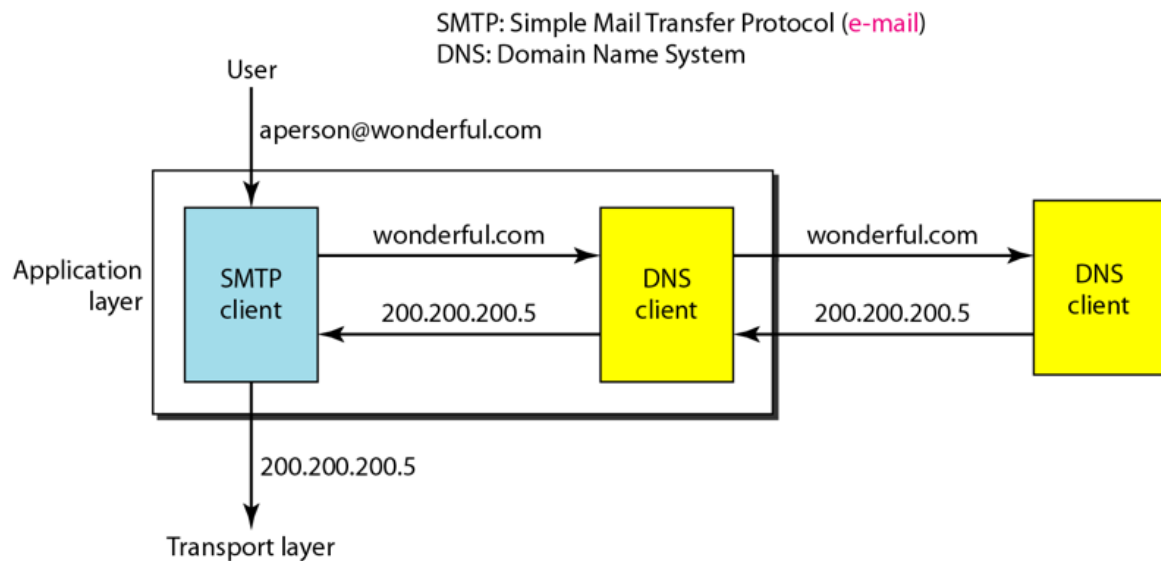


UNIT 5

Application Layer & Network Security

Domain Name System:

The Domain Name system is a client – server application that identifies each host on the Internet with a unique user - friendly name. This system organizes the name space in a hierarchical structure to decentralise responsibilities.



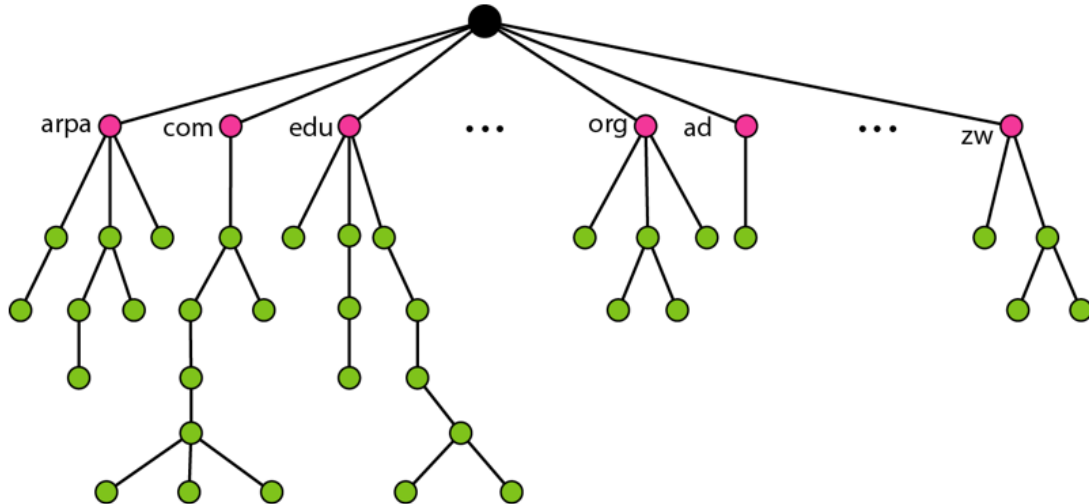
The above figure shows an example of how a DNS client/server program can support an e-mail program to find the IP address of an e-mail recipient. A user of an e-mail program may know the e-mail address of the recipient; however, the IP protocol needs the IP address. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address. The method 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).

Name Space

The names assigned to machines must be carefully selected from a namespace with complete control over the binding between the names and IP addresses. 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: fiat or hierarchical.

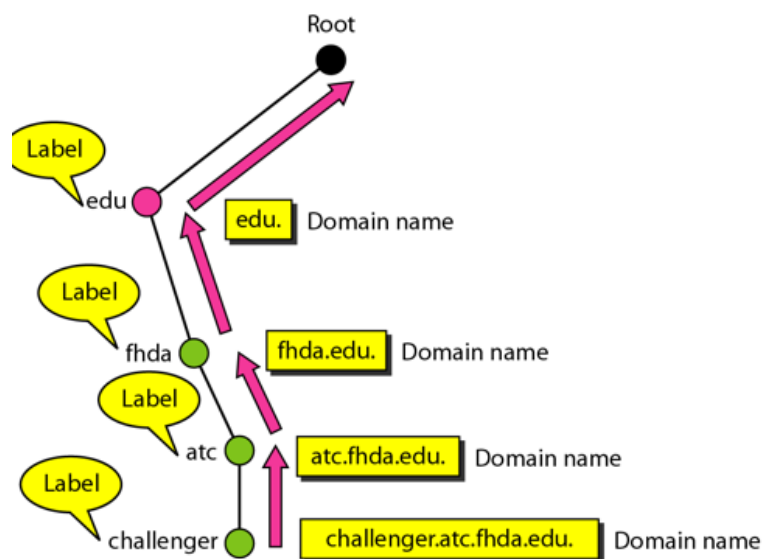
Domain Name Space

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.



