COMPUTER NETWORKS
UNIT V

Email (SMTP, MIME, IMAP, POP3) – HTTP – DNS- SNMP – Telnet – FTP – Security – PGP - SSH

- SMTP: Simple Mail Transfer Protocol is used to exchange electronic mail.
- MIME (Multipurpose Internet Mail Extensions) define the format of email messages.
- **Internet message access protocol** (**IMAP**) is one of the two most prevalent Internet standard protocols for e-mail retrieval, the other being the Post Office Protocol (POP).
- RFC-822:The internet standard format for electronic mail message headers.

Email (SMTP, MIME, IMAP, POP3)

- Email is one of the oldest network applications.
- Email works as

(1) To distinguish the user interface (i.e., your mail reader) from the underlying message transfer protocol (in this case, SMTP), and.
(2) To distinguish between this transfer protocol and a companion protocol (RFC 822 and MIME) that defines the format of the messages being exchanged.

Message Format

- RFC 822 defines messages to have two parts: a *header and a body. Both parts are* represented in ASCII text.
- The message header is a series of <CRLF>-terminated lines. (<CRLF> stands for carriage-return + line-feed, which are a pair of ASCII control characters often used to indicate the end of a line of text.)
- The header is separated from the message body by a blank line.
- Each header line contains a type and value separated by a colon.
- Many of these header lines are familiar to users since they are asked to fill them out when they compose an email message.
- For example, the To: header identifies the message recipient, and the Subject: header says something about the purpose of the message. Other headers are filled in by the underlying mail delivery system.
- Examples include Date: (when the message was transmitted), From: (what user sent the message), and Received: (each mail server that handled this message).
- RFC 822 was extended in 1993 (and updated again in 1996) to allow email messages to carry many different types of data: audio, video, images,Word documents,and so on.
- MIME consists of three basic pieces.
- The first piece is a collection of header lines that augment the original set defined by RFC 822.

- These header lines describe, in various ways, the data being carried in the message body.
- They include
- MIME-Version: (the version of MIME being used),
- Content-Description: (a human-readable description of what's in the message, analogous to the Subject: line),
- Content-Type: (the type of data contained in the message), and Content-Transfer-Encoding: (how the data in the message body is encoded).
- The second piece is definitions for a set of content types (and subtypes).
- For example, MIME defines two different still image types, denoted image/gif and image/jpeg, each with the obvious meaning.
- As another example, text/plain refers to simple text you might find in a vanilla 822-style message, while text/richtext denotes a message that contain ―marked up‖ text (e.g., text using special fonts, italics, etc.).
- As a third example, MIME defines an application type, where the subtypes correspond to the output of different application programs (e.g. application/postscript and application/msword).
- MIME also defines a multipart type that says how a message carrying more than
- one data type is structured. This is like a programming language that defines both
- base types (e.g., integers and floats) and compound types (e.g., structures and arrays).

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="--------417CA6E2DE4ABCAFBC5"
From: Alice Smith <Alice@cisco.com>
To: Bob@cs.Princeton.edu
Subject: promised material
Date: Mon, 07 Sep 1998 19:45:19 -0400

---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Bob,

Here's the jpeg image and draft report I promised.

