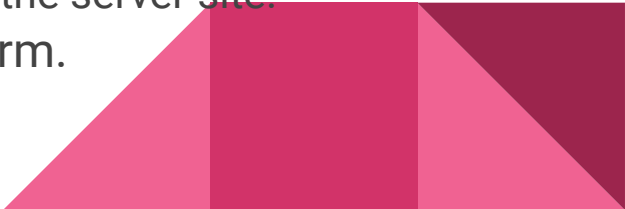
<i>Method</i>	<i>Action</i>
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

Request Message

Table 2.2 *Request Header Names*

<i>Header</i>	<i>Description</i>
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

Response Message

- A response message consists of a status line, header lines, a blank line, and sometimes a body.
 - 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.
 - 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.
 - The status **phrase** explains the status code in text form.
- 

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.
- For example, the sender can send extra information about the document.
- Each header line has a header name, a colon, a space, and a header value.
- **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.



Response Message

Table 2.3 *Response Header Names*

<i>Header</i>	<i>Description</i>
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

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 with the request to tell the server that it needs the page only if it is modified after a certain point in time.



Cookies

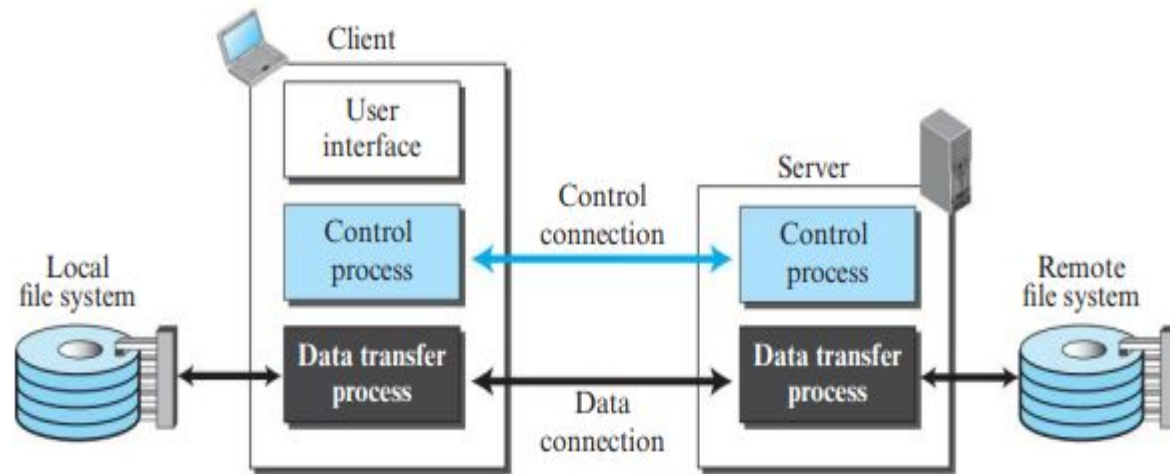


File Transfer Protocol (FTP)


- File Transfer Protocol (FTP) is the standard protocol 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 data.
 - Two systems may have different directory structures.
 - All of these problems have been solved by FTP.
 - Although we can transfer files using HTTP, FTP is a better choice to transfer large files or to transfer files using different formats. Figure 2.17 shows the basic model of FTP.
- 

File Transfer Protocol (FTP)

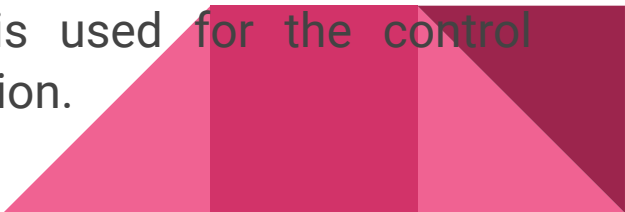
Figure 2.17 FTP



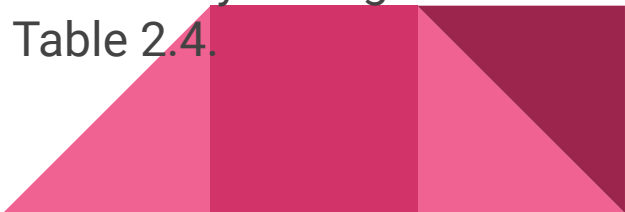
File Transfer Protocol (FTP)

- The client has three components: **user interface, client control process, and the client data transfer process.**
 - The server has two components: **the server control process and the server data transfer process.**
 - The **control connection** is made between the control processes.
 - The **data connection** is made between the data transfer processes.
 - 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.
- 

Lifetimes of Two Connections

- The two connections in FTP have different lifetimes.
 - **The control connection** remains connected during the entire interactive FTP session.
 - **The data connection** is opened and then closed for each file transfer activity.
 - 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.
 - FTP uses two well-known TCP ports: **port 21** is used for the control connection, and **port 20** is used for the data connection.
- 

Control Connection

- For control communication, FTP uses the same approach as TELNET.
 - It uses the NVT ASCII character set as used by TELNET.
 - Communication is achieved through commands and responses.
 - This simple method is adequate for the control connection because we send one command (or response) at a time.
 - Each line is terminated with a two-character (carriage return and line feed) end-of-line token.
 - During this control connection, commands are sent from the client to the server and responses are sent from the server to the client.
 - 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. Some of the most common commands are shown in Table 2.4.
- 

Data Connection

- The data connection uses the well-known port 20 at the server site.
- However, the creation of a data connection is different from the control connection.
- The following shows the steps:
 - 1.The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files.
 - 2. The client sends this port number to the server using the PORT command.
 - 3. The server receives the port number and issues an active open using the well known port 20 and the received ephemeral port number.



Communication over Data Connection

- The purpose and implementation of the data connection are different from those 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.**

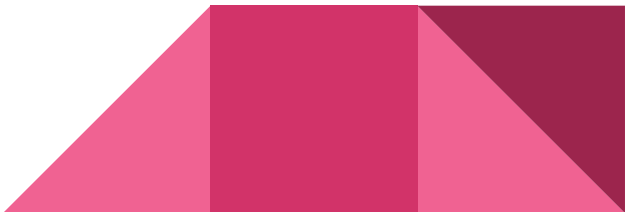


Communication over Data Connection

Data Structure

- FTP can transfer a file across the data connection using one of the following interpretations of the structure of the data: **file structure, record structure, or page structure**.
- The file structure format (used by default) has no structure. It is a continuous stream of bytes.
- In the record structure, the file is divided into records. This can be used only with text files.
- In the 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.