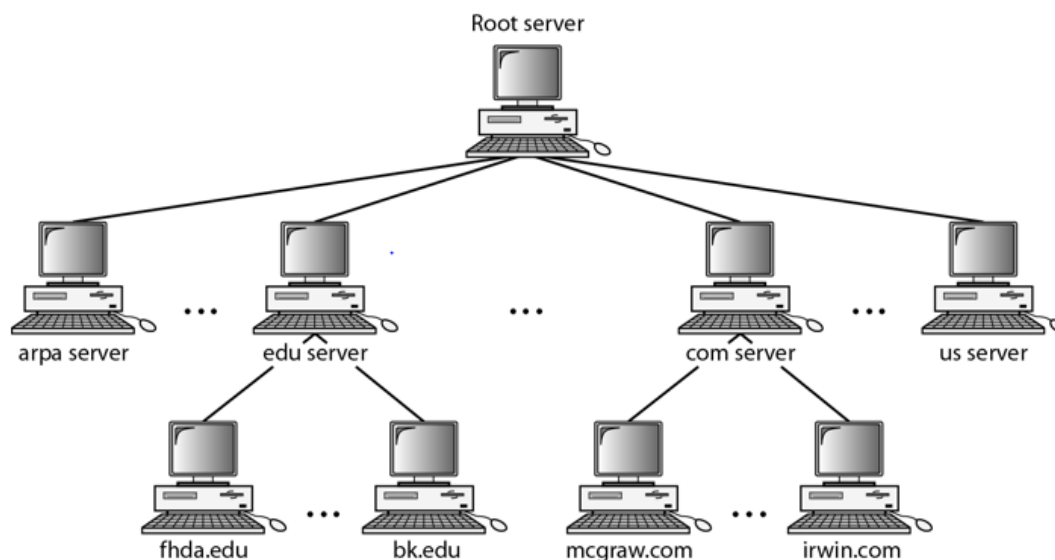
Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

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



Distribution of Name Space

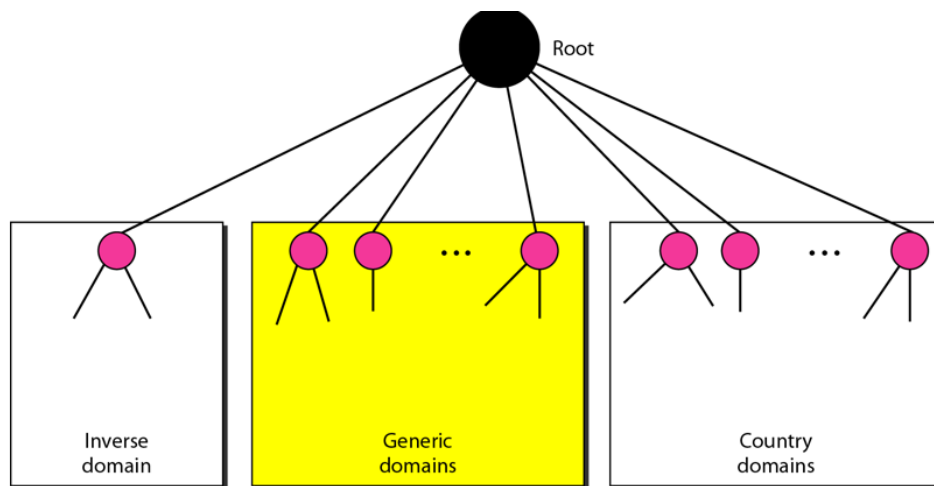
The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not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



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

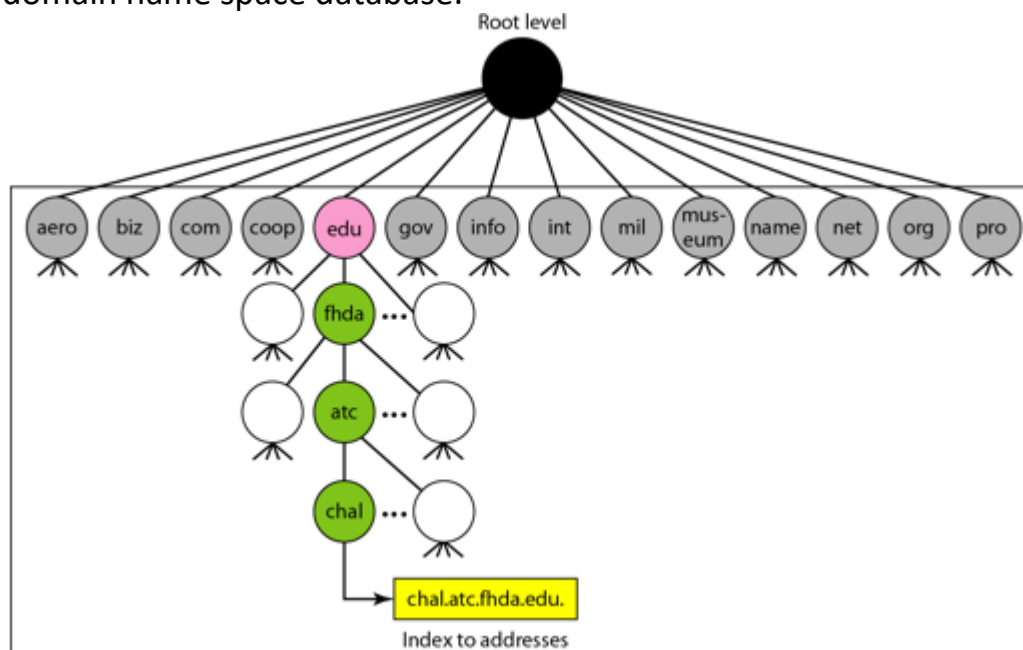
DNS In the Internet

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.



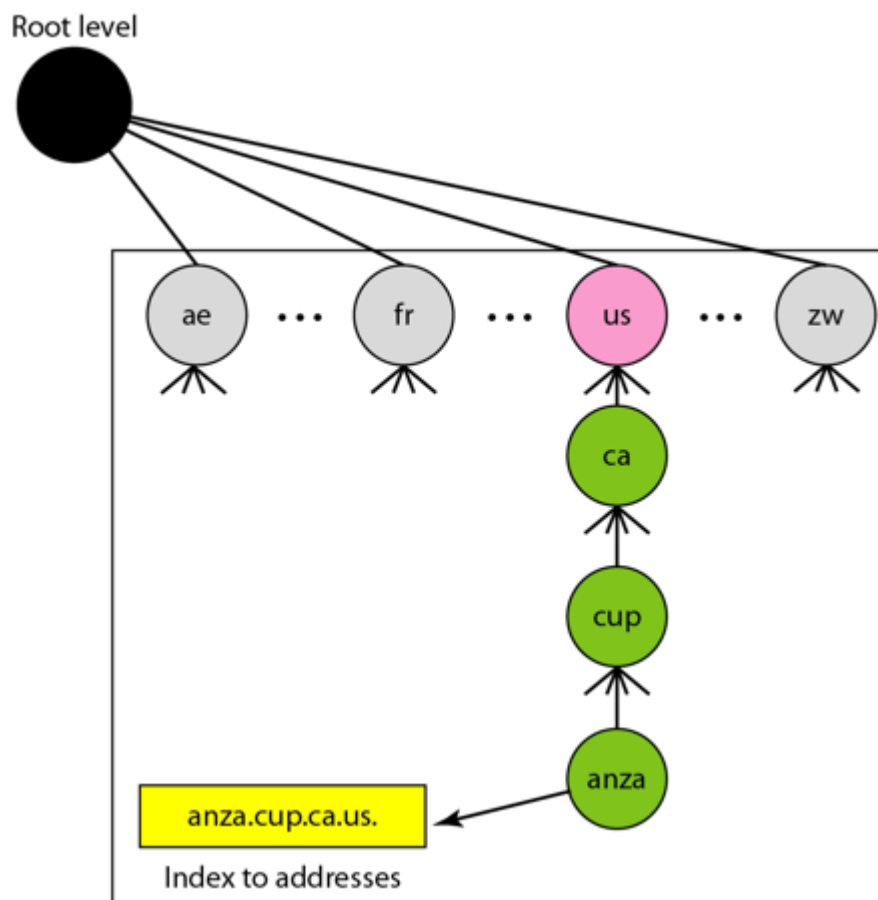
Generic Domains

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.