--Alice

---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

. . . unreadable encoding of a jpeg figure

---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit

. . . readable encoding of a PostScript document
```

MIME

- Electronic mail has a simple structure.
- It can send messages only in NVT 7-bit ASCII format.
- For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (such as French, German, Hebrew, Russian, Chinese, and Japanese).

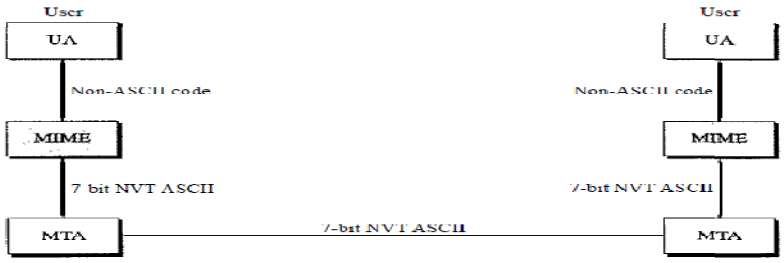
- Also, it cannot be used to send binary files or video or audio data.

- Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail.

- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client to be sent through the Internet. The message at the receiving side is transformed back to the original data.

- We can think of MIME as a set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa.

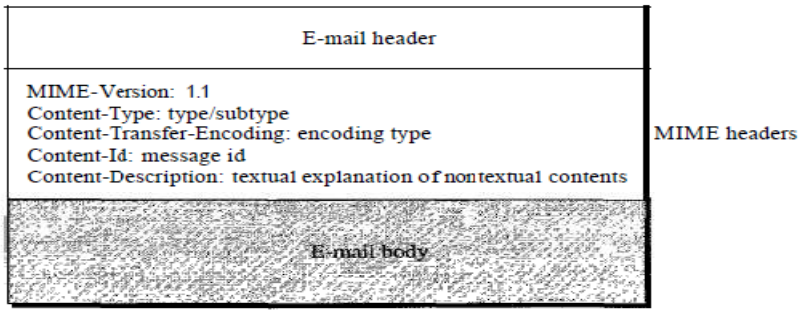
MIME



- MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters.

1. MIME-Version
2. Content-Type
3. Content-Transfer-Encoding
4. Content-Id
5. Content-Description

MIME header



- MIME-Version This header defines the version of MIME used. The current version is 1.1.

- Content-Type This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters.

Content-Type: <type /subtype; parameters>

Data types and subtypes in MIME

- MIME allows seven different types of data

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Chapter 27)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to mixed subtypes, but the default is message/ RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	IPEG	Image is in IPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single-channel encoding of voice at 8 kHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (8-bit bytes)

- Content-Transfer-Encoding This header defines the method used to encode the messages into Os and Is for transport.

Content-Transfer-Encoding: <type>

Content-transfer-encoding

Type	*Description*
7-bit	NVT ASCII characters and short lines
8-bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base-64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters encoded as an equals sign followed by an ASCII code

- Content-Id This header uniquely identifies the whole message in a multiple-message environment.

Content-Id: id=<content-id>

- Content-Description This header defines whether the body is image, audio, or video.

Content-Description: <description>

SMTP

Message Transfer

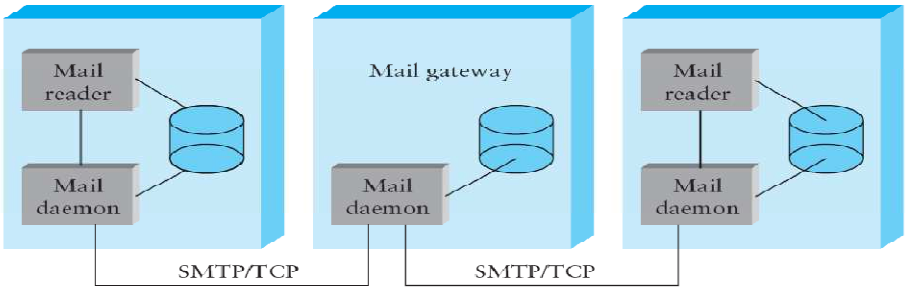
- SMTP—the protocol used to transfer messages from one host to another.
- First, users interact with a *mail reader when they compose, file, search, and read their email.*
- Most Web browsers now include a mail reader.
- Second, there is a *mail daemon (or process) running on each host. this process as playing* the role of a post office:

- Mail readers give the daemon messages they want to send to other users, the daemon uses SMTP running over TCP to transmit the message to a daemon running on another machine, and the daemon puts incoming messages into the user's *mailbox (where that user's mail reader can later find it).*

- While it is certainly possible that the sendmail program on a sender's machine establishes an SMTP/TCP connection to the sendmail program on the recipient's machine, in many cases the mail traverses one or more *mail gateways on its route from the* sender's host to the receiver's host.

- Like the end hosts, these gateways also run a sendmail process. It's not an accident that these intermediate nodes are called ―gateways‖ since their job is to store and forward email messages, much like an ―IP gateway‖ (which we have referred to as a router) stores and forwards IP datagrams.

Sequence of mail gateways store and forward email messages



- The forwarding gateway maintains a database that maps users into the machine on which they currently want to receive their mail; the sender need not be aware of this specific name.

- Another reason is that the recipient's machine may not always be up, in which case the mail gateway holds the message until it can be delivered.

- Each SMTP session involves a dialog between the two mail daemons, with one acting as the client and the other acting as the server. Multiple messages might be transferred between the two hosts during a single session.

- SMTP is best understood by a simple example.

The following is an exchange between sending host cs.princeton.edu and receiving host cisco.com. In this case, user Bob at Princeton is trying to send mail to users Alice and Tom at Cisco.

HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]
MAIL FROM:<Bob@cs.princeton.edu>
250 OK
RCPT TO:<Alice@cisco.com>
250 OK
RCPT TO:<Tom@cisco.com>
550 No such user here
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
...etc. etc. etc.
<CRLF>.<CRLF>

250 OK
QUIT
221 Closing connection

- As you can see, SMTP involves a sequence of exchanges between the client the server. In each exchange, the client posts a command (e.g., HELO, MAIL, RCPT,DATA, QUIT) and the server responds with a code (e.g., 250, 550, 354, 221).

Mail Reader

- The final step is for the user to actually retrieve his or her messages from the mailbox read them, reply to them, and possibly save a copy for future reference. The user performs all these actions by interacting with .a mail reader.

- In many cases, this reader is just a program running on the same machine as the user's mailbox resides, in which case it simply reads and writes the file that implements the mailbox.

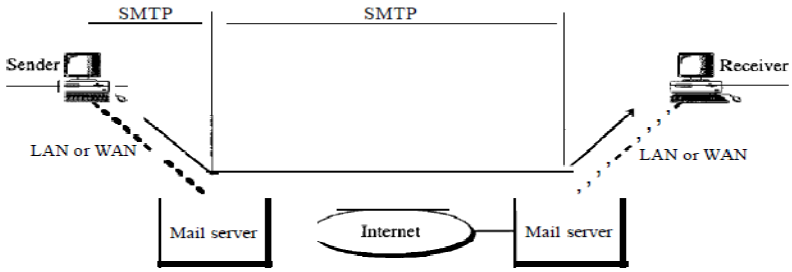
- In other cases, the user accesses his or her mailbox from a remote machine using yet another protocol, such as the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP).

Message Transfer Agent: SMTP

- The actual mail transfer is done through message transfer agents. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.

- The formal protocol that defines the MTA client and server in the Internet is called the Simple Mail Transfer Protocol (SMTP).

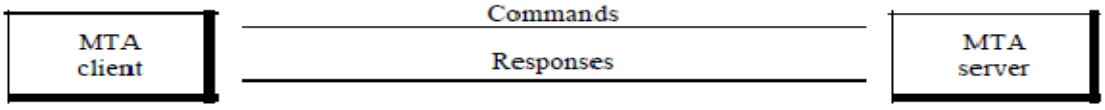
SMTP range



SMTP simply defines how commands and responses must be sent back and forth

Commands and Responses

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



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

- Commands Commands are sent from the client to the server.

- It consists of a keyword followed by zero or more arguments.

- SMTP defines 14 commands. The first five are mandatory; every implementation must support these five commands. The next three are often used and highly recommended.

- The last six are seldom used.

Commandformat

Keyword: argument(s)

Keyword	*Argument(s)*
HELO	Sender's host name
MAIL FROM	Sender of the message
RCPTTO	Intended recipient of the message
DATA	Body of the mail
QUIT	
RSET	
VRFY	Name of recipient to be verified
NOOP	
TURN	
EXPN	Mailing list to be expanded
HELP	Command name

Keyword	*Argument(s)*
SEND FROM	Intended recipient of the message
SMOLFROM	Intended recipient of the message
SMALFROM	Intended recipient of the message

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.

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

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

$ telnet mail.adelphia.net 25
Trying 68.168.78.100 •••
Connected to mail.adelphia.net (68.168.78.100).

Connection Establishment
220 mta13.adelphia.net **SMTP** serverready Fri, 6 Aug 2004 . . .
HELO mail.adelphia.net
250 mta13.adelphia.net
Mail Transfer
MAIL FROM: forouzanb@adelphia.net
250 Sender: <forouzanb@adelphia.net> Ok
RCPT TO: forouzanb@adelphia.net
250 Recipient <forouzanb@adelphia.net> Ok
DATA
354 Ok Send data **ending with** <CRLF>.<CRLF>
From: Forouzan
TO: Forouzan

This is a **test message**
to show SMTP in action.

Connection Termination
250 Message received: adelphia.net@mail.adelphia.net
QUIT
221 mta13.adelphia.net SMTP server closing connection
Connection closed by **foreign host.**

Message Access Agent: POP and IMAP

- *They are called a pull protocol; the client must pull messages from the server.*

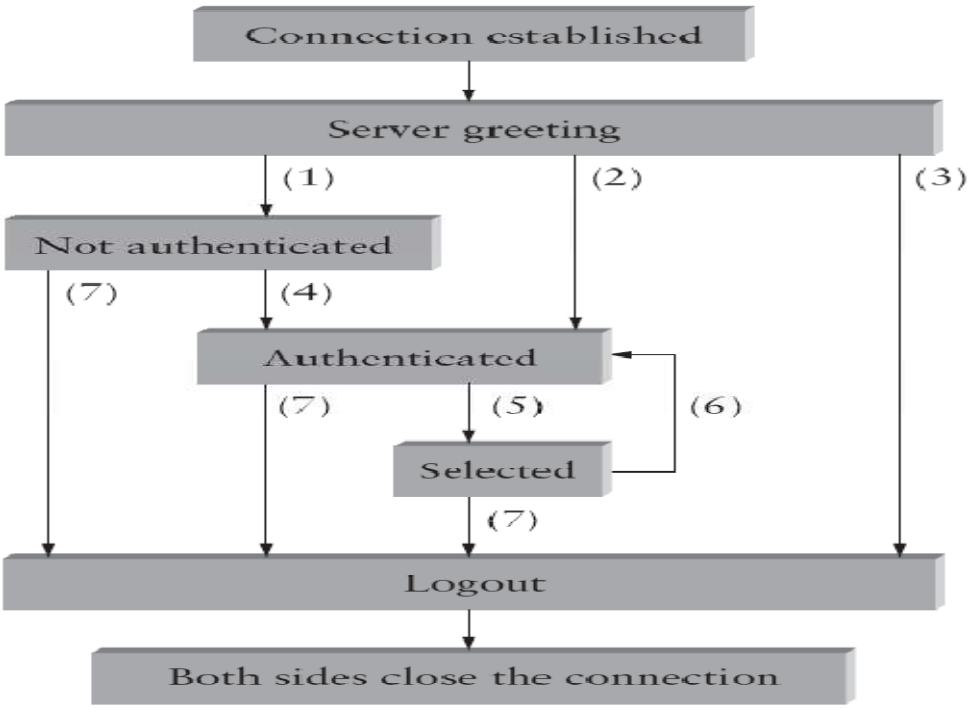
- *The direction of the bulk data is from the server to the client. The third stage uses a message access agent*

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

IMAP

- IMAP is similar to SMTP in many ways.

- It is a client/server protocol running over TCP, where the client (running on user's desktop machine) issues commands in the form of <CRLF>-terminated ASCII text lines and the mail server (running on the machine that maintains the user's mailbox) responds in kind.



(1) connection without preauthentication (OK greeting)
(2) preauthenticated connection (PREAUTH greeting)
(3) rejected connection (BYE greeting)
(4) successful LOGIN or AUTHENTICATE command
(5) successful SELECT or EXAMINE command
(6) CLOSE command, or failed SELECT or EXAMINE command
(7) LOGOUT command, server shutdown, or connection closed

- *In this diagram, LOGIN, AUTHENTICATE, SELECT, EXAMINE, CLOSE, and LOGOUT are example commands that the client can issue, while OK is one possible server response.*

- *Other common commands include FETCH, STORE, DELETE, and EXPUNGE, with the obvious meanings. Additional server responses include NO (client does not have permission to perform that operation) and BAD (command is ill formed).*

