UNIT V
APPLICATION LAYER

Traditional applications -Electronic Mail (SMTP, POP3, IMAP, MIME) – HTTP – Web Services – DNS - SNMP

TRADITIONAL APPLICATIONS

- World Wide Web and email are traditional applications
- Both of these applications use the request/reply paradigm
- Users send requests to servers, which then respond accordingly.
- These are refered as "traditional" applications because they typify the sort of applications that have existed since the early days of computer networks
- 1. It is important to distinguish between application *programs and application protocols.*
- For example, *the HyperText Transport Protocol* (HTTP) is an application protocol that is used to retrieve Web pages from remote servers.
- There can be many different application programs—that is, Web clients like Internet Explorer, Chrome, Firefox, and Safari—that provide users with a different look and feel, but all of them use the same HTTP protocol to communicate with Web servers over the Internet.
- Two very widely-used, standardized application protocols:
- SMTP: Simple Mail Transfer Protocol is used to exchange electronic mail.
- HTTP: HyperText Transport Protocol is used to communicate between Web browsers and Web servers.
- 2. The application protocols described in this section follow the same request/reply communication pattern, eventhougt they would be built on top of a Remote Procedure Call (RPC) transport protocol.
- 3. Many application layer protocols, including HTTP and SMTP, have a companion protocol that specifies the format of the data that can be exchanged.
- **Electronic Mail (SMTP, MIME, IMAP)**
- 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 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

- MIME consists of three basic pieces.
- The first piece is a collection of header lines.
- 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,
- The third piece is a way to encode the various data types so they can be shipped in an ASCII email message.
- MIME uses a straightforward encoding of binary data into the ASCII character set
- The encoding is called base64. The idea is to map every three bytes of the original binary data into four ASCII characters.
- This is done by grouping the binary data into 24-bit units and breaking each such unit into four 6-bit pieces. Each 6-bit piece maps onto one of 64 valid ASCII characters; for example, 0 maps onto A, 1 maps onto B, and so on.

- A MIME message that consists of regular text only can be encoded using 7-bit ASCII. There's also a readable encoding for mostly ASCII data.

Message Transfer(SMTP)

- In the early days of the Internet, users typically logged into the machine on which their *mailbox* resided,
- The mail reader they invoked was a local application program that extracted messages from the file system.
- Today, of course, users remotely access their mailbox fromtheir laptop or smartphone;
- they do not first log into the host that stores their mail (a mail server).
- mail transfer protocol, such as POP or IMAP, is used to remotely download email from a mail server to the user's device.
- There is a *mail daemon* (or process) running on each host that holds a mailbox. This process, also called a *message transfer agent* (MTA), as playing the role of a post office
- Users (or their 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.
- The daemon puts incoming messages into the user's mailbox (where that user's mail reader can later find them).
- The MTA on a sender's machine establishes an SMTP/TCP connection to the MTA on the recipient's mail server,
- The mail traverses one or more *mail gateways* on its route from the sender's host to the receiver's host.
- These gateways also run a message transfer agent process. It's not an accident that these intermediate nodes are called *gateways*, Their job is to store and forward email messages,
- Like an "IP gateway" (*router*) stores and forwards IP datagrams.
- The only difference is that a mail gateway typically buffers messages on disk and is willing to try retransmitting them to the next machine for several days,
- while an IP router buffers datagrams in memory and is only willing to retry transmitting them for a fraction of a second. Figure 9.1 illustrates a two-hop path from the sender to the receiver.
- Independent SMTP connection is used between each host to move the message closer to the recipient.
- 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.