Country Domains

The country domains section uses two-character country abbreviations

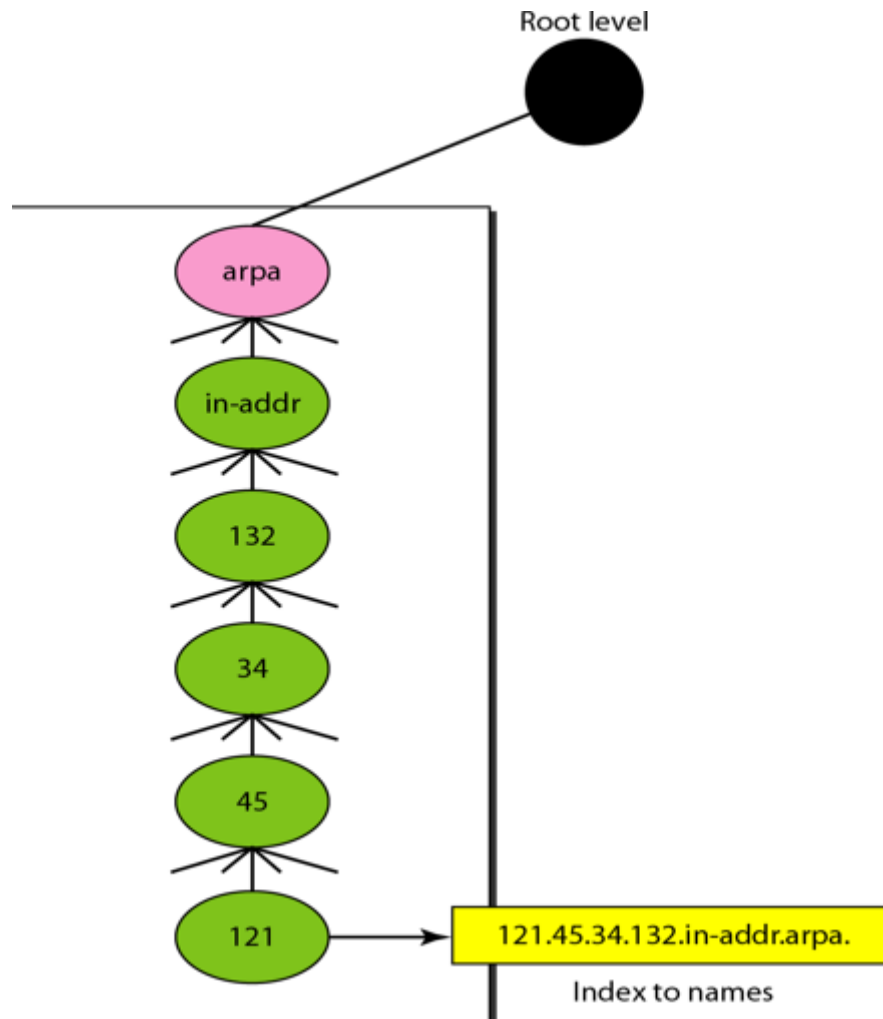


Inverse Domains

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. The servers that handle the inverse domain are also hierarchical.



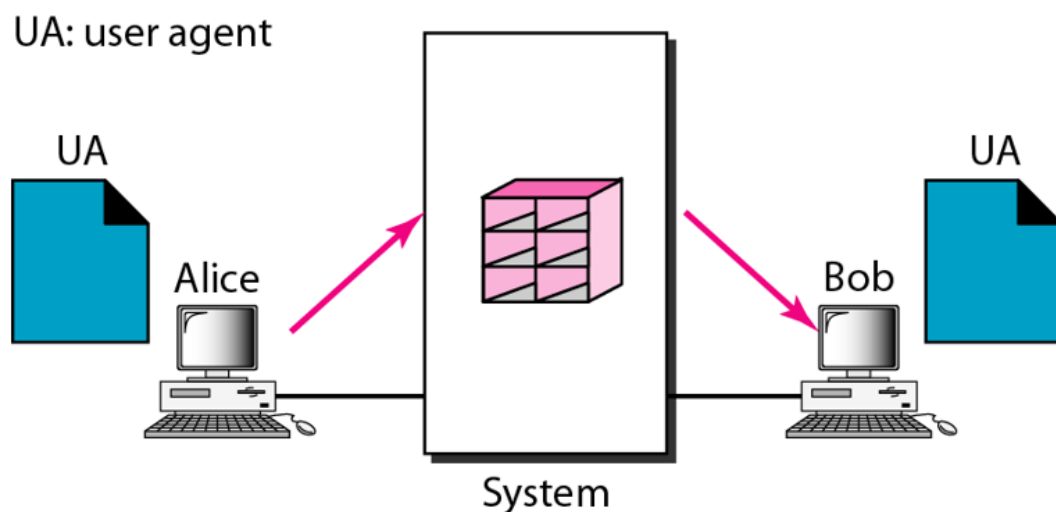
Electronic Mail

One of the most popular Internet services is electronic mail (e-mail). The designers of the Internet probably never imagined the popularity of this application program. Its architecture consists of several components that we discuss in this chapter. At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

To explain the architecture of e-mail, we give four scenarios. The 4th scenario is the most common one in the exchange of email.

1st Scenario

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same system; they are directly connected to a shared system. The administrator has created one mailbox for each user where the received messages are stored. A mailbox is part of a local hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice, a user, needs to send a message to Bob, another user, Alice runs a user agent (VA) 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.