- *When the user asks to FETCH a message, the server returns it in MIME format and the mail reader decodes it. In addition to the message itself, IMAP also defines a set of message attributes that are exchanged as part of other commands, independent of transferring the message itself.*

- *Message attributes include information like the size of the message, but more interestingly, various flags associated with the message (e.g., Seen, Answered, Deleted, and Recent).*

• These flags are used to keep the client and server synchronized; that is, when the user deletes a message in the mail reader, the client needs to report this fact to the mail server.

• Later, should the user decide to expunge all deleted messages, the client issues an EXPUNGE command to the server,which knows to actually remove all earlier deleted messages from the mailbox.

IMAP4

- *Another mail access protocol is Internet Mail Access Protocol, version 4 (IMAP4).*

- *IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.*

- *POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.).*

- *In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.*

IMAP4 provides the following extra functions:

- *A user can check the e-mail header prior to downloading.*

- *A user can search the contents of the e-mail for a specific string of characters prior to downloading.*

- *A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.*

- *A user can create, delete, or rename mailboxes on the mail server.*

A user can create a hierarchy of mailboxes in a folder for e-mail storag

POP3

- ***Mail access starts with the client when the user needs to download e-mail from the mailbox on the mail server.***

- ***The client opens a connection to the server on TCP port 110.***

- It then sends its user name and password to access the mailbox.

- The user can then list and retrieve the mail messages, one by one.

- POP3 has two modes: the delete mode and the keep mode.

- In the delete mode, the mail is deleted from the mailbox after each retrieval.

- In the keep mode, the mail remains in the mailbox after retrieval. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying.

- The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

World Wide Web (HTTP)

- Any Web browser has a function that allows the user to ―open a URL.‖ URLs (uniform resource locators) provide information about the location of objects on the Web; they look like the following.

http://www.cs.princeton.edu/index.html

- If you opened that particular URL, your Web browser would open a TCP connection to the Web server at a machine called www.cs.princeton.edu and immediately retrieve and display the file called index.html.

- Most files on the Web contain images and text, and some have audio and video clips.

- They also include URLs that point to other files, and your Web browser will have some way in which you can recognize URLs and ask the browser to open them. These embedded URLs are called *hypertext links.*

- When you select to view a page, your browser (the client) fetches the page from the server using HTTP running over TCP. LikeSMTP,

HTTPis a text-oriented protocol.
At its core, each HTTP message has the general form

START_LINE <CRLF>
MESSAGE_HEADER <CRLF>
<CRLF>
MESSAGE_BODY <CRLF>
<CRLF> stands for carriage-return-line-feed.

The first line (START LINE) indicates whether this is a request message or a response message. There are zero or more of these MESSAGE HEADER lines—the set is terminated by a blank line—each of which looks like a header line in an email message.

- HTTP defines many possible header types, some of which pertain to request messages, some to response messages, and some to the data carried in the message body.

- Instead of giving the full set of possible header types, though, we just give a handful of representative examples.

- Finally, after the blank line comes the contents of the requested

- message (MESSAGE BODY); this part of the message is typically empty for request messages.

Request Messages

- The first line of an HTTP request message specifies three things: the operation to be performed, the Web page the operation should be performed on, and the version of HTTP being used.

- Two most widely used request operations are

- GET (fetch the specified Web page) and

- HEAD (fetch status information about the specifiedWeb page).

- The former is obviously used when your browser wants to retrieve and display a Web page.

- The latter is used to test the validity of a hypertext link or to see if a particular page has been modified since the browser last fetched it.

HTTP request operations

Operation	Description
OPTIONS	request information about available options
GET	retrieve document identified in URL
HEAD	retrieve metainformation about document identified in URL
POST	give information (e.g., annotation) to server
PUT	store document under specified URL
DELETE	delete specified URL
TRACE	loopback request message
CONNECT	for use by proxies

For example, the START LINE
GET http://www.cs.princeton.edu/index.html HTTP/1.1 says that the client wants the server on host www.cs.princeton.edu to return the page named index.html.
This particular example uses an *absolute URL. It is also possible*
to use a *relative identifier and specify the host name in one of the MESSAGE HEADER* lines; for example,
GET index.html HTTP/1.1

Host: www.cs.princeton.edu

Response Messages

- Like request messages, response messages begin with a single START LINE.

- In this case, the line specifies the version of HTTP being used, a three-digit code indicating whether or not the request was successful, and a text string giving the reason for the response.

For example, the START LINE
HTTP/1.1 202 Accepted
indicates that the server was able to satisfy the request, while
HTTP/1.1 404 Not Found
indicates that it was not able to satisfy the request because the page was not found.

Five types of HTTP result codes

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

TCP Connections

- The original version of HTTP (1.0) established a separate TCP connection for each data item retrieved from the server.

- It's not too hard to see how this was a very inefficient mechanism: Connection setup and teardown messages had to be exchanged between the client and server even if all the client wanted to do was verify that it had the most recent copy of a page.

- Thus, retrieving a page that included some text and a dozen icons or other small graphics would result in 13 separate TCP connections being established and closed.

- The most important improvement in the latest version of HTTP (1.1) is to allow *persistent connections—the client and server can exchange multiple request/response* messages over the same TCP connection.

- Persistent connections have two advantages

- First, they obviously eliminate the connection setup overhead, thereby reducing the load on the server, the load on the network caused by the additional TCP packets, and the delay perceived by the user.

- Second, because a client can send multiple request messages down a single TCP connection, TCP's congestion window mechanism is able to operate more efficiently.

Caching

- how to effectively cache Web pages.

Caching has many benefits:
- the client's perspective, a page that can be retrieved from a nearby cache can be displayed much more quickly than if it has to be fetched from across the world.

- From the server's perspective, having a cache intercept and satisfy a request reduces the load on the server.

- Caching can be implemented in many different places:

- a user's browser can cache recently accessed pages, and simply display the cached copy if the user visits the same page again.

- ISPs can cache pages

- The machine is caching pages on behalf of the site, and they configure their browsers to connect directly to the caching host. This node is sometimes called a *proxy*.

- ISP router

- This router can peek inside the request message and look at the URL for the requested page. If it has the page in its cache, it returns it.

- If not, it forwards the request to the server and watches for the response to fly by in the other direction.

- When it does, the router saves a copy in the hope that it can use it to satisfy a future request.

- cache needs to make sure it is not responding with an out-of-date version of the page.

- the server assigns an expiration date (the Expires header field) to each page it sends back to the client (or to a cache between the server and client).

- The cache remembers this date and knows that it need not re verify the page each time it is requested until after that expiration date has passed.
- After that time (or if that header field is not set) the cache can use the HEAD or conditional GET operation (GET with If-Modified-Since header line) to verify that it has the most recent copy of the page.

- More generally, there is a set of ―cache directives‖ that must be obeyed by all caching mechanisms along the request/response chain.

- These directives specify whether or not a document can be cached, how long it can be cached, how fresh a document must be,and so on.

DOMAIN NAME SERVICE (DNS)

⦿ Naming service- can be developed to map user-friendly names into router-friendly addresses.

⦿ Name services are sometimes called *middleware because they fill a* gap between applications and the underlying network.

⦿ Host names differ from host addresses in two important ways. First, they are usually of variable length and mnemonic, thereby making them easier for humans to remember.