File Type

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

Communication over Data Connection

Transmission Mode

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



File Transfer

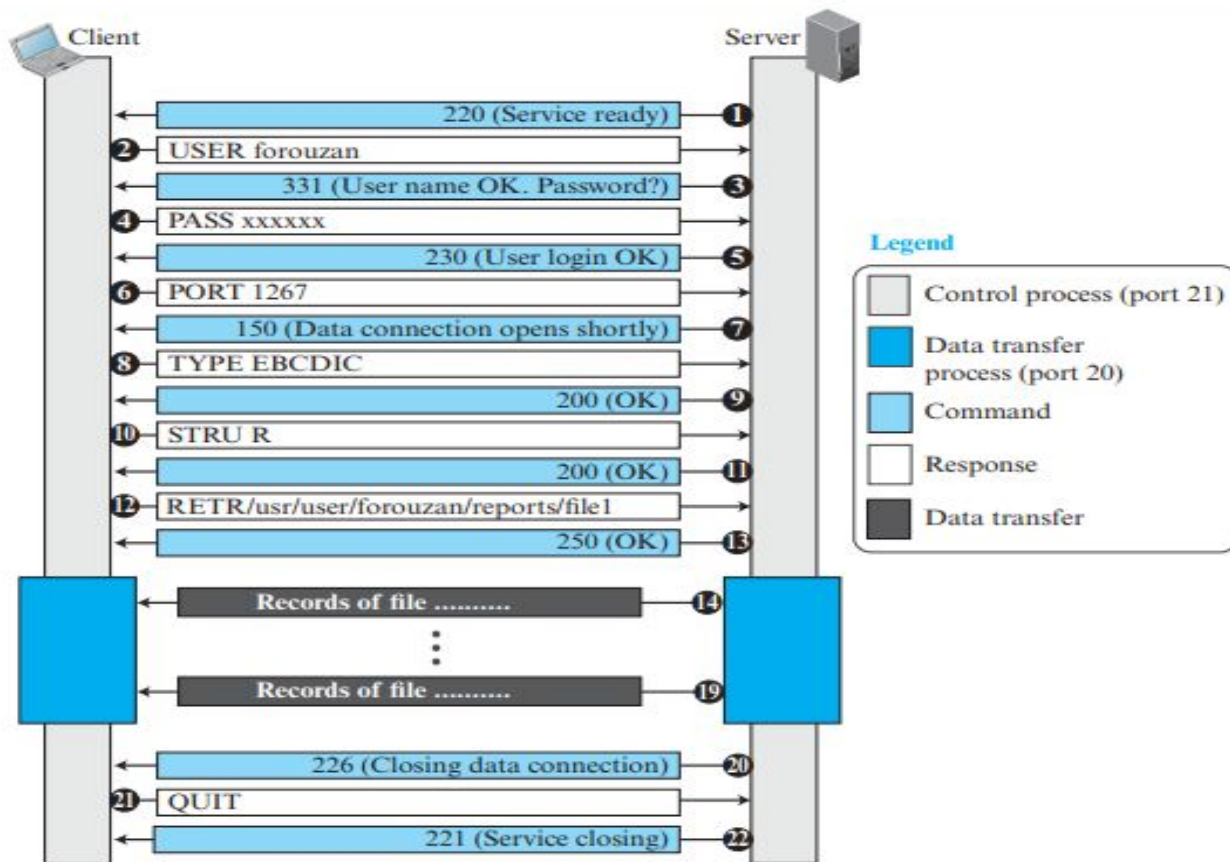
- File transfer occurs over the data connection under the control of the commands sent over the control connection.
- However, we should remember that file transfer in FTP means one of three things: **retrieving a file (server to client)**, **storing a file (client to server)**, and **directory listing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




Example 2.11

Figure 2.18 shows an example of using FTP for retrieving a file. The figure shows only one file to be transferred. The control connection remains open all the time, but the data connection is

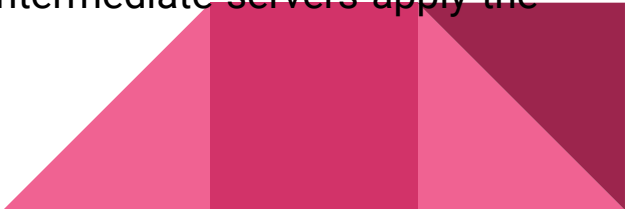
Figure 2.18 Example 2.11



File Transfer

- **Example-** Figure 2.18 shows an example of using FTP for retrieving a file.
 - The figure shows only one file to be transferred.
 - The control connection remains open all the time, but the data connection is opened and closed repeatedly.
 - We assume the file is transferred in six sections.
 - After all records have been transferred, the server control process announces that the file transfer is done.
 - Since the client control process has no file to retrieve, it issues the QUIT command, which causes the service connection to be closed.
- 