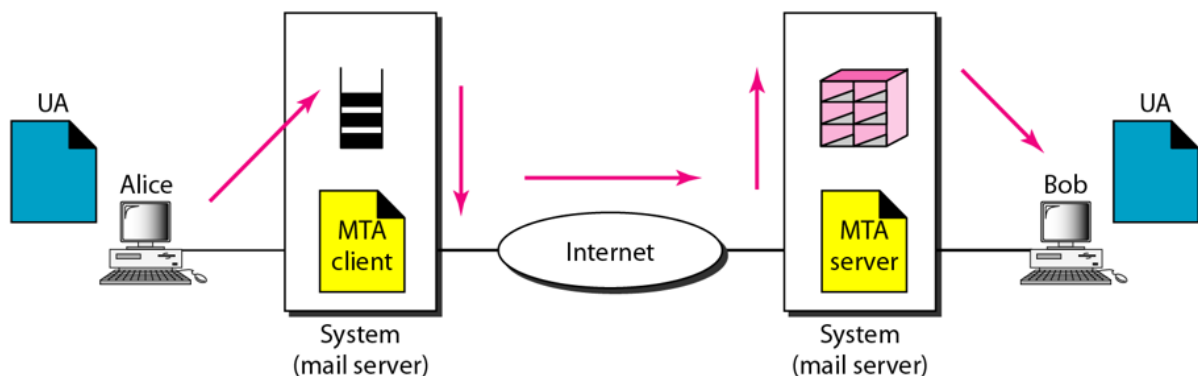


2nd Scenario

In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different systems. The message needs to be sent over the Internet. Here we need user agents (VAs) and message transfer agents (MTAs). Alice needs to use a user agent program to send her message to the system at her own site. The system (sometimes called the mail server) at her site uses a queue to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site. The message, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one 'client' and one server. Like most client/server programs on the Internet,

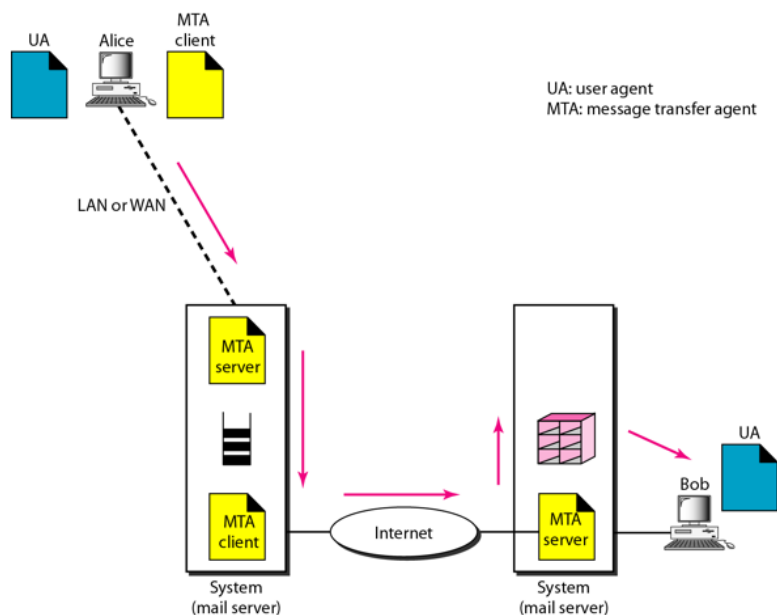
the server needs to run all the time because it does not know when a client will ask for a connection. The client, on the other hand, can be alerted by the system when there is a message in the queue to be sent.