⦿ (In contrast, fixed-length numeric addresses are easier for routers to process.)

⦿ Second, names typically contain no information that helps the network locate (route packets toward) the host. Addresses, in contrast, sometimes have routing information embedded in them; *flat addresses (those not divisible into component parts)* are the exception.

⦿ First, a *name space defines the set of possible names.*

⦿ A name space can be either *flat (names are not divisible into components) or hierarchi-cal (Unix file names are the obvious example). Second, the naming system maintains* a collection of *bindings of names to values.*

⦿ *The value can be anything we want the* naming system to return when presented with a name; in many cases it is an address.

⦿ a *resolution mechanism is a procedure that, when invoked with a name, returns* the corresponding value. A *name server is a specific implementation of a resolution* mechanism that is available on a network and that can be queried by sending it a message.

⦿ The Internet has a particularly well-developed naming system—**the *domain name system (DNS).***

⦿ Because of its large size, the Internet has a particularly well-developed naming system in place—the *domain name system (DNS).*

⦿ Early in its history, when there were only a few hundred hosts on the Internet, a central authority called the Network Information Center (NIC) maintained a flat table of name-to-address bindings; this table was called hosts.txt.

⦿ Whenever a site wanted to add a new host to the Internet, the site administrator sent email to the NIC giving the new host's name/address pair.

⦿ This information was manually entered into the table, the modified table was mailed out to the various sites every few days, and the system administrator at each site installed the table on every host at the site.

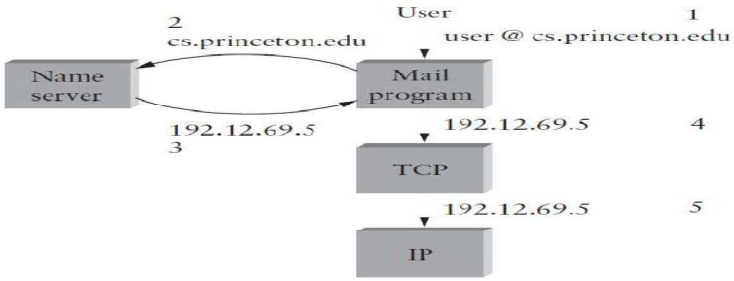
⦿ It should come as no surprise that the hosts.txt approach to naming did not work well as the number of hosts in the Internet started to grow.

⦿ Therefore, in the mid-1980s, the domain naming system was put into place.

⦿ DNS employs a hierarchical name space rather than a flat name space, and the ―table‖ of bindings that implements this name space is partitioned into disjoint pieces and distributed throughout the Internet.

⦿ These subtables are made available in name servers that can be queried over the network.

Names translated into addresses, where the numbers 1–5 show the se- quence of steps in the process.



Domain Hierarchy

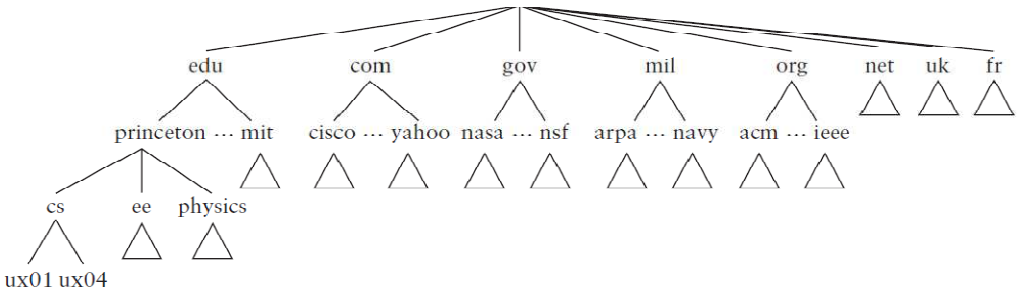
⦿ DNS implements a hierarchical name space for Internet objects.

⦿ DNS names are processed from right to left and use periods as the separator.

⦿ the DNS hierarchy can be visualized as a tree, where each node in the tree corresponds to a domain and the leaves in the tree correspond to the hosts being named.

⦿ There ―big six‖ domains for each country : edu, com, gov, mil, org, and net.

Example of a domain hierarchy



Name Servers

⦿ The first step is to partition the hierarchy into subtrees called *zones*.

⦿ Each zone can be thought of as corresponding to some administrative authority that is responsible for that portion of the hierarchy.

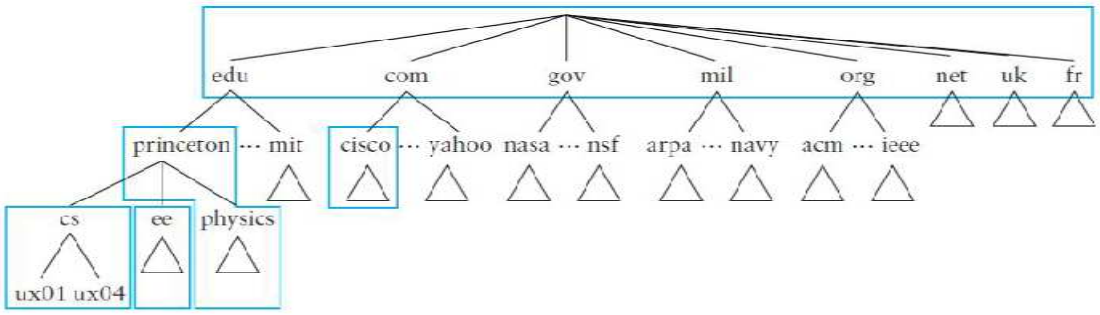
⦿ Specifically, the information contained in each zone is implemented in two or more name servers.

⦿ Each name server, in turn, is a program that can be accessed over the Internet.

⦿ Clients send queries to name servers, and name servers respond with the requested information.

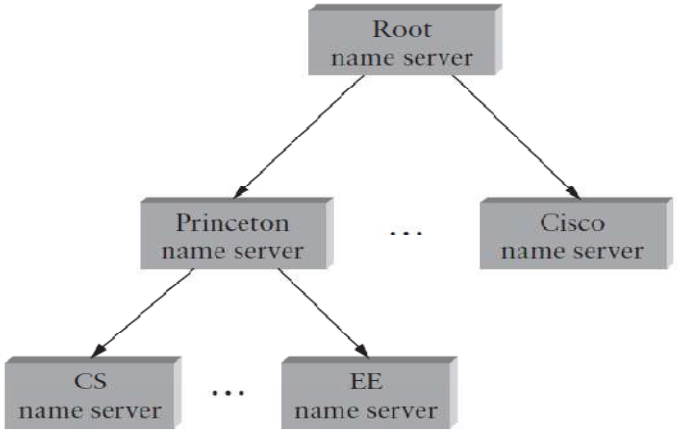
For example, the top level of the hierarchy forms a zone that is managed by the NIC.

Domain hierarchy partitioned into zones



⦿ Sometimes the response contains the final answer that the client wants, and sometimes the response contains a pointer to another server that the client should query next.

Hierarchy of name servers



⦿ Each name server implements the zone information as a collection of *resource records. In essence, a resource record is a name-to-value binding, or more specifically,* a 5-tuple that contains the following fields:
Name, Value, Type, Class, TTL

⦿ The Name and Value fields are exactly what you would expect, while the Type field specifies how the Value should be interpreted.

⦿ For example, Type = A indicates that the Value is an IP address. Thus, A records implement the name-to-address mapping we have been assuming.

⦿ NS: The Value field gives the domain name for a host that is running a name server that knows how to resolve names within the specified domain.

⦿ CNAME: The Value field gives the canonical name for a particular host; it is used to define aliases.

⦿ MX: The Value field gives the domain name for a host that is running a mail server that accepts messages for the specified domain.

⦿ The Class field was included to allow entities other than the NIC to define useful record types.

⦿ To date, the only widely used Class is the one used by the Internet; it is denoted IN. Finally, the TTL field shows how long this resource record is valid.

⦿ It is used by servers that cache resource records from other servers; when the TTL expires, the server must evict the record from its cache.

⦿ First, the root name server contains an NS record for each second-level server. It also has an A record that translates this name into the corresponding IP address.

⦿ Taken together, these two records effectively implement a pointer from the root name server to each of the second-level servers.

princeton.edu, cit.princeton.edu, NS, IN

cit.princeton.edu, 128.196.128.233, A, IN
cisco.com, ns.cisco.com, NS, IN
ns.cisco.com, 128.96.32.20, A, IN
⦿ Next, the domain princeton.edu has a name server available on host cit.princeton edu that contains the following records.

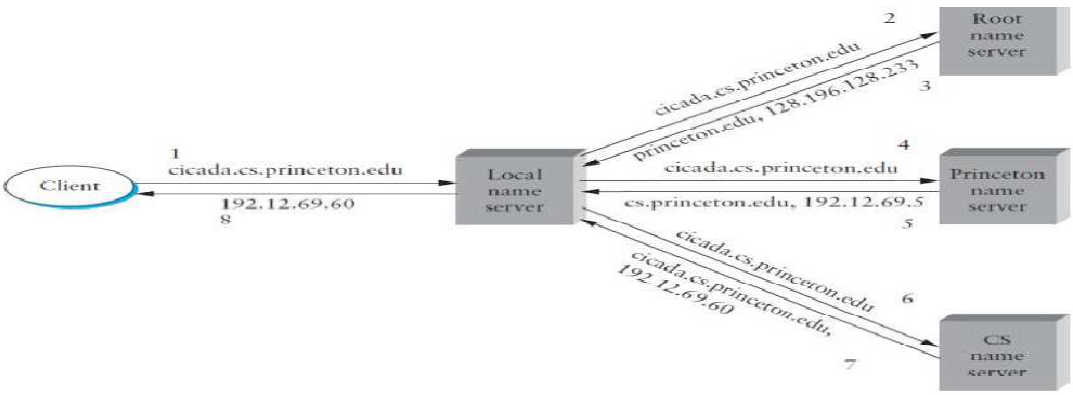
⦿ Note that some of these records give the final answer (e.g., the address for host saturn.physics.princeton.edu), while others point to third-level name servers.

cs.princeton.edu, gnat.cs.princeton.edu, NS, IN
gnat.cs.princeton.edu, 192.12.69.5, A, IN
ee.princeton.edu, helios.ee.princeton.edu, NS, IN
helios.ee.princeton.edu, 128.196.28.166, A, IN
jupiter.physics.princeton.edu, 128.196.4.1, A, IN
saturn.physics.princeton.edu, 128.196.4.2, A, IN
mars.physics.princeton.edu, 128.196.4.3, A, IN
venus.physics.princeton.edu, 128.196.4.4, A, IN
cs.princeton.edu, gnat.cs.princeton.edu, MX, IN
cicada.cs.princeton.edu, 192.12.69.60, A, IN
cic.cs.princeton.edu, cicada.cs.princeton.edu, CNAME, IN
gnat.cs.princeton.edu, 192.12.69.5, A, IN
gna.cs.princeton.edu, gnat.cs.princeton.edu, CNAME, IN
www.cs.princeton.edu, 192.12.69.35, A, IN
cicada.cs.princeton.edu, roach.cs.princeton.edu, CNAME, IN

Name Resolution

⦿ Resolving a name actually involves a client querying the local server, which in turn acts as a client that queries the remote servers on the original client's behalf.

Name resolution in practice, where the numbers 1–8 show the sequence of steps in the process.



Advantages

1.All the hosts in the Internet do not have to be kept up-to-date on where the current root servers are located; only the servers have to know about the root.

2. The local server gets to see the answers that come back from queries that are posted by all the local clients. The local server caches these responses and is sometimes able to resolve future queries without having to go out over the network.

SNMP

- This means we need a protocol that allows us to read, and possibly write, various pieces of state information on different network nodes. The most widely used protocol for this purpose is the Simple Network Management Protocol (SNMP).

- SNMP is essentially a specialized request/reply protocol.

- It supports two kinds of request messages: GET and SET.

- GET- is used to retrieve a piece of state from some node,

SET- is used to store a new piece of state in some node.

- Whenever the administrator selects a certain piece of information that he or she wants to see, the client program uses SNMP to request that information from the node in question.

- An SNMP server running on that node receives the request, locates the appropriate piece of information, and returns it to the client program, which then displays it to the user.

- There is only one complication to this , Exactly how does the client indicate which piece of information it wants to retrieve, and likewise, how does the server know which variable in memory to read to satisfy the request.

- SNMP depends on a companion specification called the management information base (MIB).

The MIB defines the specific pieces of information—the MIB *variables—that you can retrieve from a network node.*

- The current version of MIB, called MIB-II, organizes variables into 10 different *groups.*

- System: general parameters of the system (node) as a whole, including where t he node is located, how long it has been up, and the system's name.

- Interfaces: information about all the network interfaces (adaptors) attached to this node, such as the physical address of each interface, how many packets have been sent and received on each interface.

- Address translation: information about the Address Resolution Protocol (ARP), and in particular, the contents of its address translation table.

- IP: variables related to IP, including its routing table, how many datagrams it has successfully forwarded, and statistics about datagram reassembly.

- Includes counts of how many times IP drops a datagram for one reason or another.