Electronic Mail

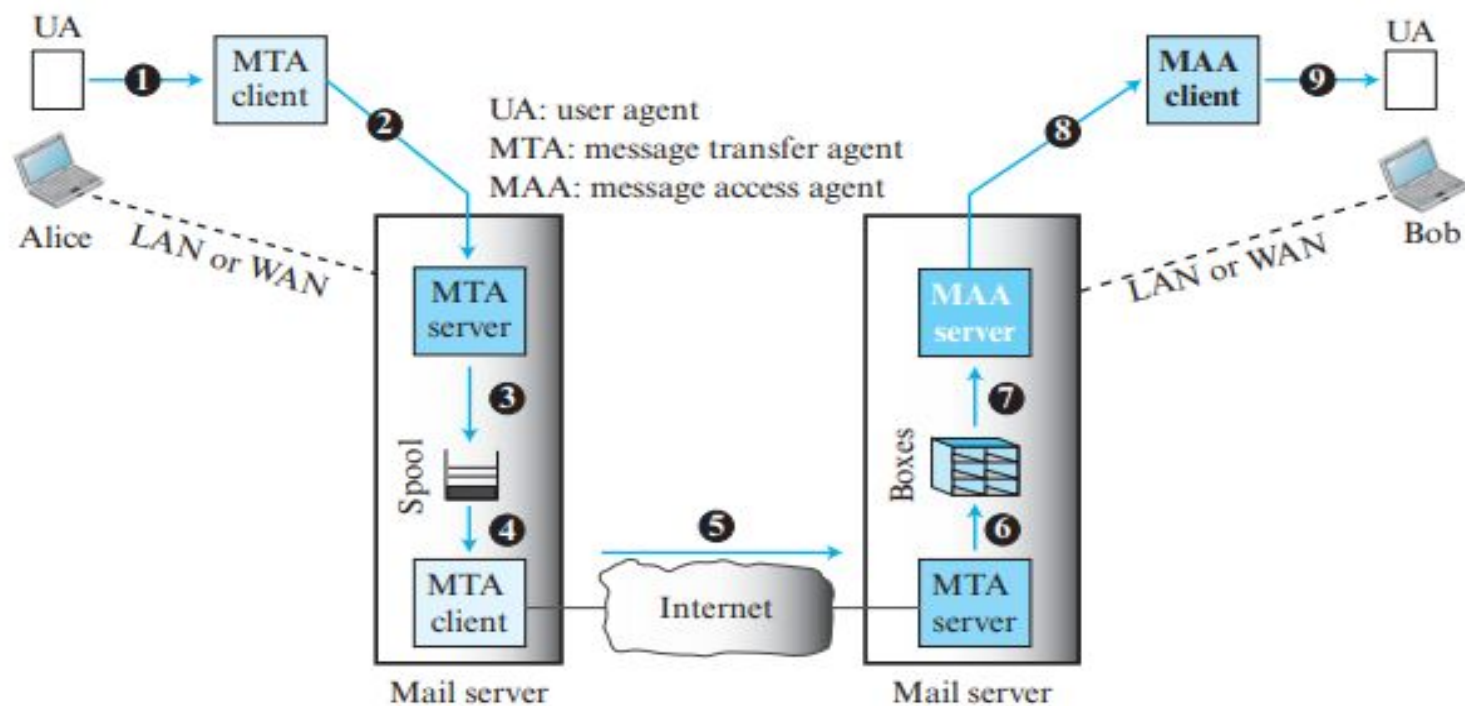
- **Electronic mail** (or e-mail) allows users to exchange messages.
 - In an application such as HTTP or FTP, the server program is running all the time, waiting for a request from a client. When the request arrives, the server provides the service. There is a request and there is a response.
 - In the case of electronic mail, the situation is different. First, e-mail is considered a **one-way transaction**.
 - When Alice sends an e-mail to Bob, she may expect a response, but this is not a mandate. Bob may or may not respond. If he does respond, it is another one-way transaction.
 - Second, it is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him. Bob may turn off his computer when he is not using it.
 - This means that the idea of client/ server programming should be implemented in another way: using some intermediate computers (servers).
 - The users run only client programs when they want and the intermediate servers apply the client/server paradigm.
- 

Architecture

- To explain the architecture of e-mail, we give a common scenario, as shown in Figure 2.19.
- Another possibility is the case in which Alice or Bob is directly connected to the corresponding mail server, in which LAN or WAN connection is not required.



Figure 2.19 *Common scenario*



Architecture

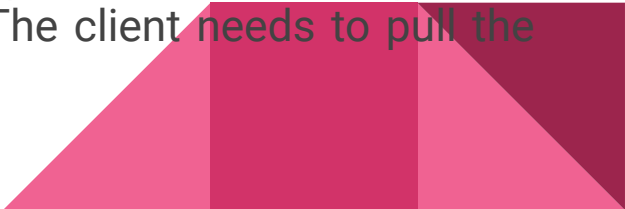
- In the common scenario, the sender and the receiver of the e-mail, Alice and Bob respectively, are connected via a LAN or a WAN to two mail servers.
- The administrator has created one **mailbox** for each user where the received messages are stored.
- **A mailbox** is part of a server hard drive, a special file with permission restrictions.
- Only the owner of the mailbox has access to it.
- The administrator has also created a queue (spool) to store messages waiting to be sent.



Architecture

- A simple e-mail from Alice to Bob takes nine different steps, as shown in the figure.
- Alice and Bob use three different agents: **a User Agent (UA), a Mail Transfer Agent (MTA), and a Message Access Agent (MAA).**
- When Alice needs to send a message to Bob, she runs a UA program to prepare the message and send it to her mail server.
- The mail server at her site uses a queue (spool) to store messages waiting to be sent.
- The message, however, needs to be sent through the Internet from Alice's site to Bob's site using an MTA.
- Here two message transfer agents are needed: **one client and one server.**
- Like most client-server programs on the Internet, **the server** needs to run all the time because it does not know when a client will ask for a connection.
- **The client**, on the other hand, can be triggered by the system when there is a message in the queue to be sent.

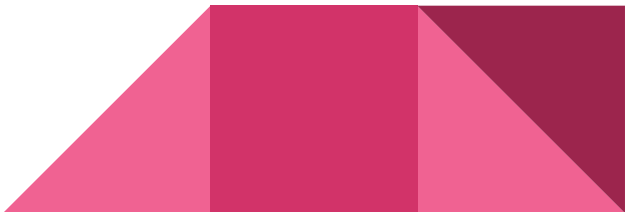
Architecture

- The user agent at the Bob site allows Bob to read the received message.
 - Bob later uses an MAA client to retrieve the message from an MAA server running on the second server.
 - There are two important points we need to emphasize here. 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.
 - This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN.
 - If he is connected through a WAN, he must keep the connection up all the time.
 - 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.
- 

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.
- 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.
- 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. A command-driven user agent normally accepts a one character command from the keyboard to perform its task.
- For example, a user can type the character r, at the command prompt, to reply to the sender of the message, or type the character R to reply to the sender and all recipients. 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.
- They have graphical components such as icons, menu bars, and windows that make the services easy to access. Some examples of GUI-based user agents are Eudora and Outlook

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** (see Figure 2.20).
 - **The envelope** usually contains the sender address, the receiver address, and other information.
 - **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.

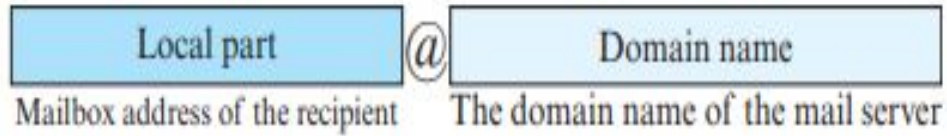


Addresses

- 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 (see Figure 2.21).
 - The local part defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.
 - 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).
- 

Addresses

Figure 2.21 *E-mail address*



Mailing List or Group List

- Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list.
- Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA.