UA: user agent
MTA: message transfer agent



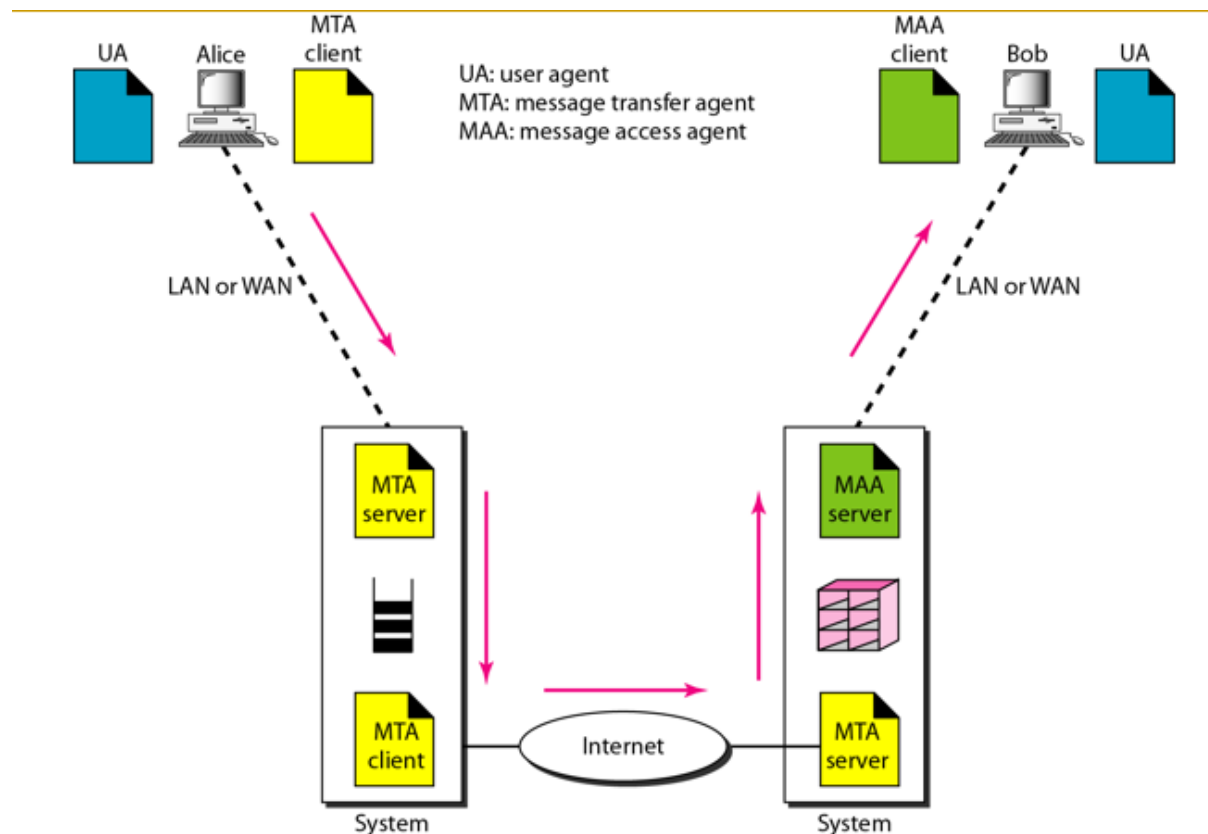
3rd Scenario

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



4th Scenario

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



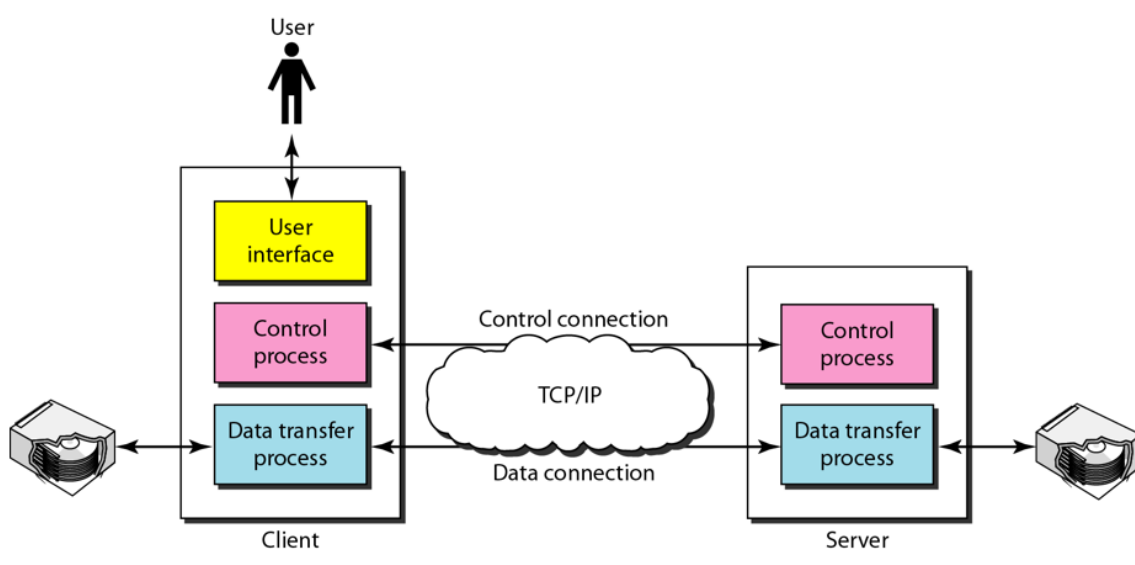
There are two important points here. First, Bob cannot bypass the mail server and use the MTA server directly. To use MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time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

File Transfer

Transferring files from one computer to another is one of the most common tasks expected from a networking or internetworking environment. As a matter of fact, the greatest volume of data exchange in the Internet today is due to file transfer. In this section, we discuss one popular protocol involved in transferring files: File Transfer Protocol (FTP).

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 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. However, the difference in complexity is at the FTP level, not TCP. For TCP, both connections are treated the same. FTP uses two well-known TCP ports: Port 21 is used for the control connection, and port 20 is used for the data connection.

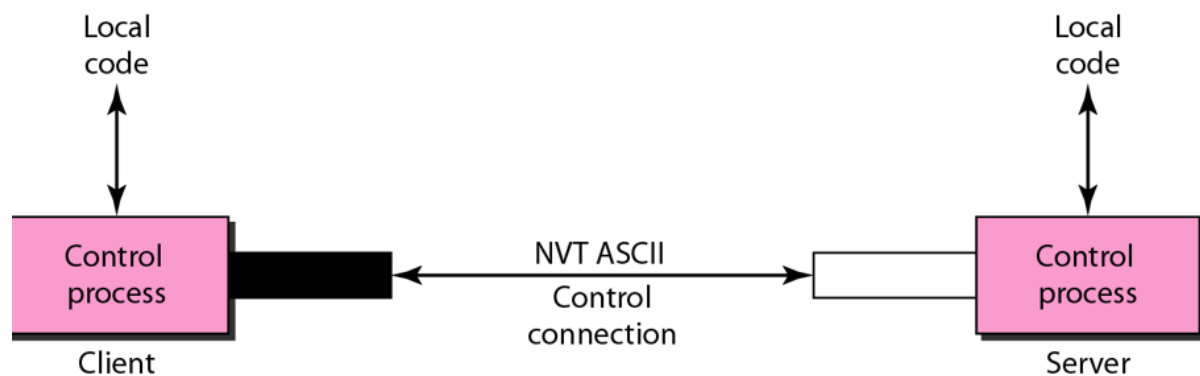


The purpose of the data connection is different from that of the control connection. We want to transfer files through the data connection. File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things:

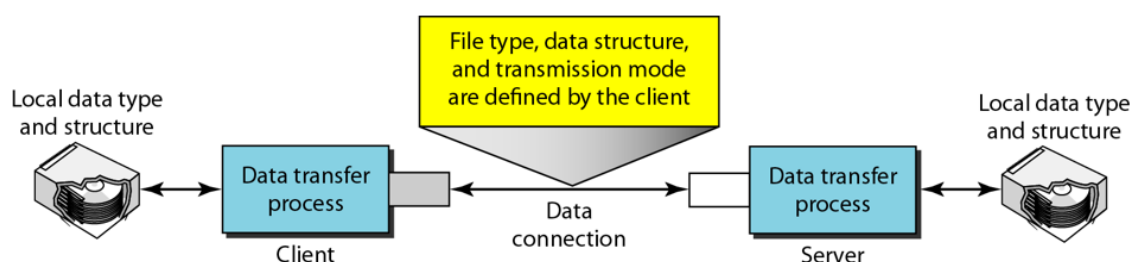
- o A file is to be copied from the server to the client. This is called retrieving a file. It is done under the supervision of the RETR command,
- o A file is to be copied from the client to the server. This is called storing a file. It is done under the supervision of the STOR command.
- o A list of directory or file names is to be sent from the server to the client. This is done under the supervision of the LIST command. Note that FTP treats a list of directory or file names as a file. It is sent over 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.

Control Connection



Data Connection



FTP Sample Session:

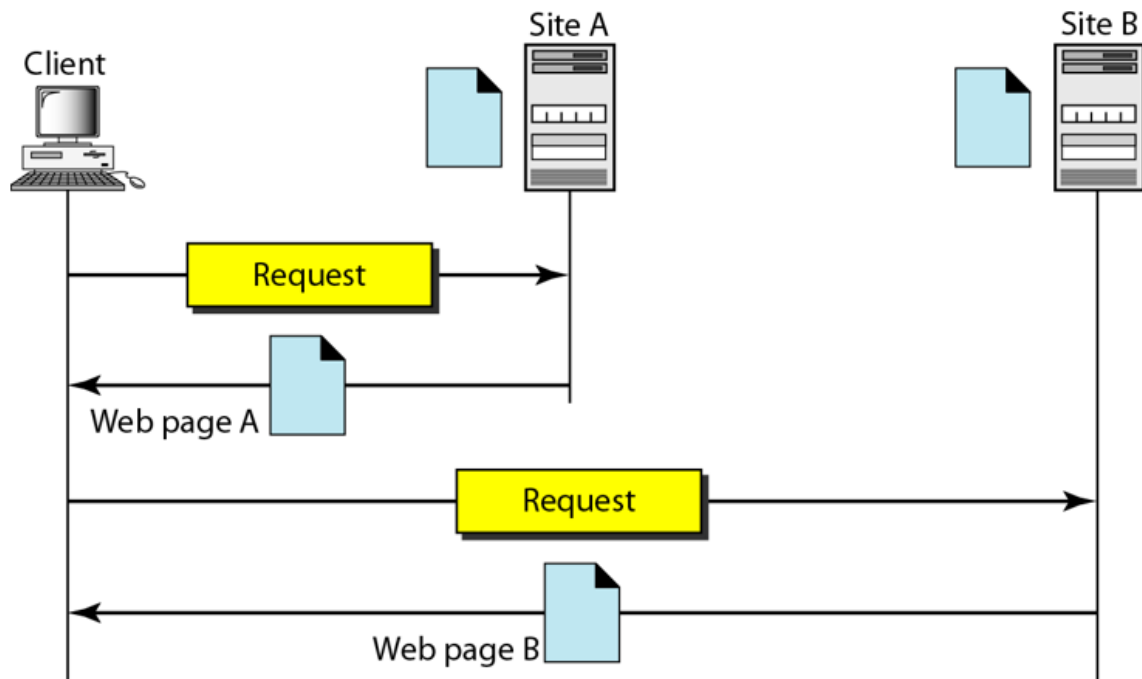
1. After the control connection is created, the FTP server sends the response.
2. The client sends its name.
3. The server responds.
4. The client sends the password.
5. The server responds (user log-in is OK).
6. The client sends the list command (ls reports) to find the list of files on the directory named report.
7. Now the server responds and opens the data connection.
8. The server then sends the list of the files or directories on the data connection.
9. The client sends a QUIT command.
10. The server responds.