- TCP: information about TCP connections, such as the number of passive and active opens, the number of resets, the number of timeouts, default timeout settings, and so on.

- Per-connection information persists only as long as the connection exists.

- UDP: information about UDP traffic, including the total number of UDP datagrams that have been sent and received.

- Two problems remain.

- First, we need a precise syntax for the client to use to state which of the MIB variables it wants to fetch.

- Second, we need a precise representation for the values returned by the server. Both problems are addressed using ASN.1.

- The MIB uses this identification system to assign a globally unique identifier to each MIB variable

- These identifiers are given in a ―dot‖ notation, not unlike domain names.

- For example, 1.3.6.1.2.1.4.3 is the unique ASN.1 identifier for the IP-related MIB variable ipInReceives; this variable counts the number of IP datagrams that have been received by this node. Thus, network management works as follows. The SNMP client puts the ASN.1 identifier for the MIB variable it wants to get into the request message, and it sends this message to the server.

- Abstract Syntax Notation One (ASN.1) is an ISO standard that defines, among other things, a representation for data sent over a network. The representation-specific part of ASN.1 is called the Basic Encoding Rules (BER).

- The server then maps this identifier into a local variable (i.e., into a memory location where the value for this variable is stored), retrieves the current value held in this variable, and uses ASN.1 BER to encode the value it sends back to the client.

TELNET

- TELNET –is a client/server application program.

- TELNET is an abbreviation for *TErminaL NETwork.*

- It is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO).

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

Timesharing Environment

- TELNET was designed at a time when most operating systems, such as UNIX, were operating in a timesharing environment.

- In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse.

- Even a microcomputer can simulate a terminal with a terminal emulator.

Logging

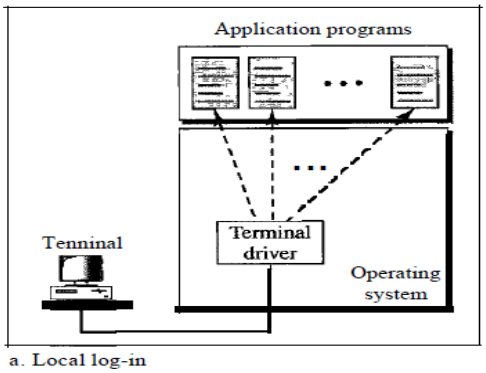
- In a timesharing environment, users are part of the system with some right to access resources.

- Each authorized user has an identification and probably,a password. The user identification defines the user as part of the system.

- To access the system, the user logs into the system with a user id or log-in name.

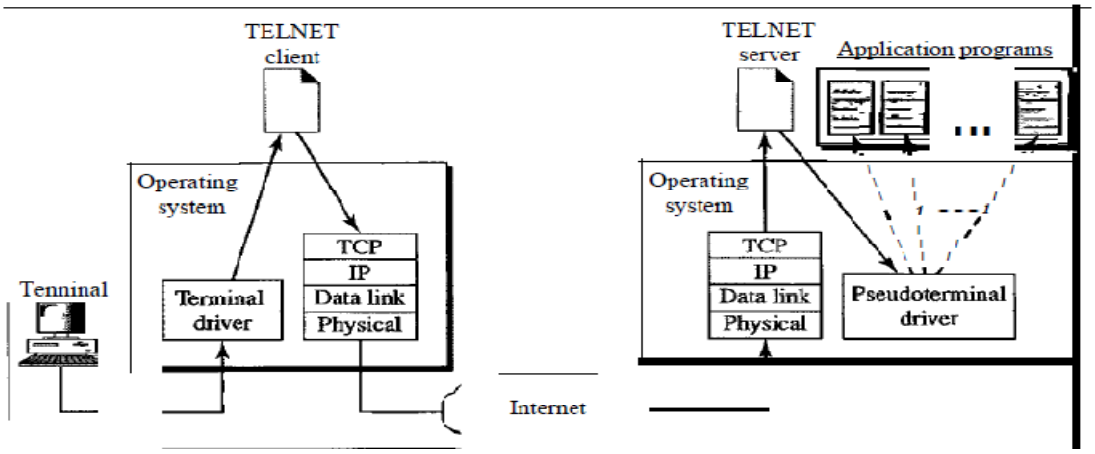
- The system also includes password checking to prevent an unauthorized user from accessing the resources.

LOCAL LOG IN



a. Local log-in

REMOTE LOG IN



LOCAL LOG IN

- When a user logs into a local timesharing system, it is called local log-in. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver.

- The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility .

REMOTE LOG IN

- When a user wants to access an application program or utility located on a remote machine, then it is called remote log-in.

- Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them.

- The characters are sent to the TELNET client, which transforms the characters to a universal character set called *network virtual terminal (NVT) characters and delivers them to the local TCP/IP* protocol stack.

- The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine.

- Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer.

- solution is to add a piece of software called a *pseudo terminal driver which pretends that the characters are coming from* a terminal.

- The operating system then passes the characters to the appropriate application program.

Network Virtual Terminal

- We are dealing with heterogeneous systems.

- If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer.

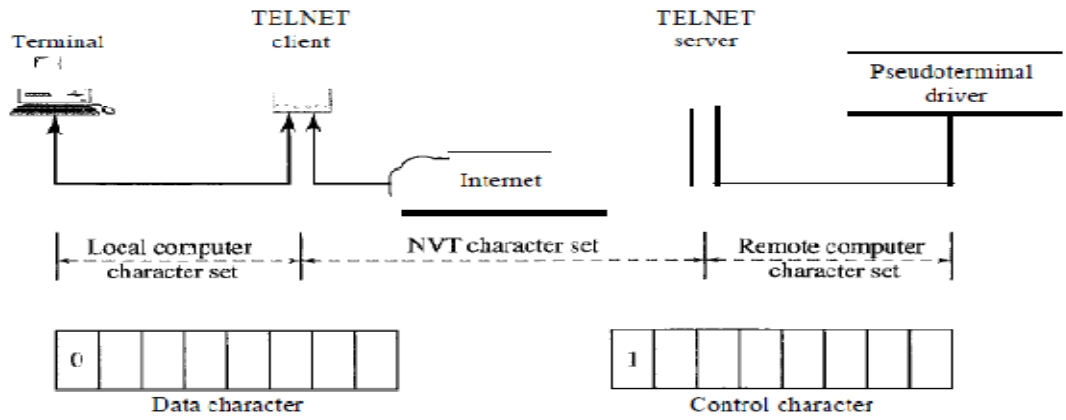
- TELNET solves this problem by defining a universal interface called the network virtual terminal (NVT) character set.

- Via this interface, the client TELNET translates characters

- (data or commands) that come from the local terminal into NVT form and delivers them to the network.

- The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer.

Concept ofNVT



NVT Character Set

- NVT Character Set NVT uses two sets of characters, one for data and the other for control.

- Both are 8-bit bytes. For data, NVT is an 8-bit character set in which the 7 lowest-order bits are the same as ASCII and the highest-order bit is O.

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

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

Embedding

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

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

Typed at the remote terminal

Options

- TELNET lets the client and server negotiate options before or during the use of the service.
- Options are extra features available to a user with a more sophisticated terminal.

Users with simpler terminals can use default features

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

NVT character set for option negotiation

Character	*Decimal*	*Binary*	*Meaning*
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

Mode of Operation

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

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

- Character Mode In the character mode, each character typed is sent by the client to the server. The server normally echoes the character back to be displayed on the client screen.

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

FILE TRANSFER PROTOCOL(FTP):

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).*

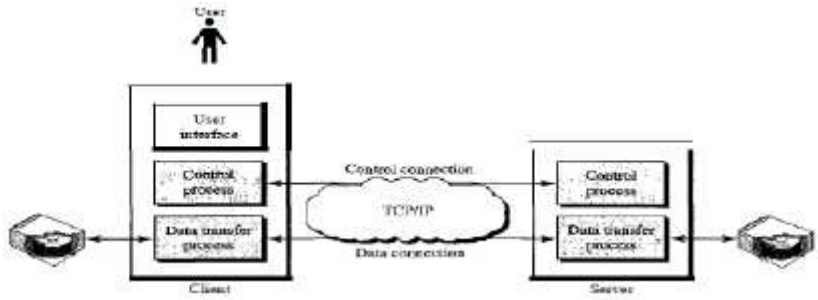
File Transfer Protocol (FTP)

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. Wc 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.

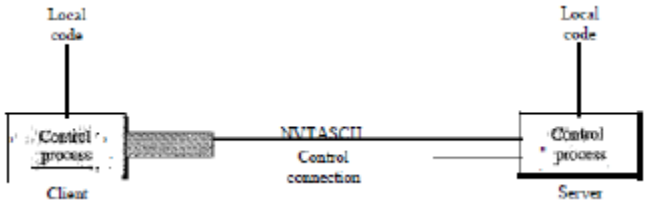
Figure 26.21 shows the basic model of FTP. The client has three components: user interface, client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes.



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

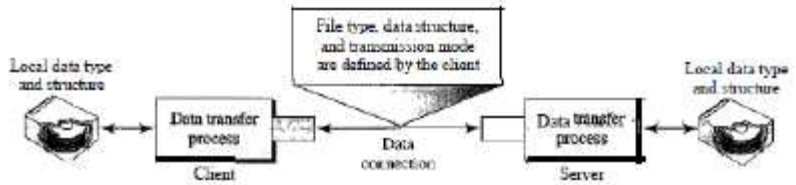
Communication over Control Connection

FTP uses the same approach as SMTP to communicate across the control connection. It uses the 7-bit ASCII character set (see Figure 26.22). 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 command or response is only one short line, so we need not worry about file format or file structure.Each line is terminated with a two-character (carriage return and line feed) end-of-line token.



Communication over 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 aft/e.* 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 aft/e.* 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 (see Figure 26.23).



File Type FTP can transfer one of the following file types across the data connection: an ASCII file, EBCDIC file, or image file. The ASCII file is the default format for transferring text files . Each character is encoded using 7-bit ASCII. The sender transforms the file from its own representation into ASCII characters, and the receiver transforms the ASCII characters to its own representation.

If one or both ends of the connection use EBCDIC encoding (the file format used by IBM), the file can be transferred using EBCDIC encoding.

The image file is the default format for transferring binary files. The file is sent as continuous streams of bits without any interpretation or encoding. This is mostly used to transfer binary files such as compiled programs.

Data Structure FTP can transfer a file across the data connection by using one of the following interpretations about the structure of the data: file structure, record structure, and page structure. In the file structure format, the file is a continuous stream of bytes. Inthe 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.

Transmission Mode FTP can transfer a file across the data connection by using one of the following three transmission modes: stream mode, block mode, and compressed mode. The stream mode is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes.

TCP is responsible for chopping data into segments of appropriate size. If the data are simply a stream of bytes (file structure), no end-of-file is needed. End-of-file in this case is the closing of the data connection by the sender. If the data are divided into records (record structure), each record will have a I-byte endof- record (EOR) character and the end of the file will have a I-byte end-of-file (EOF) character. In block mode, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a 3-byte header.

The first byte is called the *block descriptor;* the next 2 bytes define the size of the block in bytes. In the compressed mode, if the file is big, the data can be compressed. The compression method normally used is run-length encoding. In this method, consecutive appearances of a data unit are replaced by one occurrence and the number of repetitions. In a text file, this is usually spaces (blanks). In a binary file, null characters are usually compressed.

Example 26.4

The following shows an actual *FTP* session for retrieving a list of items in a directory. The colored lines show the responses from the server control connection; the black lines show the commands sent by the client. The lines in white with a black background show data transfer.