User Agent

Message Transfer Agent: SMTP

Message Access Agent: POP and IMAP



Message Transfer Agent: SMTP

- The formal protocol that defines the MTA client and server in the Internet is called Simple Mail Transfer Protocol (SMTP).
- SMTP is used two times, between the sender and the sender's mail server and between the two mail servers.
- SMTP simply defines how commands and responses must be sent back and forth.



Message Transfer Agent: SMTP

Commands and Responses

Commands

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

Keyword: argument(s)

- It consists of a keyword followed by zero or more arguments.
- SMTP defines 14 commands, listed in Table 2.6.

Table 2.6 SMTP Commands

<i>Keyword</i>	<i>Argument(s)</i>	<i>Description</i>
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message

Message Transfer Agent: SMTP

Responses


- Responses are sent from the server to the client.
- A response is a three digit code that may be followed by additional textual information.
- Table 2.7 shows the most common response types.

Table 2.7 *Responses*

<i>Code</i>	<i>Description</i>
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

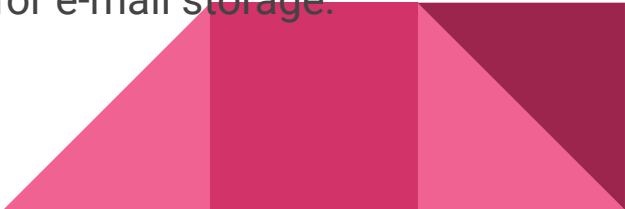
Message Access Agent: POP and IMAP

POP3


- **Post Office Protocol, version 3 (POP3)** is simple but 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 username and password to access the mailbox.
 - The user can then list and retrieve the mail messages, one by one.
- 

Message Access Agent: POP and IMAP

IMAP4

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

TELNET

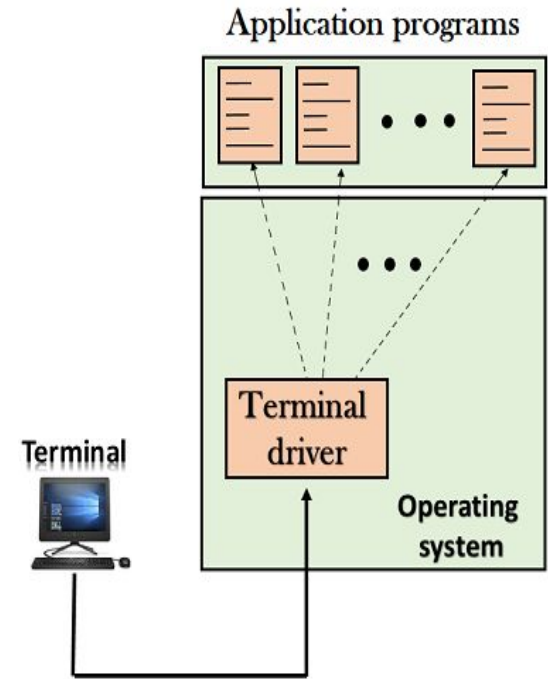
- The main task of the internet is to provide services to users. For example, users want to run different application programs at the remote site and transfers a result to the local site. This requires a client-server program such as FTP, SMTP. But this would not allow us to create a specific program for each demand.
 - The better solution is to provide a general client-server program that lets the user access any application program on a remote computer. Therefore, a program that allows a user to log on to a remote computer. A popular client-server program Telnet is used to meet such demands. Telnet is an abbreviation for **Terminal Network**.
 - Telnet provides a connection to the remote computer in such a way that a local terminal appears to be at the remote side.
- 

TELNET

There are two types of login:

Local Login

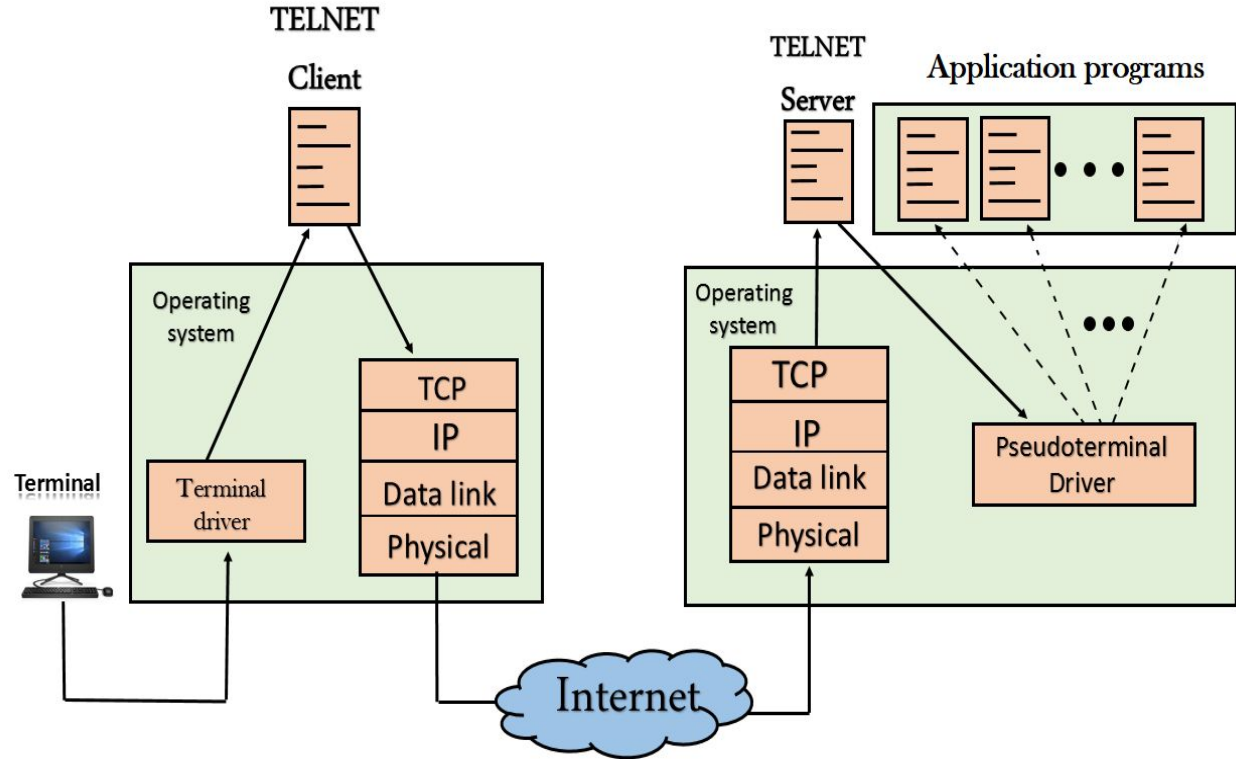
- When a user logs into a local computer, then it is known as local login.
- When the workstation running terminal emulator, the keystrokes entered by the user are accepted by the terminal driver. The terminal driver then passes these characters to the operating system which in turn, invokes the desired application program.
- However, the operating system has special meaning to special characters. For example, in UNIX some combination of characters have special meanings such as control character with "z" means suspend. Such situations do not create any problem as the terminal driver knows the meaning of such characters. But, it can cause the problems in remote login.