WWW

The World Wide Web (WWW) is a repository of information linked together from points all over the world. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research. 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.

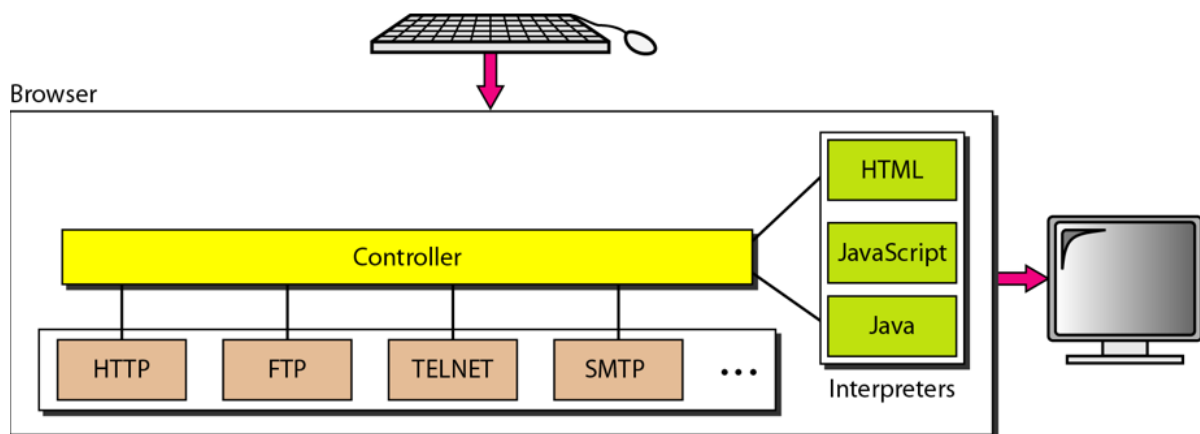
Architecture of WWW:

Each site holds one or more documents, referred to as Web pages. Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers. The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL, which we will discuss shortly. The server at site A finds the document and sends it to the client. When the user views the document, she finds some references to other documents, including a Web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.



Browser

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters. 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 previously such as FfP or HTIP (described later in the chapter). The interpreter can be HTML, Java, or JavaScript, depending on the type of document.



URL

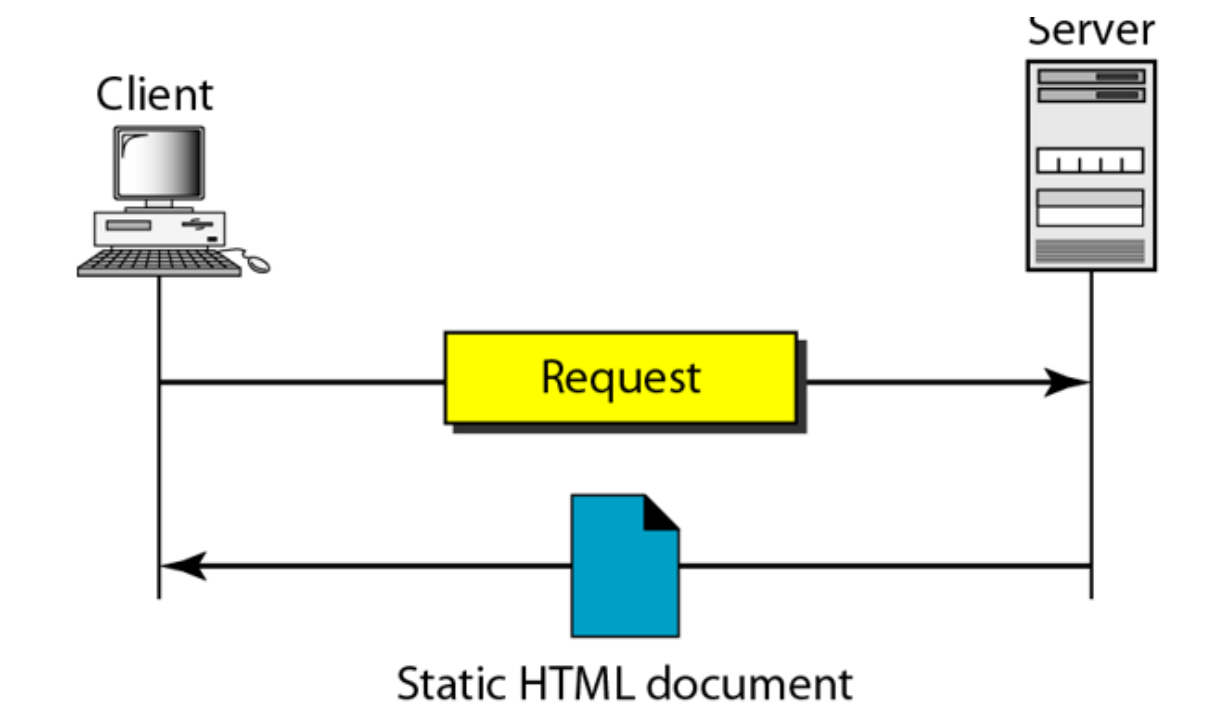
A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet.



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 at which the contents of the document are determined.