```
$ ftp voyager.deanza.tbda.edu
Connected to voyager.deanza.tbda.edu.
220 (vsFTPd 1.2.1)
530 Please login with USER and PASS.
Name (voyager.deanza.tbda.edu:forouzan): forouzan
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> Is reports
227 Entering Passive Mode (153,18,17,11,238,169)
150 Here comes the directory listing.
drwxr-xr-x 23027 411 4096 Sep 24 2002 business
drwxr-xr-x 23027 411 4096 Sep 24 2002 personal
drwxr-xr-x 23027 411 4096 Sep 24 2002 school
226 Directory send OK.
```

ftp>quit
221 Goodbye.
1. After the control connection is created, the FIP server sends the 220 (service ready) response on the control connection.
2. The client sends its name.
3. The server responds with 331 (user name is OK, password is required).
4. The client sends the password (not shown).
5. The server responds with 230 (user log-in is OK).
6. The client sends the list command Os reports) to find the list of files on the directory named report.
7. Now the server responds with 150 and opens the data connection.
8. The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.
9. The client now has two choices. It can use the QUIT command to request the closing of the control connection, or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.
10. After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control connection.
Anonymous FTP
To use FfP, a user needs an account (user name) and a password on the remote server. Some sites have a set of files available for public access, to enable anonymous FTP. To access these files, a user does not need to have an account or password. Instead, the user can use *anonymous* as the user name and *guest* as the password.
User access to the system is very limited. Some sites allow anonymous users only a subset of commands. For example, most sites allow the user to copy some files, but do not allow navigation through the directories.
Example 26.5
We show an example of anonymous FTP. We assume that some public data are available at

intemic.net.
$ ftp intemic.net
Connected to internic.net
220 Server ready
Name: anonymous
331 Guest login OK, send "guest".as password'
.Password~guest ...
-ftp>pwd"
257 *'/'* is current directory
jip>1s
:200-0K
150 Opening ASCII mode
bin
...
ttp> close
221 Goodbye
ftp>quit

Network Security

- The idea of encryption is simple enough:

- The sender applies an *encryption function* to the original *plaintext message, the resulting ciphertext message is sent over the* network.

- and the receiver applies a reverse function (called *decryption) to recover the* original plaintext.

- The encryption/decryption process generally depends on a secret *key* shared between the sender and the receiver.

- This familiar use of cryptography is designed to ensure privacy—preventing the unauthorized release of information.

- Privacy, however, is not the only service that cryptography provides

- *authentication (verifying the identity of the remote* participant) and *integrity (making sure that the message* has not been altered).

three most common cryptographic algorithms:
Data Encryption Standard (DES);
Rivest, Shamir, and Adleman (RSA); and
Message Digest 5 (MD5).
Cryptographic Algorithms

- there are three types of cryptographic algorithms

- *secret key algorithms,*

- *public key algorithms,*

- *and hashing algorithms*

- Secret key algorithms are symmetric in the sense that both participants in the communication share a single key.

- DES (Data Encryption Standard) is the best-known example of a secret key encryption function, while IDEA (International Data Encryption Algorithm) is another.

Secret key encryption

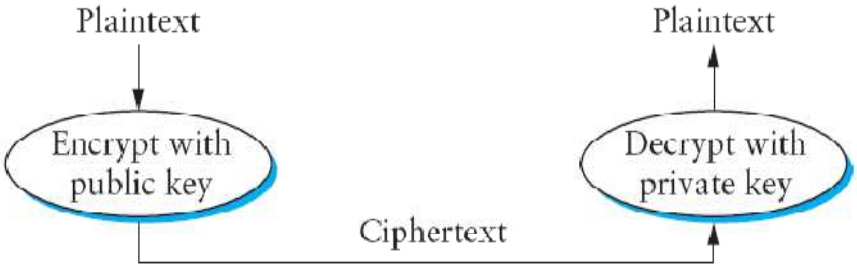
- In contrast to a pair of participants sharing a single secret key, *public key cryptography* involves each participant having a *private key that is shared with no one else* and a *public key that is published so everyone knows it.*

- To send a secure message to this participant, you encrypt the message using the widely known public key.

- The participant then decrypts the message using his or her private key.

- Rivest, Shamir, and Adleman(RSA)—is the best-known public key encryption algorithm.

Public key encryption



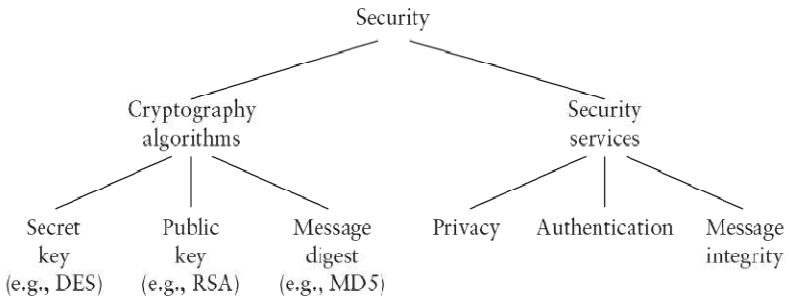
- The third type of cryptography algorithm is called a *hash or message digest* function.

- the idea is to map a potentially large message into a small fixed-length number, analogous to the way a regular hash function maps values from a large space into values from a small space.

- The best way to think of a cryptographic hash function is that it computes a *cryp-tographic checksum over a message. That is, just as a regular checksum protects the* receiver from accidental changes to the message, a cryptographic checksum protects the receiver from malicious changes to the message.

- The most widely used cryptographic checksum algorithm is Message Digest version 5 (MD5).

Taxonomy of network security



Requirements

- The basic requirement for an encryption algorithm is that it be able to turn plaintext into cipher text in such a way that only the intended recipient—the holder of the decryption key—can recover the plaintext.

Secret Key Encryption (DES)

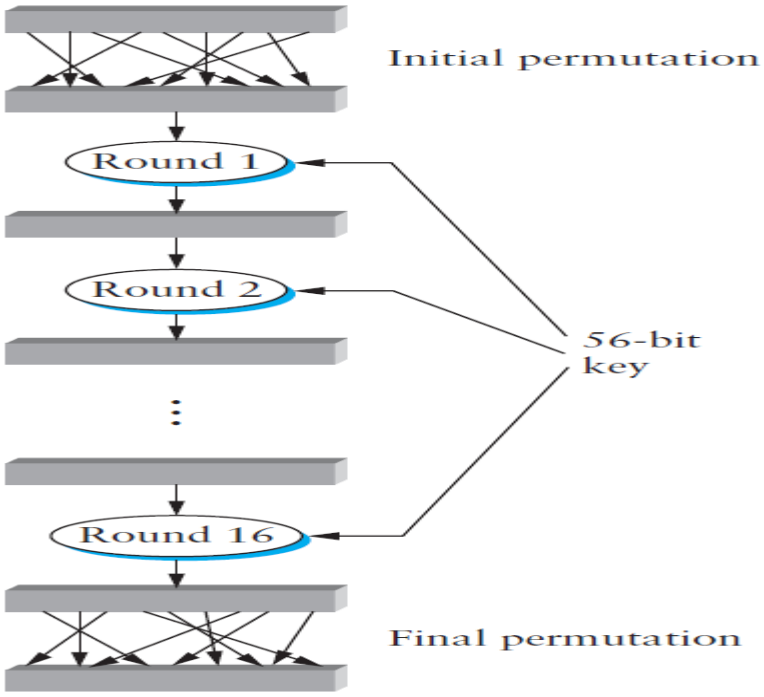
- DES encrypts a 64-bit block of plaintext using a 64-bit key.

- The key actually contains only 56 usable bits—the last bit of each of the 8 bytes in the key is a parity bit for that byte.

DES has three distinct phases

1. The 64 bits in the block are permuted (shuffled).
2.Sixteen rounds of an identical operation are applied to the resulting data and the key.
3. The inverse of the original permutation is applied to the result.

High-level outline of DES.



Initial (and final) DES permutation. Table

Input Position	1	2	3	4	5	...	60	61	62	63	64
Output Position	40	8	48	16	56	...	9	49	17	57	25

- During each round, the 64-bit block is broken into two 32-bit halves, and a different 48 bits are selected from the 56-bit key. If we denote the left and right halves of the block at round *i as Li and Ri , respectively, and the 48-bit keyat round i as Ki* .

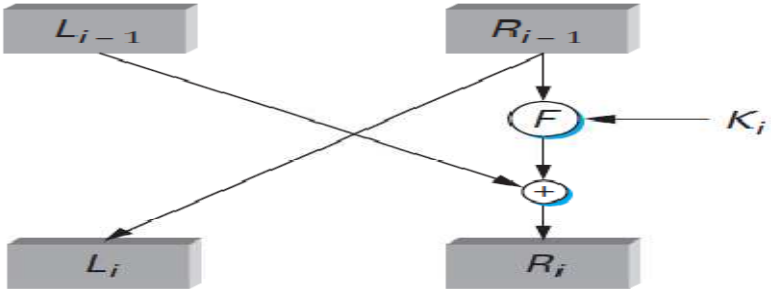
- then these three pieces are combined during round *i according to the following rule:*

Li = Ri−1

Ri = Li−1 ⊕ F (Ri−1, Ki)

- where *F is a combiner function described below and ⊕ is the exclusive-OR (XOR)* operator.

Manipulation at each round of DES.



DES key permutation

Input Position	1	2	3	4	5	...	59	60	61	62	63
Output Position	8	16	24	56	52	...	17	25	45	37	29

Key generation

- Note that every eighth bit is ignored (i.e., bit 64 is missing from the table), reducing the key from 64 bits to 56 bits. Then for each round, the current 56 bits are divided into two 28-bit halves and each half is independently rotated left either one or two bit positions, depending on the round.

- We now need to define function *F and show how each Ki is derived from the n*56-bit key. We start with the key. Initially, the 56-bit key is permuted according to the following table.

- Note that every eighth bit is ignored (i.e., bit 64 is missing from the table), reducing the key from 64 bits to 56 bits. Then for each round, the current 56 bits are divided into two 28-bit halves and each half is independently rotated left either one or two bit positions, depending on the round.
DES key rotation amount per round.

DES compression

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rotation Amount	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES compression permutation

Input Position	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17
Output Position	5	24	7	16	6	10	20	18	12	3	15	23	1	9	19	2

Input Position	19	20	21	23	24	26	27	28	29	30	31	32	33	34	36	37
Output Position	14	22	11	13	4	17	21	8	47	31	27	48	35	41	46	28

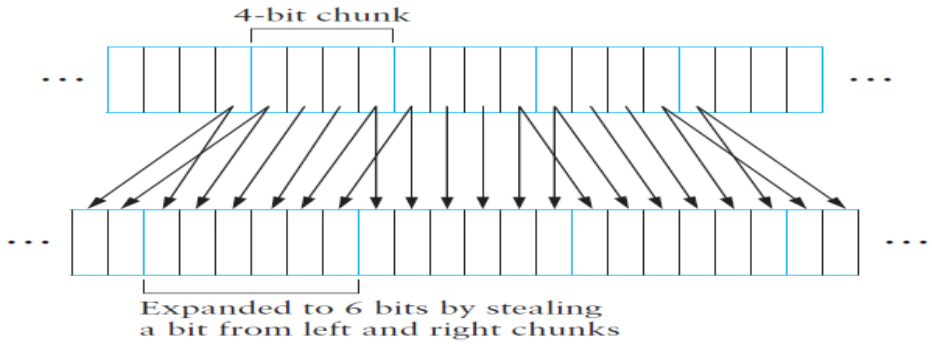
Input Position	39	40	41	42	44	45	46	47	48	49	50	51	52	53	55	56
Output Position	39	32	25	44	37	34	43	29	36	38	45	33	26	42	30	40

- The 56 bits that result from this shift are used both a input for the next round (i.e., the preceding shift is repeated) and to select the 48 bits that make up the key for the current round.

- First, function *F expands R from 32 bits into* 48 bits so that it can be combined with the 48-bit *K. It does this by breaking R into* eight 4-bit chunks and expanding each chunk into 6 bits by stealing the rightmost and leftmost bit from the left and right adjacent 4-bit chunks, respectively.

- Next, the 48-bit *K is divided into eight 6-bit chunks, and each chunk is XORed* with the corresponding chunk that resulted from the previous expansion of *R. Finally*, each resulting 6-bit value is fed through something called a *substitution box (S box)*, which reduces each 6-bit chunk back into 4 bits.

Expansion phase of DES.



Example DES S box (bits 1–6).
There are actually eight different s boxes, one for each of the 6-bit chunks. You can think of an S box as just performing a many-to-one mapping from 6-bit numbers to 4-bit numbers

Input	000000	000001	000010	000011	000100	000101	...
Output	1110	0100	1101	0001	0010	1111	...

Input	...	111010	111011	111100	111101	111110	111111
Output	...	0011	1110	1010	0000	0110	1101

- Notice that the preceding description does not distinguish between encryption and decryption. One of the nice features of DES is that both sides of the algorithm work exactly the same.

- The only difference is that the keys are applied in the reverse order, that is, *K16, K15, . . . , K1*.

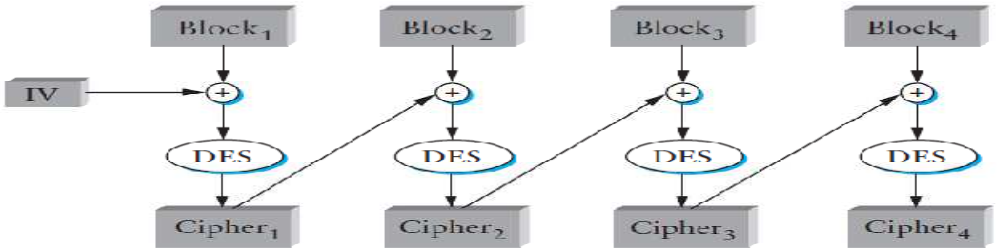
- encrypt a longer message using DES, a technique known as *cipher block chaining (CBC) is typically used.*

- *The idea of CBC is simple: The ciphertext for* block *i is XORed with the plaintext for block i +1 before running it through DES.*

- *An initialization vector (IV) is used in lieu of the nonexistent ciphertext for block 0.*

- *This* vector IV, which is a random number generated by the sender, is sent along with the message so that the first block of plaintext can be retrieved.

Cipher block chaining (CBC) for large messages



- We conclude by noting that there is no published mathematical proof that DES is secure.

- What security it achieves it does through the application of two techniques: confusion and diffusion.

- What we can say is that the only known way to break DES is to exhaustively search all possible 2^56 keys, although

- on average you would expect to have to search only half of the key space, or 2^55 = 3.6×10^16 keys.

- it would take 5.0×10^116μs to break a key.

- For this reason, many applications now use triple-DES (3DES), that is, encrypt the data three times. This can be done with three separate keys, or with two keys: The first is used, then the second, and finally the first key is used again .

Public Key Encryption (RSA)

- it involves different keys for encryption (public key) and decryption (private key)

- RSA commonly uses a key length of 1024 bits.

- The first task is to generate a public and private key.

- To do this, choose two large prime numbers *p and q, and multiply them together to get n.*

- *Both p and q* should be roughly 256 bits long.

- choose the encryption key *e*

$$d = e{-}1 \bmod ((p - 1) \times (q - 1))$$

- The public key is constructed from the pair (*e, n) and the private key is given by the* pair (*d, n).*

Given these two keys, encryption is defined by the following formula:

$$c = m_e \bmod n$$

and decryption is defined by

$m = c_d \bmod n$