TELNET

Remote login

When the user wants to access an application program on a remote computer, then the user must perform remote login.



TELNET

How remote login occurs At the local site

- The user sends the keystrokes to the terminal driver, the characters are then sent to the TELNET client.
- The TELNET client which in turn, transforms the characters to a universal character set known as network virtual terminal characters and delivers them to the local TCP/IP stack.

At the remote site

- The commands in NVT forms are transmitted to the TCP/IP at the remote machine.
- Here, the characters are delivered to the operating system and then pass to the TELNET server.
- The TELNET server transforms the characters which can be understandable by a remote computer.
- However, the characters cannot be directly passed to the operating system as a remote operating system does not receive the characters from the TELNET server.
- Therefore it requires some piece of software that can accept the characters from the TELNET server.
- The operating system then passes these characters to the appropriate application program.

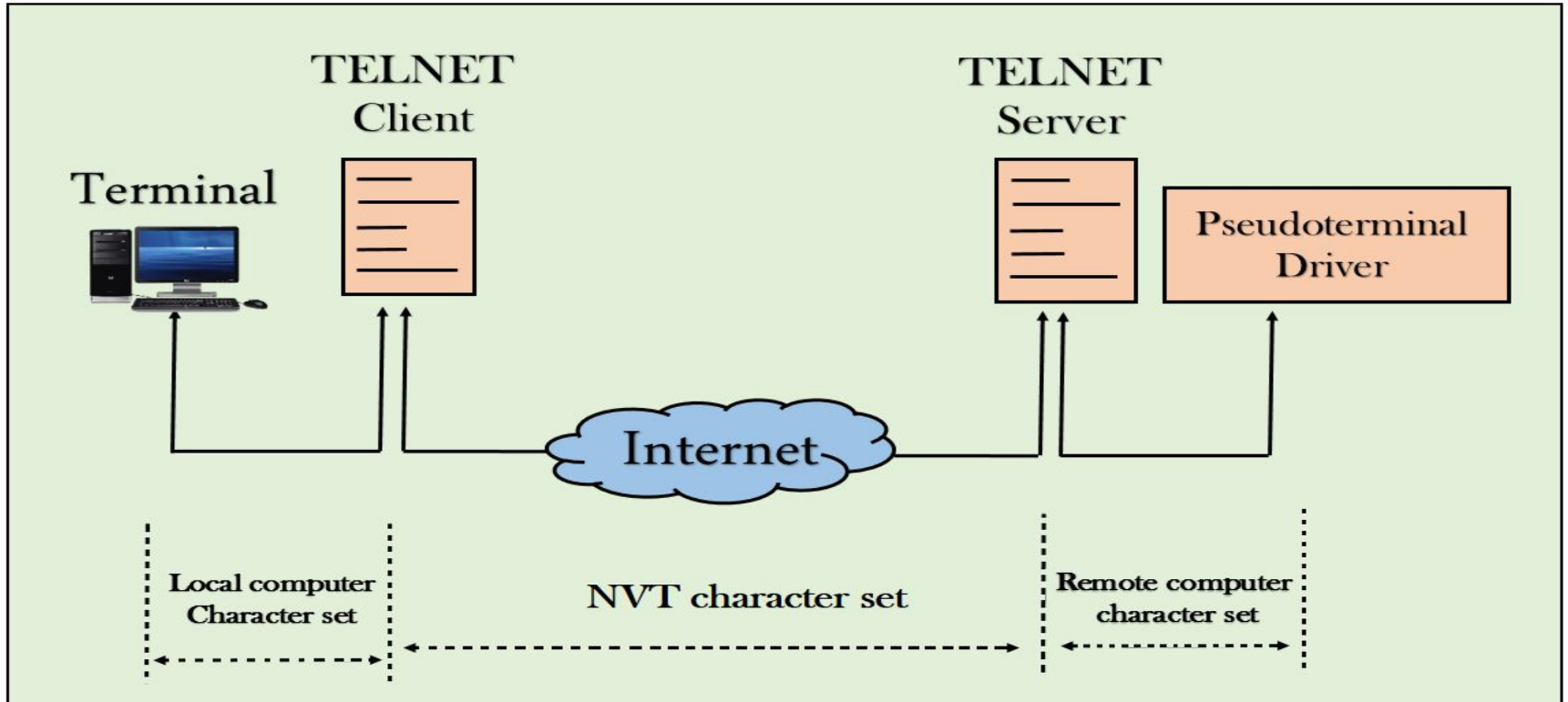
TELNET

Network Virtual Terminal (NVT)


- The network virtual terminal is an interface that defines how data and commands are sent across the network.
- In today's world, systems are heterogeneous. For example, the operating system accepts a special combination of characters such as end-of-file token running a DOS operating system *ctrl+z* while the token running a UNIX operating system is *ctrl+d*.
- TELNET solves this issue by defining a universal interface known as network virtual interface.
- The TELNET client translates the characters that come from the local terminal into NVT form and then delivers them to the network. The Telnet server then translates the data from NVT form into a form which can be understandable by a remote computer.