Static Documents

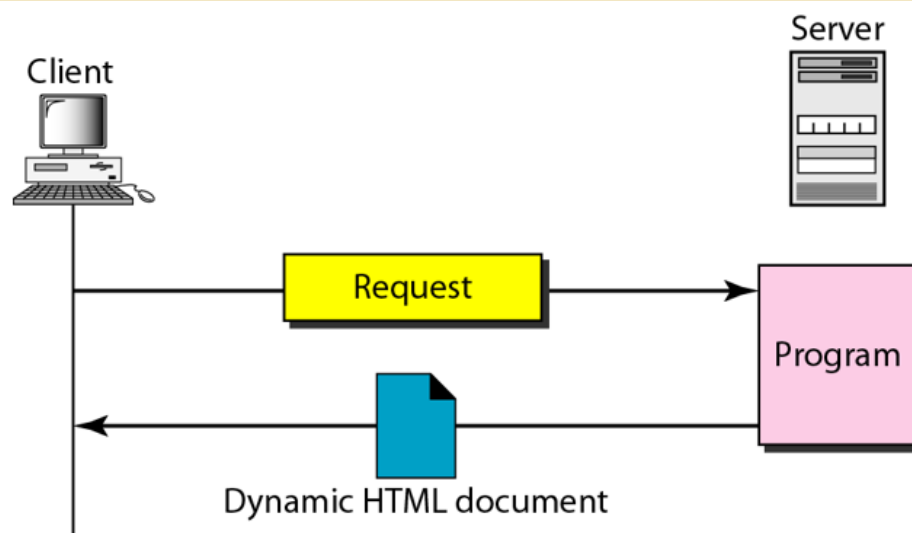


Static documents are fixed-content documents that are created and stored in a server. The client can get only a copy of the document. In other words, the contents of the file are determined when the file is created, not when it is used. Of course, the contents in the server can be changed, but the user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

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 can vary from one request to another. A very simple example of a dynamic document is the retrieval of the time and date from a server. Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the date program in UNIX and send the result of the program to the client.

Dynamic document using 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.

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. For example, a generic copy program can be written to copy any file to another. A user can use the program to copy a file named x to another file named y by passing x and y as parameters. The input from a browser to a server is sent by using a form. If the information is small (such as a word), it can be appended to the URL after a question mark. For example, the following URL is carrying information (23, a value):

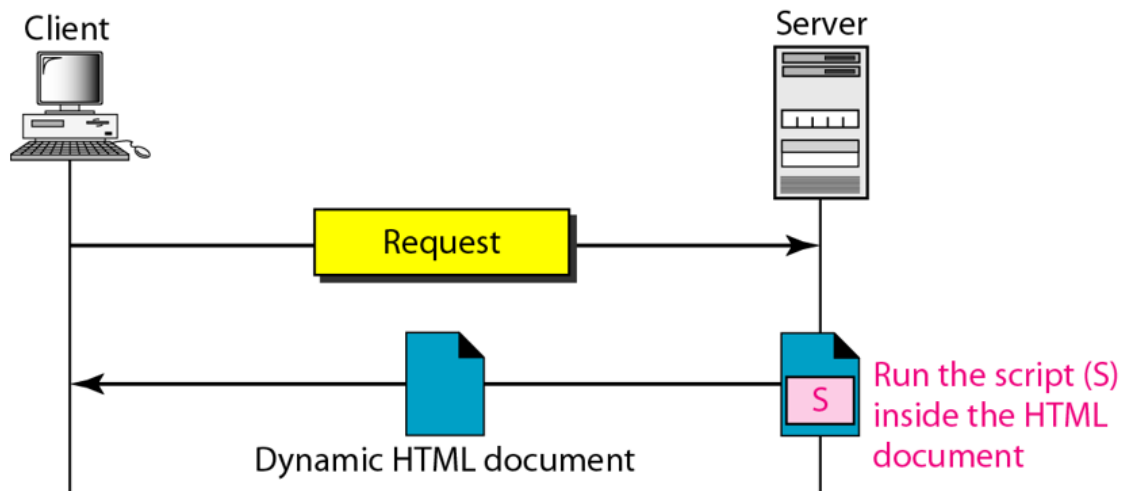
<http://www.deanzalcgi-bin/prog.pl?23>

When the server receives the URL, it uses the part of the URL before the question mark to access the program to be run, and it interprets the part after the question mark (23) as the input sent by the client. It stores this string in a variable. When the CGI program is executed, it can access this value. If the input from a browser is too long to fit in the query string, the browser can ask the server to send a phone. The browser can then fill the phone with the input data and send it to the server. The information in the phone can be used as the input to the CGI program.

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. It can be graphics or binary data, a status code, instructions to the browser to cache the result, or instructions to the server to send an existing document instead of the actual output. To let the client know about the type of document sent, a CGI program creates headers. As a matter of fact, the output of the CGI program always consists of two parts: a header and a body. The header is separated by a blank line from the body. This means any CGI program creates first the header, then a blank line, and then the body. Although the header and the blank line are not shown on the browser screen, the header is used by the browser to interpret the body.

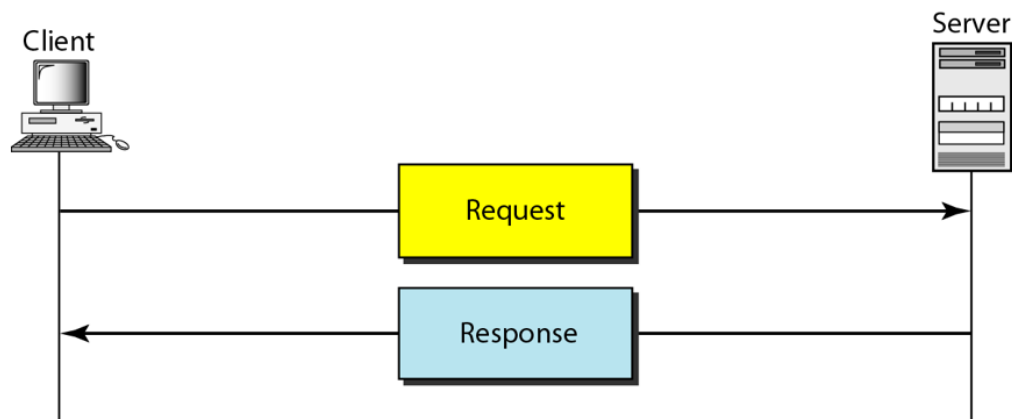
Dynamic document using server-side script



HTTP

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP. It is similar to FfP because it transfers files and uses the services of TCP. However, it is much simpler than FfP 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). SMTP messages are stored and forwarded, but HTTP messages are delivered immediately. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