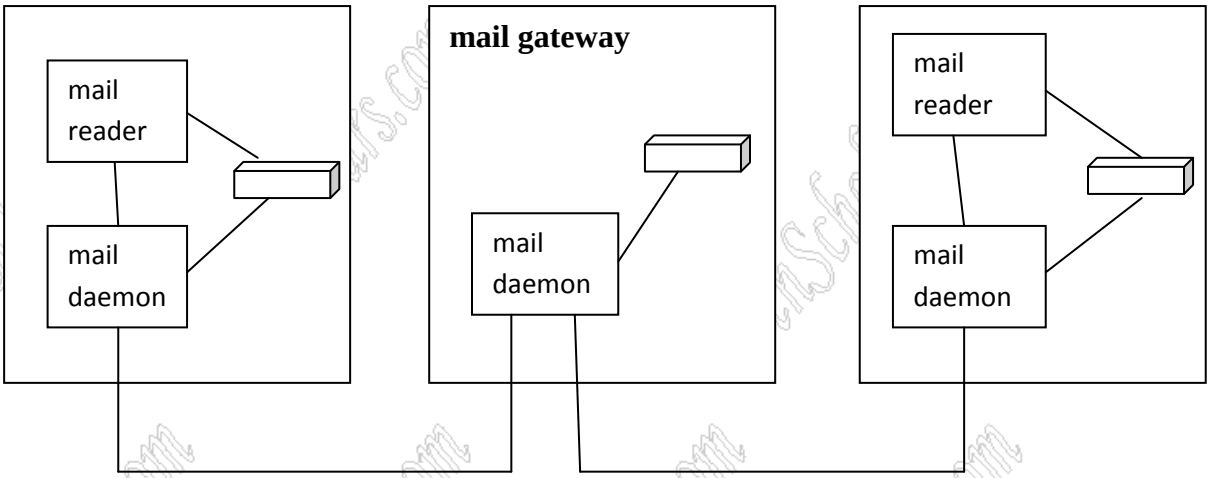


FIGURE 9.1 Sequence of mail gateways store and forward email messages.

- SMTP involves a sequence of exchanges between the client and the server. In each exchange, the client posts a command (e.g., HELO, MAIL, RCPT, DATA, QUIT)
- The server responds with a code (e.g., 250, 550, 354, 221). The server also returns a human-readable explanation for the code (e.g., No such user here).
- The server can respond to a client's RCPT command with a 251 code, which indicates that the user does not have a mailbox on this host,
- but that the server promises to forward the message onto another mail daemon. In other words, the host is functioning as a mail gateway.
- The client can issue a VRFY operation to verify a user's email address, but without actually sending a message to the user

Mail Reader(IMAP)

- 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.
- This reader was a program running on the same machine as the user's mailbox, in which case it could simply read and write the file that implements the mailbox.
- the user accesses his or her mailbox from a remote machine using yet another protocol, such as POP or IMAP.
- IMAP is a client/server protocol running over TCP, where the client (running on the 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.

- The exchange begins with the client authenticating him- or herself and identifying the mailbox

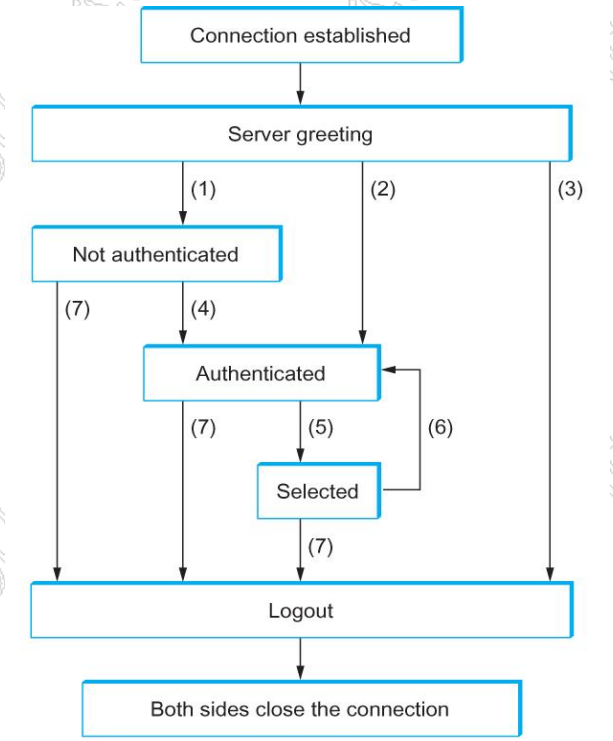


FIGURE 9.2 IMAP state transition diagram.

- 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
- Additional server responses include NO (client does not have permission to performthat operation) and BAD (command is ill formed).
- 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 and,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;
- 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.

NAME SERVICE(DNS)

- The name service is used to translate host names into host addresses;
- The application allows the users of other applications to refer to remote hosts by name rather than by address.
- A name service is usually used by other applications, rather than by humans.
- Naming service can 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.
    - They are usually of variable length and mnemonic, (In contrast, fixed-length numeric addresses are easier for routers to process.)
    - 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;
- ***name space*** defines the set of possible names. A name space can be either *flat* (names are not divisible into components) or *hierarchical*
- 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
- ***resolution mechanism*** is a procedure that, when invoked with a name, returns the corresponding alue.
- 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.
- 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- toaddress 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,
- The system administrator at each site installed the table on every host at the site.
- Name resolution was then simply implemented by a procedure that looked up a host's name in the local copy of the table and returned the corresponding address.
- DNS employs a hierarchical namespace rather than a flat name space,
- 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.
- In the Internet is that a user presents a host name to an application program (possibly embedded in a compound name such as an email address or URL),
- This program engages the naming system to translate this name into a host address.
- The application then opens a connection to this host by presenting some transport protocol (e.g., TCP) with the host's IP address.
- This situation is illustrated (in the case of sending email) in Figure 9.14.