where *m is the plaintext message and c is the resulting ciphertext.*

- Note that *m must* be less than *n, which means that it can be no more than the 1024 bits long.*

- *A larger* message is simply treated as the concatenation of multiple 1024-bit blocks.

example

Suppose we pick $p = 7$ *and* $q = 11$

$n = 7 \times 11 = 77$

and

$(p - 1) \times (q - 1) = 60$

so we need to pick a value of *e that is relatively prime to 60. We choose e = 7; 7 and* 60 have no common factor except 1.

$d = 7^{-1} \bmod ((7 - 1) \times (11 - 1))$

which is to say

$7 \times d = 1 \bmod 60$

It turns out that $d = 43$, since

So now we have the public key *e, n = 7, 77 and the private key*

d, n = 43, 77.
Now consider a simple encryption operation. Suppose we wantto encrypt message containing the value 9.
Following the encryption algorithm above:
$c = m^e \bmod n$
= 97 mod 77
= 37
So 37 is the ciphertext that we would send.

- On receipt of the message, the ciphertext would be decrypted as follows:

$m = c^d \bmod n$
$= 37^{43} \bmod 77$
= 9

- Thus, as required, the original message is recovered.

Message Digest Algorithms (MD5)

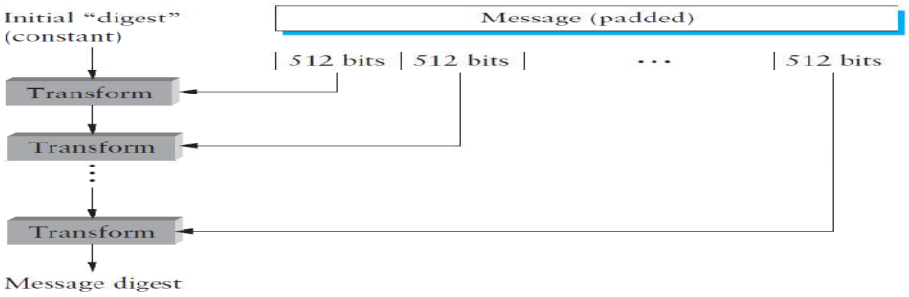
- There are a number of popular message digest algorithms known as MD*n for various* values of *n. MD5 is the most widely used at the time of writing.*

- The secure hash algorithm (SHA) is another well-known message digest function.

- which is to compute a fixed-length cryptographic checksum from an arbitrarily long input message.

- These algorithms operate on a message 512 bits at a time, so the first step is to pad the message to a multiple of 512 bits.

Overview of message digest operation



- The digest calculation begins with the digest value initialized to a constant; this value is combined with the first 512 bits of the message to produce a new value for the digest, using a complex transformation described below; the new value is combined with the next 512 bits of the message using the same transformation, and so on, until the final value of the digest is produced.

- The main ingredient of the MD5 algorithm is thus the transformation that takes as its input the current value of the 128-bit digest, plus 512 bits of message, and outputs a new 128-bit digest.

- So we can think of the current digest value as four 32-bit words (*d0, d1, d2, d3) and the piece of message currently being digested as sixteen* 32-bit words (*m0 through m15).*

$$d_0 = (d_0 + F(d_1, d_2, d_3) + m_0 + T_1) \leftarrow 7$$
$$d_3 = (d_3 + F(d_0, d_1, d_2) + m_1 + T_2) \leftarrow 12$$
$$d_2 = (d_2 + F(d_3, d_0, d_1) + m_2 + T_3) \leftarrow 17$$
$$d_1 = (d_1 + F(d_2, d_3, d_0) + m_3 + T_4) \leftarrow 22$$

$d_0 = (d_0 + F(d_1, d_2, d_3) + m_4 + T_5) \leftarrow 7$
$d_1 = (d_3 + F(d_0, d_1, d_2) + m_5 + T_6) \leftarrow 12$
The T_i *s are constants*

- The second pass looks pretty much the same as the first pass (especially if your eyes are glazing over). The differences are the following

• F is replaced by a slightly different function *G.*

• The constants *T1 through T16 are replaced by another set (T17 through T32).*

- The amount of the left rotation is {5, 9, 14, 20, 5, 9, . . .} at each step.

- Instead of taking the bytes of the message in order m_0 *through* m_5 *, the message* byte that is used at stage *i is m(5i+1)mod16.*

In the third pass:

- *G is replaced by yet another function H, which is just the XOR of its arguments.*

- Another set of constants (*T33 through T48) are used.*

- the amount of the left rotation is {4, 11, 16, 23, 4, 11, . . .} at each step.

The message byte that is used at stage *i is the fourth pass has the following properties:*

- *H is replaced by the function I, which is a combination of bitwise XOR, OR, and NOT on its arguments.*

- *Another set of constants (T49 through T64) are used.*

- *The amount of the left rotation is {6, 10, 16, 21, 6, 10, . . .} at each step.*

- *The message byte that is used at stage i is m(7i)mod16.*

- *m(3i+5)mod16*

Authentication Protocols

- This section describes three common protocols for implementing authentication.

- The first two use secret key cryptography (e.g., DES), while the third uses public key cryptography (e.g., RSA).

Simple Three-Way Handshake

- A simple authentication protocol is possible when the two participants who want to authenticate each other—think of them as a client and a server—already share a secret key.

Three-Way Handshake

- the client first selects a random number *x and encrypts* it using its secret key, which we denote as CHK (client handshake key).

- The client then sends E(*x, CHK), along with an identifier (ClientId), for itself to the server.*

- *The server* uses the key it thinks corresponds to client ClientId (call it SHK for server handshake key) to decrypt the random number. The server adds 1 to the number it recovers and sends the result back to the client.

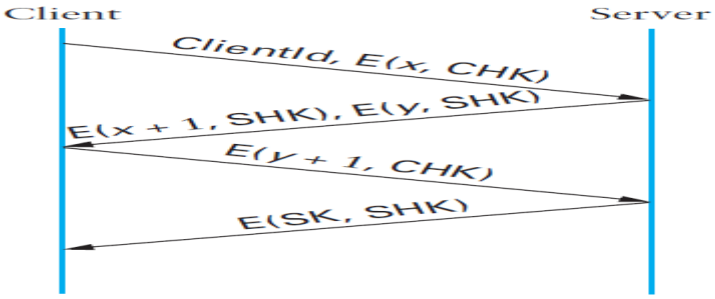
- It also sends back a random number *y that has been* encrypted with SHK.

- Next, the client decrypts the first half of this message and if the result is 1 more than the random number *x that it sent to the server, it knows that* the server possesses its secret key.

- At this point, the client has authenticated the server. The client also decrypts the random number the server sent it (this should yield *y),*

- encrypts this number plus 1, and sends the result to the server. If the server is able to recover *y + 1, then it knows the client is legitimate.*

- After the third message, each side has authenticated itself to the other. The fourth message in corresponds to the server sending the client a session key (SK) ,encrypted using SHK (which is equal to CHK).



Trusted Third Party

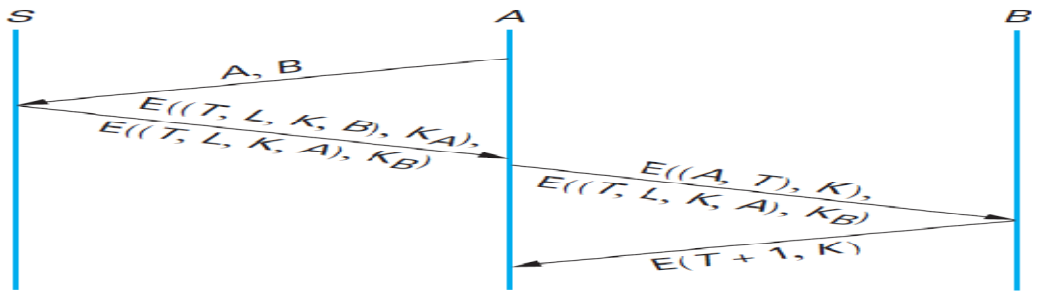
- A more likely scenario is that the two participants know nothing about each other, but both trust a third party. This third party is sometimes called an *authentication server,* and it uses a protocol to help the two participants authenticate each other.

- The one we describe is the one used in Kerberos, a TCP/IP-based security system developed at MIT.

- **In** the following, we denote the two participants who want to authenticate each other as *A and B, and we call the trusted authentication server S.*

- *The Kerberos protocol* assumes that *A and B each share a secret key with S; we denote these two keys as KA* and *KB, respectively.*

Third-party authentication in Kerberos



- As illustrated in Figure participant *A first sends a message to server S that* identifies both itself and *B.*

- *The server then generates a timestamp T, a lifetime L,* and a new session key *K. Timestamp T is going to serve much the same purpose as* the random number in the simple three-way

handshake protocol given above, plus it is used in conjunction with *L to limit the amount of time that session key K is* valid.

- Participants *A and B will have to go back to server S to get a new session* key when this time expires. The idea here is to limit the vulnerability of any one session key.

- Server *S then replies to A with a two-part message. The first part encrypts the* three values *T, L, and K, along with the identifier for participant B, using the key* that the server shares with *A.*

- *The second part encrypts the three values T, L, and K,* along with participant *A's identifier, but this time using the key that the server shares* with *B.*

- Clearly, when *A receives this message, it will be able to decrypt the first* part but not the second part. *A simply passes this second part on to B, along with* the encryption of *A and T using the new session key K.*

- Finally, *B decrypts* the part of the message from *A that was originally encrypted by S, and in so doing,* recovers *T, K, and A.*

- *It uses K to decrypt the half of the message encrypted* by *A and, upon seeing that A and T are* consistent in the two halves of the message, replies with a message that encrypts *T + 1* using the new session key *K*

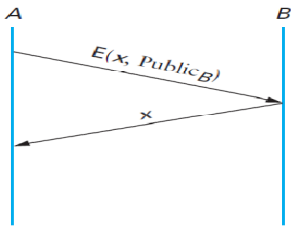
- *A and B can now communicate with* each other using the shared secret session key *K to ensure privacy.*

Public Key Authentication

- Our final authentication protocol uses public key cryptography.

- The public key protocol is a useful one because the two sides need not share a secret key; they only need to know the other side's public key.

- participant *A* encrypts a random number *x using participant B's public key, and B proves it knows* the corresponding private key by decrypting the message and sending *x back to A. A* could authenticate itself to *B in exactly the* same way.



Message Integrity Protocols

- One way to ensure the integrity of a message is to encrypt it using DES with cipher block chaining, and then to use the CBC *residue (the last block output by the* CBC process) as a *message integrity code(MIC).*

The plaintext message plus the MIC would be transmitted to the receiver, with the MIC acting as a sort of checksum—if the receiver could not reproduce the attached MIC using the secret key it shares with the sender, then either the message was not sent by the sender, or it was modified since it was transmitted.

Three alternatives for ensuring message integrity

- The first uses RSA to produce a digital signature.

- The second and third approaches use MD5.

Digital Signature Using RSA

- A *digital signature is a special case of a message* integrity code, where the code can have been generated only by one participant.

- Easiest digital signature algorithm is an RSA signature.

- since a given participant is the only one that knows its own private key, the participant uses this key to produce the signature. Any other participant can verify this signature using the corresponding public key.

- to sign a message, you encrypt it using your private key, and to verify a signature, you decrypt it using the public key of the purported sender.

Keyed MD5

- MD5 produces a cryptographic checksum for a message.

- There are two ways to use MD5 to implement message integrity.

- The first method, which is commonly referred to as *keyedMD5*

- Suppose that we can arrange for the sender and receiver of a message to share a secret key *k. This might be done by preconfiguration of the key, or by some more dynamic* mechanism such as Kerberos.

- The sender then runs MD5 over the concatenation of the message (denoted *m) and this key. In practice, the key k is attached to the end of* the message for the purpose of running MD5; *k is then removed from the message* once MD5 is finished.

- The sender now transmits

m+ MD5(m+ k

- The sender picks *k at random, encrypts it using RSA and* the receiver's public key, and then encrypts the result with it own private key. The result can now be sent to the receiver along with the original message and the MD5 checksum.

m+ MD5(m+ k) + E(E((k, rcv public), snd private)

- The receiver recovers the random key using the purported sender's public RSA key and its own private key, and proceeds to run MD5 on the concatenation of the received message and *k.*

MD5 with RSA Signature

- The sender runs MD5 over the original message it wants to protect, producing an MD5 checksum. It then signs this checksum with its own private RSA key.

- That is, the sender does not sign the entire message, it just signs the checksum. The original message, theMD5checksum, and the RSA signature for the checksum are then transmitted.

m+ E(MD5(m), private