TELNET



Secure Shell (SSH)

- SSH stands for **Secure Shell or Secure Socket Shell**.
 - It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet.
 - It is used to login to a remote server to execute commands and data transfer from one machine to another machine.
 - The SSH protocol was developed by SSH communication security Ltd to safely communicate with the remote machine.
 - Secure communication provides **a strong password authentication and encrypted communication** with a public key over an insecure channel.
 - It is used to replace unprotected remote login protocols such as **Telnet, rlogin, rsh, etc.**, and insecure file transfer protocol **FTP**.
- 

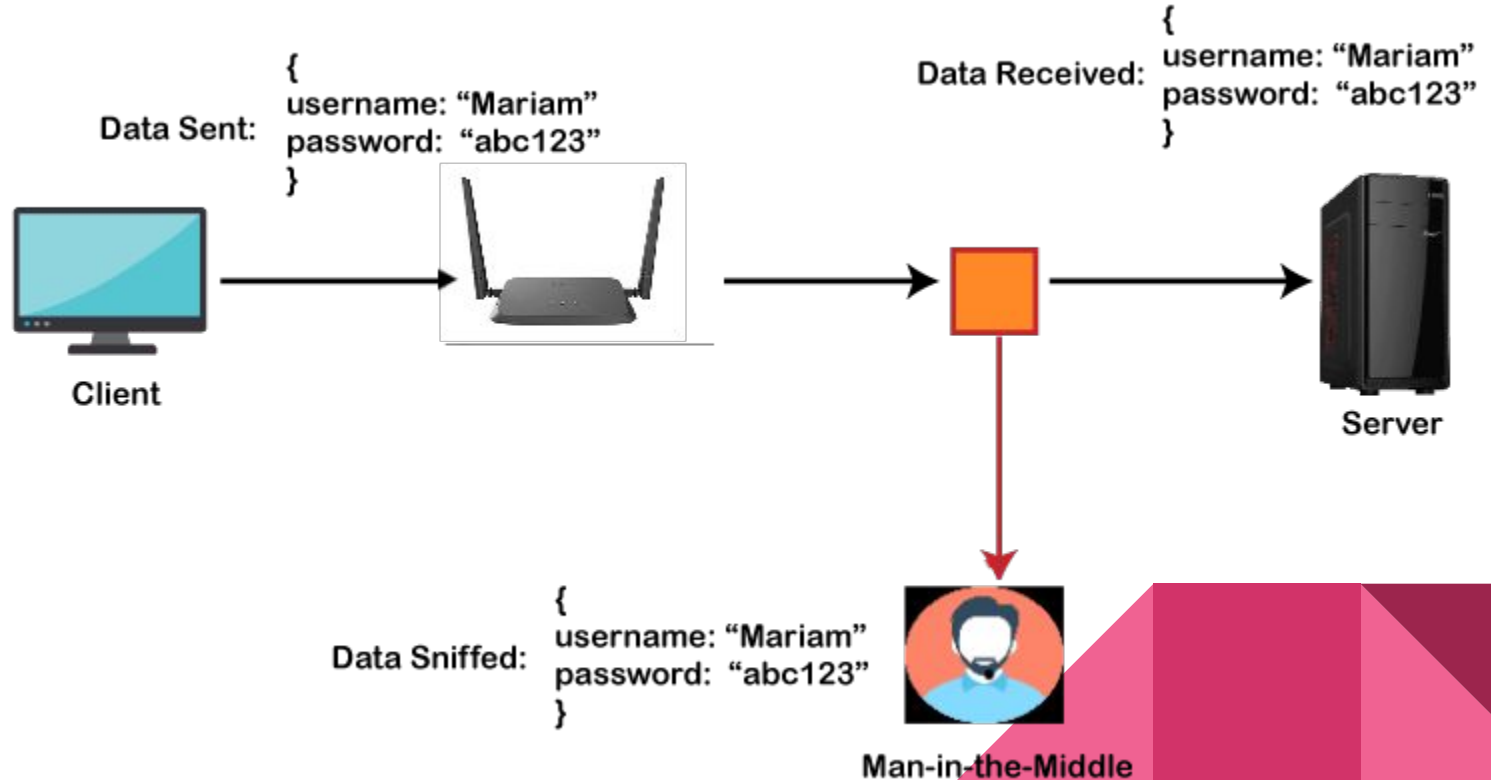
Secure Shell (SSH)

- Its security features are widely used by network administrators for managing systems and applications remotely.
- The SSH protocol protects the network from various attacks such as **DNS spoofing, IP source routing, and IP spoofing.**
- For example, suppose you want to transfer a package to one of your friends. Without SSH protocol, it can be opened and read by anyone. But if you will send it using SSH protocol, it will be encrypted and secured with the public keys, and only the receiver can open it.



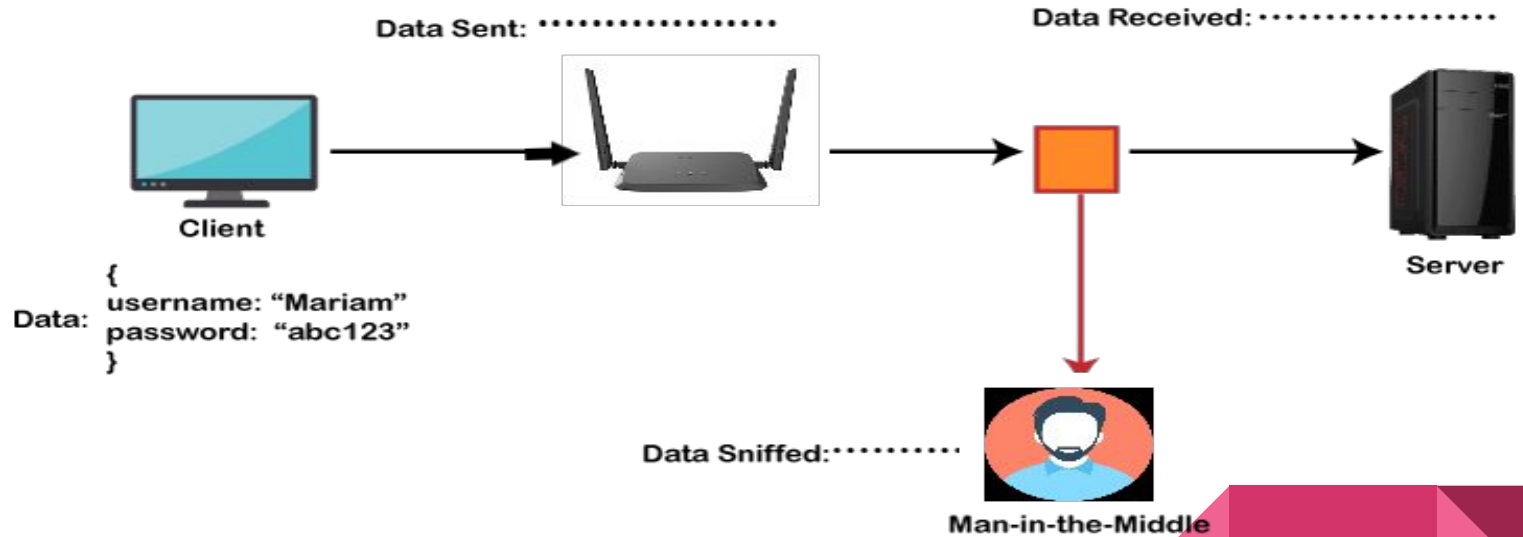
Secure Shell (SSH)

Before SSH:



Secure Shell (SSH)

After SSH:



Secure Shell (SSH)

How does SSH Works?