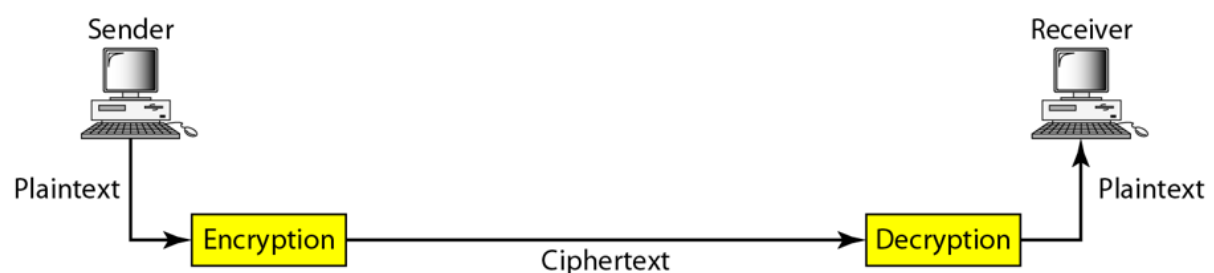
HTTP transaction

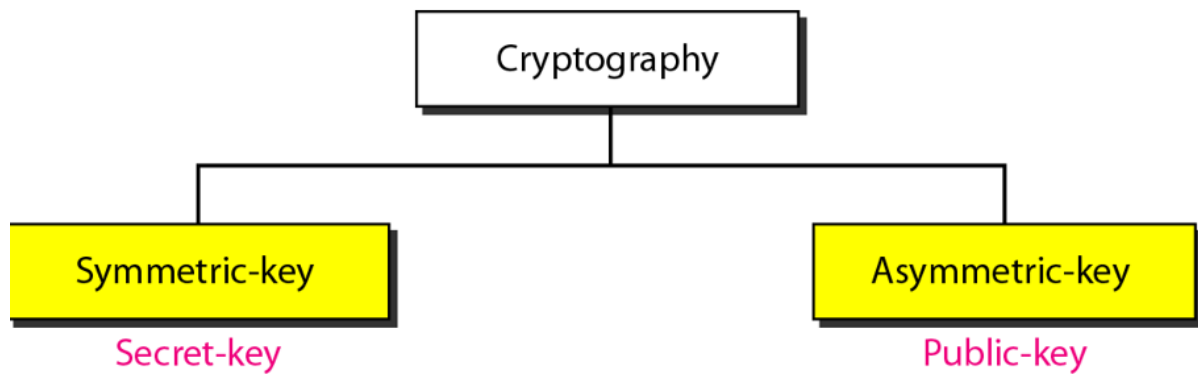


Network Security

1. *Cryptography is the science and art of transforming messages to make them secure and immune to attacks.*
2. *The plaintext is the original message before transformation; the ciphertext is the message after transformation.*
3. *An encryption algorithm transforms plaintext to ciphertext; a decryption algorithm transforms ciphertext to plaintext.*
4. *A combination of an encryption algorithm and a decryption algorithm is called a cipher.*
5. *The key is a number or a set of numbers on which the cipher operates.*

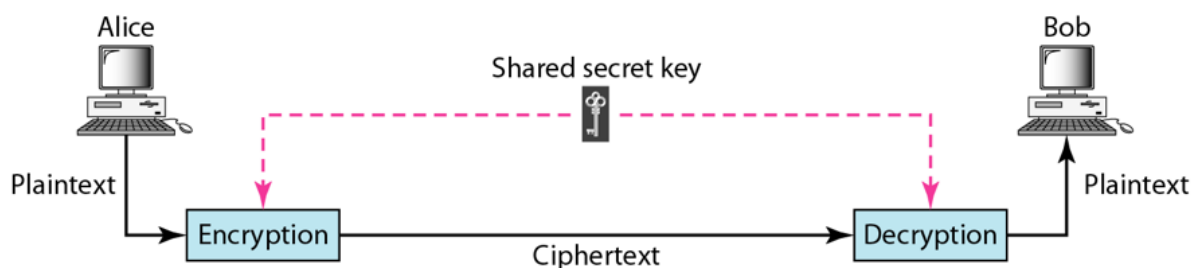
Cryptography components





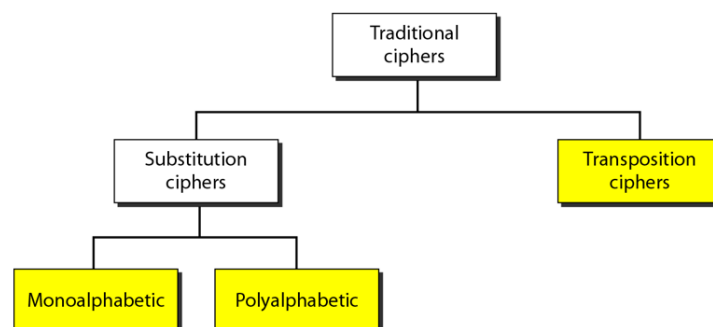
Symmetric Key Cryptography

In symmetric-key cryptography, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data.



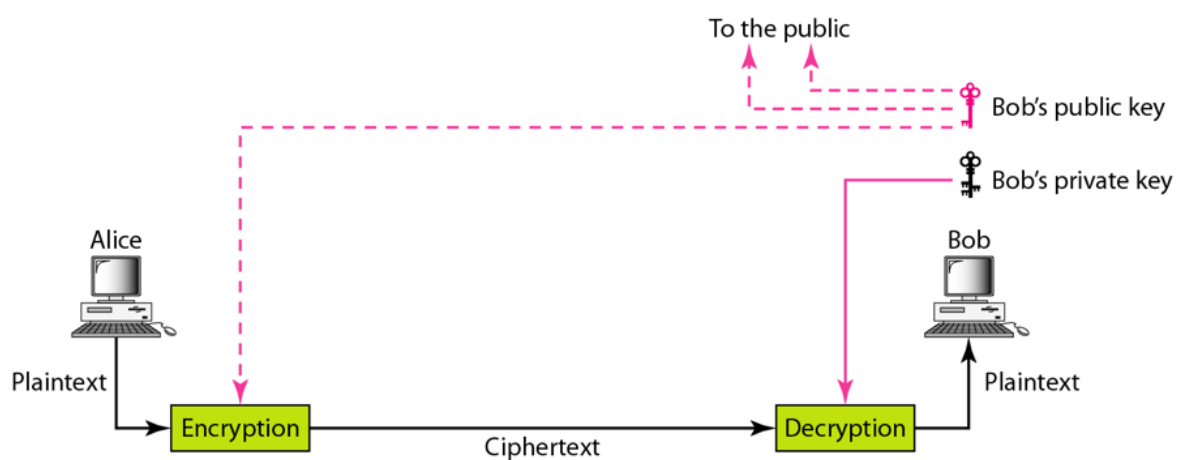
Traditional Ciphers

We briefly introduce some traditional ciphers, which are character-oriented. Although these are now obsolete, the goal is to show how modern ciphers evolved from them. We can divide traditional symmetric-key ciphers into two broad categories: substitution ciphers and transposition ciphers.



Asymmetric Key Cryptography

In asymmetric or public-key cryptography, there are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public. In the Figure, imagine Alice wants to send a message to Bob. Alice uses the public key to encrypt the message. When the message is received by Bob, the private key is used to decrypt the message. In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption. The public key is available to the public; the private key is available only to an individual.



Encryption can be thought of as electronic locking; decryption as electronic unlocking. The sender puts the message in a box and locks the box by using a key; the receiver unlocks the box with a key and takes out the message. The difference lies in the mechanism of the locking and unlocking and the type of keys used.

In symmetric-key cryptography, the same key locks and unlocks the box. In asymmetric-key cryptography, one key locks the box, but another key is needed to unlock it.