- The receiver verifies the message by

  - running the MD5 algorithm on the received message

  - decrypting the signed checksum with the sender's public key

comparing the two checksums.

- If they match, this means that the message was not modified since the time the sender computed the MD5 checksum and signed it.

Public Key Distribution (X.509)

- Public key cryptography is an extremely powerful technology, but it depends on the distribution of public keys.

- Suppose participant *A wants to convey his public key to participant B. He can't* just use email or a bulletin board to send it, because without *A's public key, B has* no way to authenticate the key as having really come from *A.*

- *Some third party could* send a public key to *B and claim that the message came from A.*

- *If A and B are* individuals who know each other, then they can get together in the same room and *A can give his public key to B directly, perhaps on a business card. However, there* are clear shortcomings to this approach, such as the inability to receive a key from someone unless you can be in the same room with them.

- The basic solution to the problem relies on the use of *digital certificates. The* following sections explain what certificates are and some issues that arise in using them to achieve widespread key distribution.

- Certificates

- A digital signature proves that the data was generated by the owner of a certain key and that it has not been modified since it was signed.

- A certificate is just a special type of digitally signed document.

- The document says, in effect, ―I certify that the public key in this document belongs to the entity named in this document, signed *X." X in this case could be anyone with* a public key.

- It is commonly the case that *X would be a certification authority (CA),4* that is, an administrative entity that is in the business of issuing certificates.

- It should be clear that this certificate is only useful to a participant who already holds the public key for *X because that key is needed to verify the signature.*

- Thus, certificates do not in themselves solve the key distribution problem, but they give us a way to make inroads on it.

- Clearly, once you have a public key for one entity *X, you can start to accumulate* more public keys from other participants if those participants can get certificates issued by *X.*

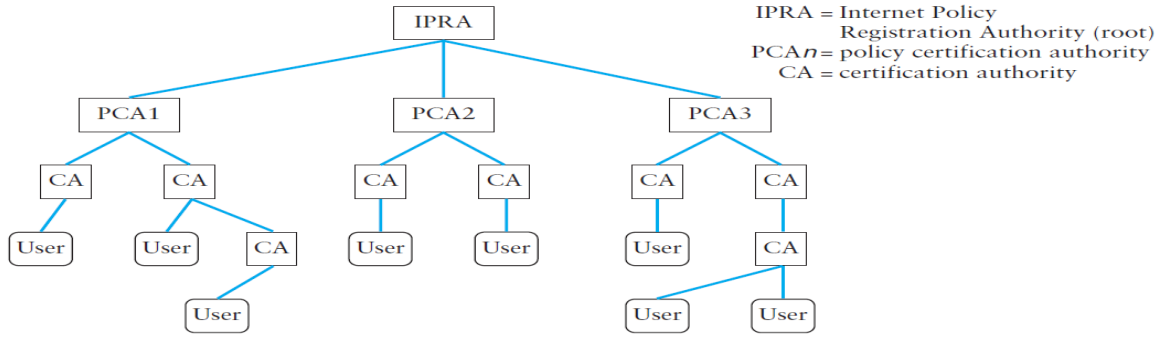
- The idea of certificates allows the building of ―chains of trust.‖

- If *X certifies that* a certain public key belongs to *Y, and then Y goes on to certify that another public* key belongs to *Z, then there exists a chain of certificates from X to Z, even though X* and *Z may have never met.*

- *If Z wants to provide his public key to A, he can provide* the complete chain of certificates—the certificate for *Y's public key issued by X, and* the certificate for *Z's key issued by Y.*

- *If A has the public key for X, he can use the* chain to verify that the public key of *Z is legitimate.*

With this idea of building chains of trust, public key distribution becomes somewhat more tractable.



- There are still significant issues with building chains of trust. First of all, even if you are certain that you have the public key of the root CA, you need to be sure that every CA from the root on down is doing its job properly.

- If some CA is willing to issue certificates to individuals without verifying their identity, then what looks like a valid chain of certificates becomes meaningless.

- A different approach to this problem, in which chains of trust form arbitrary meshes rather than a rigid tree.

- One of the major standards for certificates is known as X.509. This standard leaves a lot of details open, but specifies a basic structure for certificates.

- Components of a certificate clearly must include

- the name of the entity being certified

- the public key of the entity

- the name of the certificate authority

- a digital signature

Certificate Revocation

- One issue that arises with certificates is how to revoke, undo, a certificate.

- The basic solution to the problem is simple enough. A certification authority can issue a *certificate revocation list (CRL), which is a digitally signed list of certificates* that have been revoked.

- The CRL is periodically updated and made publicly available.

- Because it is digitally signed, it can just be posted on a bulletin board. Now, when participant *A receives a certificate for B that he wants to verify, A will first consult* the latest CRL issued by the CA. As long as the certificate has not been revoked, it is valid.

Pretty Good Privacy (PGP)

- Pretty Good Privacy (PGP) is a popular approach to providing encryption and authentication capabilities for electronic mail.

- PGP provides authentication, confidentiality, data integrity and non-repudiance.

- PGP has become quite popular in the networking community.

- PGP key signing parties are a regular feature of IETF meetings.

- Systems that operate at the application layer include Pretty Good Privacy (PGP), which provides secure electronic mail, and Secure Shell (SSH), a secure remote login facility.

- In between these are a number of protocols that operate at the transport layer, notably the IETF's Transport Layer Security (TLS) standard and the older protocol from which it derives, SSL (Secure Socket Layer).

- The following sections describe the salient features of each of these approaches.

PGP has become quite popular in the networking community, and PGP key signing parties are a regular feature of IETF meetings. At these gatherings, an individual can

collect public keys from others whose identity he knows

provide his public key to others

get his public key signed by others, thus collecting certificates that will be persuasive to an increasingly large set of people

sign the public key of other individuals, thus helping them build up their set of certificates that they can use to distribute their public keys

collect certificates from other individuals whom he trusts enough to sign keys

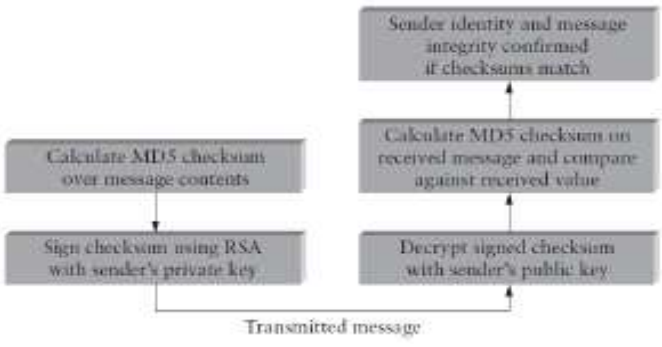
Example of how PGP works

- Now suppose user *A wants to send a message to user B and prove to B that it truly came* from *A.*

PGP follows the following sequence of steps

1. *A digitally signs the message using its private key.*
2. Then the PGP application generates a one time session key using which A's message is encrypted.
3. The session key is encrypted using B's public key.
4. The encrypted message and encrypted session key are encoded using base64 encoding mechanism to get the ASCII compatible representation.
5. This is then sent to the receiver B.
6. The above steps are performed in reverse order to get the original message

PGP message integrity and authentication.

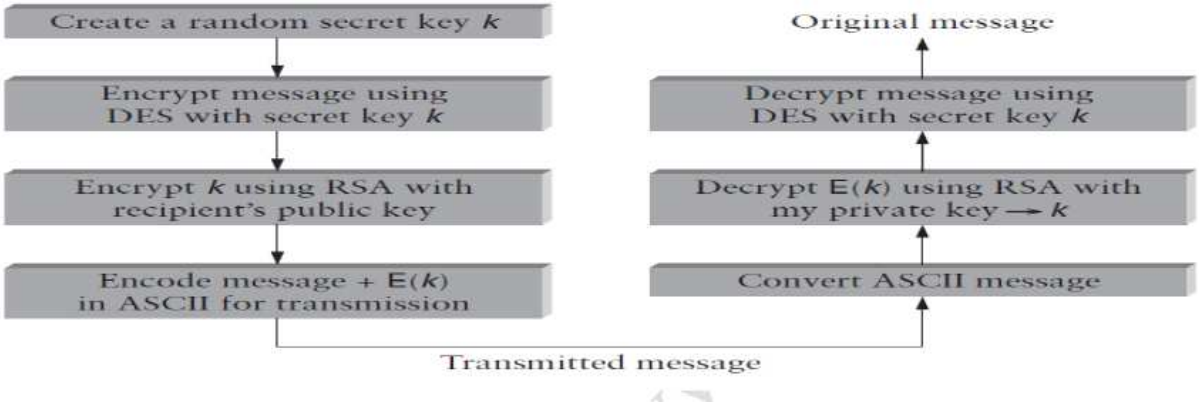


Encryption of a message is equally straightforward and is summarized in the following ,*A* randomly picks a per-message key that is used to encrypt the message using a symmetric algorithm such as DES.

The per-message key is encrypted using the public key of the recipient. PGP obtains this key from *A*'s key ring and notifies *A* of the level of trust he has assigned to this key.

The message is encoded to prevent damage by mail gateways and sent to *B*. On receipt, *B* uses his private key to decrypt the per-message key, and then uses the appropriate algorithm to decrypt the message.

PGP message encryption.



Secure Shell (SSH)

- The Secure Shell (SSH) provides security to a remote login service.

- It is intended to replace the less secure Telnet and rlogin programs used in the early days of the Internet SSH is most often used to provide strong client/server authentication.

- The SSH client runs on the user's desktop machine and the SSH server runs on some remote machine that the user wants to log into.

- but it also supports message integrity and confidentiality

- When the users login, both their passwords and all the data they send or receive potentially passes through countless un trusted networks.

- SSH provides a way to encrypt the data sent over these connections and to improve the strength of the authentication mechanism they use to login.

- The latest version of SSH, version 2, consists of three protocols:

1. SSH-TRANS: a transport layer protocol
2. 2. SSH-AUTH: an authentication protocol
3. 3. SSH-CONN: a connection protocol
4. SSH-TRANS

- It provides an encrypted channel between the client and server machines.

- It runs on top of a TCP connection.

  - Any time a user uses SSH to log onto a remote machine, the first step is to set up an SSH-TRANS channel between those two machines.

  - The two machines establish this secure channel by first having the client authenticate the server using RSA.

  - Once authenticated, the client and server establish a session key that they will use to encrypt any data sent over the channel.

  Also, SSH-TRANS includes a message integrity check of all data exchanged over the channel.

3DES is commonly selected.

- **How does the client possess the server's public key that it needs to authenticate the server?**

- The server tells the client its public key at connection time.

- The first time a client connects to a particular server, SSH warns the user that it has never talked to this machine before and asks if the user wants to continue.

- Although it is a risky thing to do, because SSH is effectively not able to authenticate the server, users often say ―yes‖ to this question.

- SSH then remembers the server's public key, and the next time the user connects to that same machine, it compares this saved key with the one the server responds with.

- If they are the same, SSH authenticates the server.

- If they are different, however, SSH again warns the user that something is amiss, and the user is then given an opportunity to abort the connection

SSH-AUTH

- The next step is for the user to actually log onto the machine, or more specifically, authenticate him- or herself to the server.

SSH allows three different mechanisms for doing this.

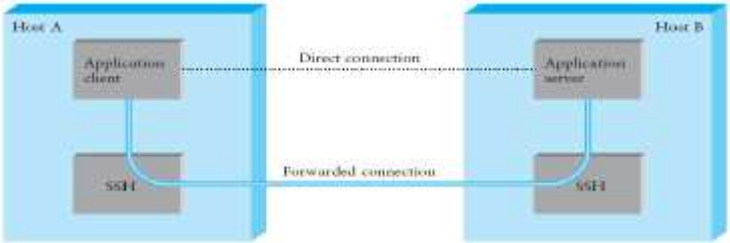
1. First, since the two machines are communicating over a secure channel, it is OK for the user to simply send his or her password to the server.
2. The second mechanism uses public key encryption. This requires that the user has already placed his or her public key on the server.
- 3. The third mechanism, called host-based authentication, basically says that any user claiming to be so-and-so from a certain set of trusted hosts is automatically believed to be that same user on the server.

- Finally, SSH has proven so useful as a system for securing remote login that it has been extended to also support other insecure TCP-based applications.

- The idea is to run these applications over a secure ―SHH tunnel.‖ This capability is called ***port forwarding, and it uses the SSH-CONN protocol.***

- ***The idea is illustrated in*** Figure where we see a client on host A indirectly communicating with a server on host B by forwarding its traffic through an SSH connection.

Using SSH port forwarding to secure other TCP-based applications. The mechanism is called port forwarding because when messages arrive at the well known SSH port on the server, SSH first decrypts the contents, and then ―forwards‖ the data to the actual port at which the server is listening.



**********ALL THE BEST**********