- The SSH protocol works in a **client-server** model, which means it connects a secure shell client application (End where the session is displayed) with the SSH server (End where session executes).
- It was initially developed to replace insecure login protocols such as Telnet, rlogin, and hence it performs the same function.
- The basic use of SSH is to connect a remote system for a terminal session and to do this, following command is used:
- `ssh UserName@SSHserver.test.com`
- The above command enables the client to connect to the server, named *server.test.com*, using the ID *UserName*.

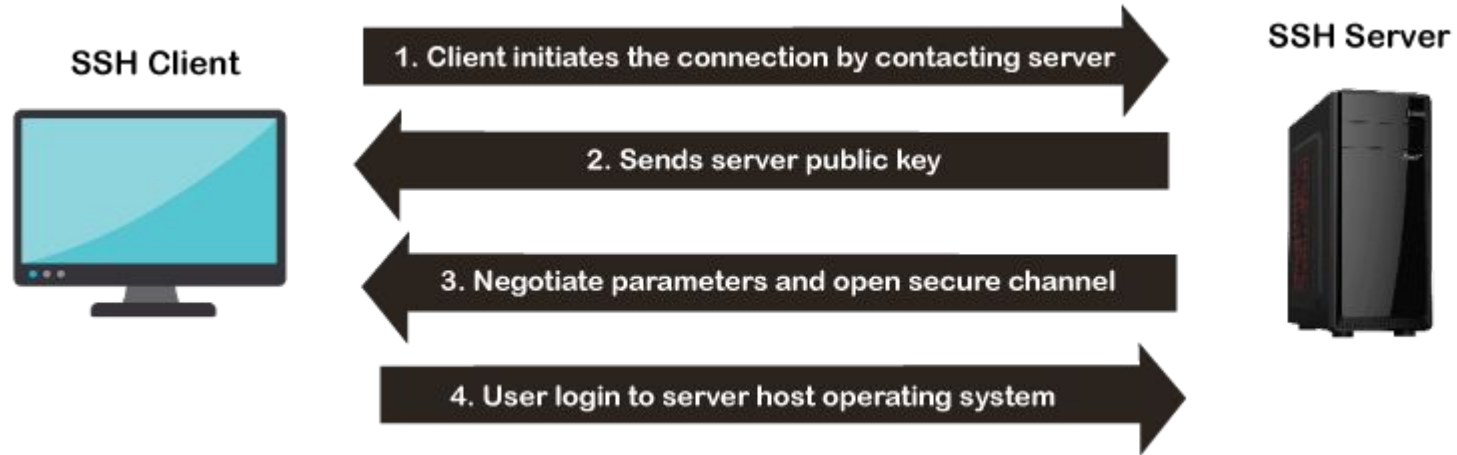


Secure Shell (SSH)

- If we are connecting for the first time, it will prompt the remote host's public key fingerprint and ask to connect. The below message will be prompt:
- The authenticity of host 'sample.ssh.com' cannot be established.
- DSA key fingerprint is 01:23:45:67:89:ab:cd:ef:ff:fe:dc:ba:98:76:54:32:10.
- Are you sure you want to continue connecting (yes/no)?
- To continue the session, we need to click yes, else no. If we click yes, then the host key will be stored in the known_hosts file of the local system. The key is contained within the hidden file by default, which is **/.ssh/known_hosts** in the home directory. Once the host key is stored in this hidden file, there is no need for further approval as the host key will automatically authenticate the connection.



Secure Shell (SSH)



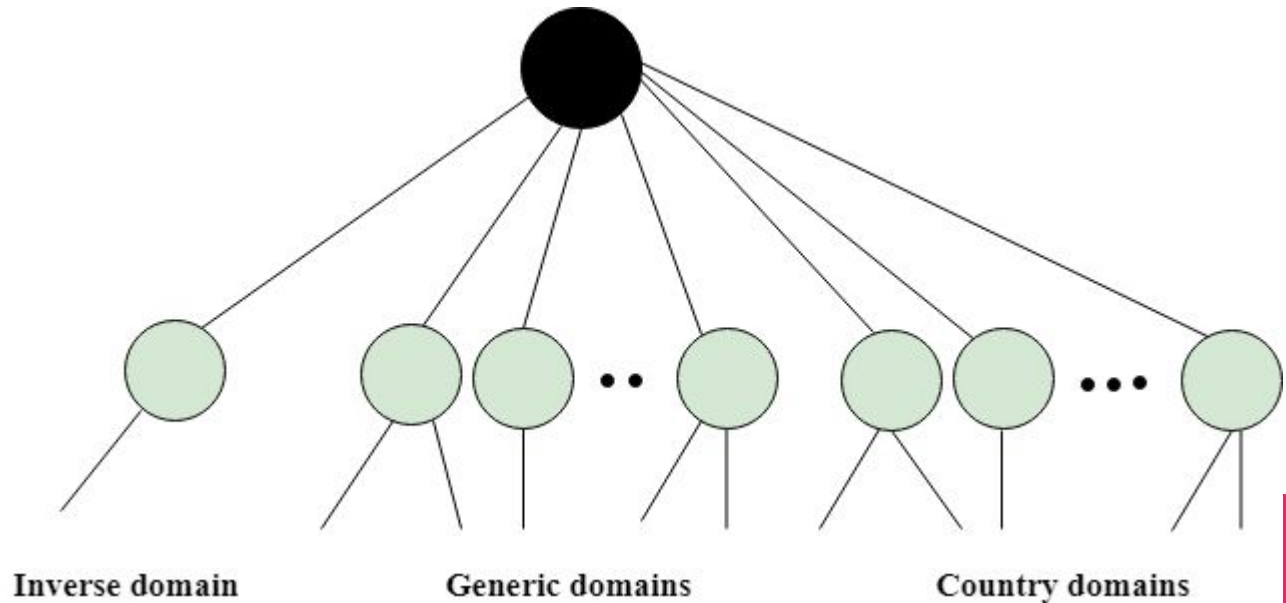
Domain Name System (DNS)

- DNS is a directory service that provides **a mapping between the name of a host on the network and its numerical address.**
- DNS is required for the functioning of the internet.
- Each node in a tree has a domain name, and a full domain name is a sequence of symbols specified by dots.
- DNS is a service that translates the domain name into IP addresses. This allows the users of networks to utilize user-friendly names when looking for other hosts instead of remembering the IP addresses.
- For example, suppose the FTP site at EduSoft had an IP address of 132.147.165.50, most people would reach this site by specifying ftp.EduSoft.com. Therefore, the domain name is more reliable than IP address.



Domain Name System (DNS)

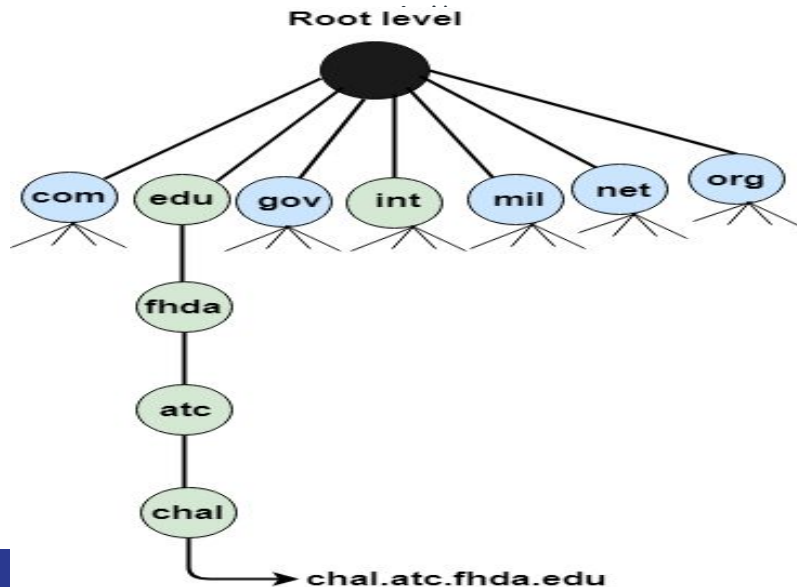
- DNS is a TCP/IP protocol used on different platforms.
- The domain name space is divided into three different sections: generic domains, country domains, and inverse domain.



Domain Name System (DNS)

Generic Domains

- It defines the registered hosts according to their generic behavior.
- Each node in a tree defines the domain name, which is an index to the DNS database.
- It uses three-character labels, and these labels describe the organization type.
- .com(commercial), .edu(educational), .mil(military), .org(nonprofit organization), .net(similar



domains.


Label	Description
aero	Airlines and aerospace companies
biz	Businesses or firms
com	Commercial Organizations
coop	Cooperative business Organizations
edu	Educational institutions
gov	Government institutions

Domain Name System (DNS)

Country Domain

- The format of country domain is same as a generic domain, but it uses two-character country abbreviations (e.g., us for the United States) in place of three character organizational abbreviations.
- .in (India) .us .uk

Inverse Domain

- It is used for mapping an address to a name.
 - When a client requests to the server, the server has a list of authorized clients.
 - It sends the query to the DNS server to verify the client belongs to the list of authorized clients and sends a query to the DNS server to map an address to the name.
- 

Domain Name System (DNS)

Working of DNS


- DNS is a client/server network communication protocol. DNS clients send requests to the server while DNS servers send responses to the client.
 - Client requests contain a name which is converted into an IP address known as a forward DNS lookups while requests containing an IP address which is converted into a name known as reverse DNS lookups.
 - DNS implements a distributed database to store the name of all the hosts available on the internet.
 - If a client like a web browser sends a request containing a hostname, then a piece of software such as **DNS resolver** sends a request to the DNS server to obtain the IP address of a hostname. If DNS server does not contain the IP address associated with a hostname, then it forwards the request to another DNS server. If IP address has arrived at the resolver, which in turn completes the request over the internet protocol.
- 

Figure 2.37 *Domain names and labels*

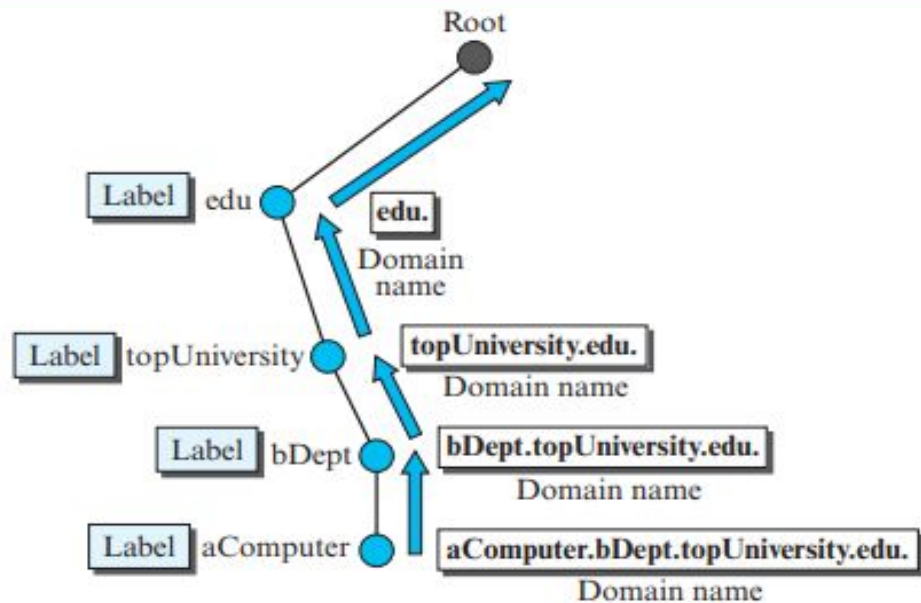


Figure 2.38 *Domains*