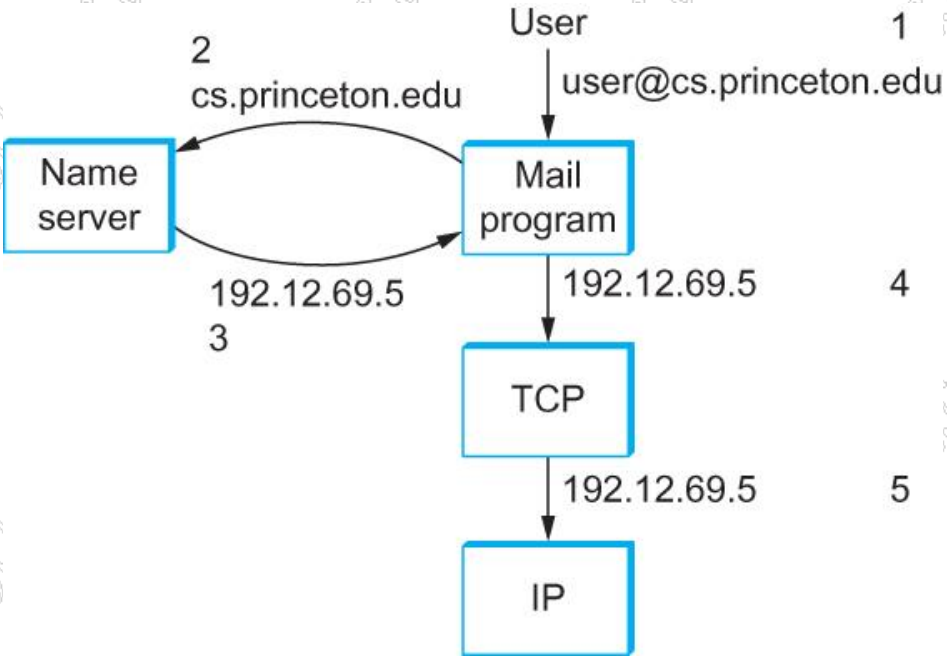


FIGURE 9.14 Names translated into addresses, where the numbers 1 to 5 show the sequence of steps in the process.

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.
- DNS is not strictly used to map host names into host addresses. but DNS maps domain names into values.
- 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.

- Figure 9.15 gives an example of a domain hierarchy.
- The hierarchy is not very wide at the first level consists of domains for each country, plus the "big six" domains: .edu, .com, .gov, .mil, .org, and .net.
- The newer top-level domains include .biz, .coop, and .info.

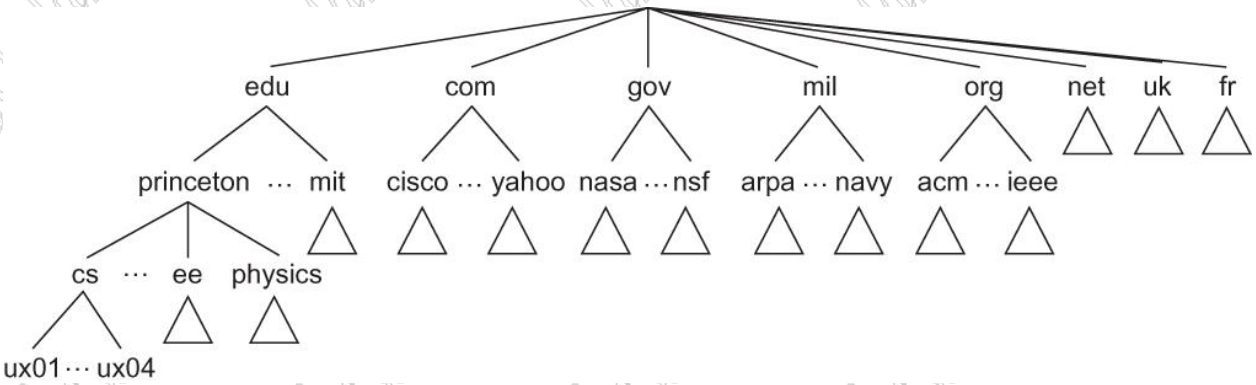


FIGURE 9.15 Example of a domain hierarchy.

Name Servers

- Hierarchy is actually implemented by partition the hierarchy into subtrees called *zones*.
- Figure 9.16 shows how the hierarchy given in Figure 9.15 might be divided into zones.
- Each zone corresponds to some administrative authority that is responsible for that portion of the hierarchy.
- The top level of the hierarchy forms a zone that is managed by the Internet Corporation for Assigned Names and Numbers (ICANN).

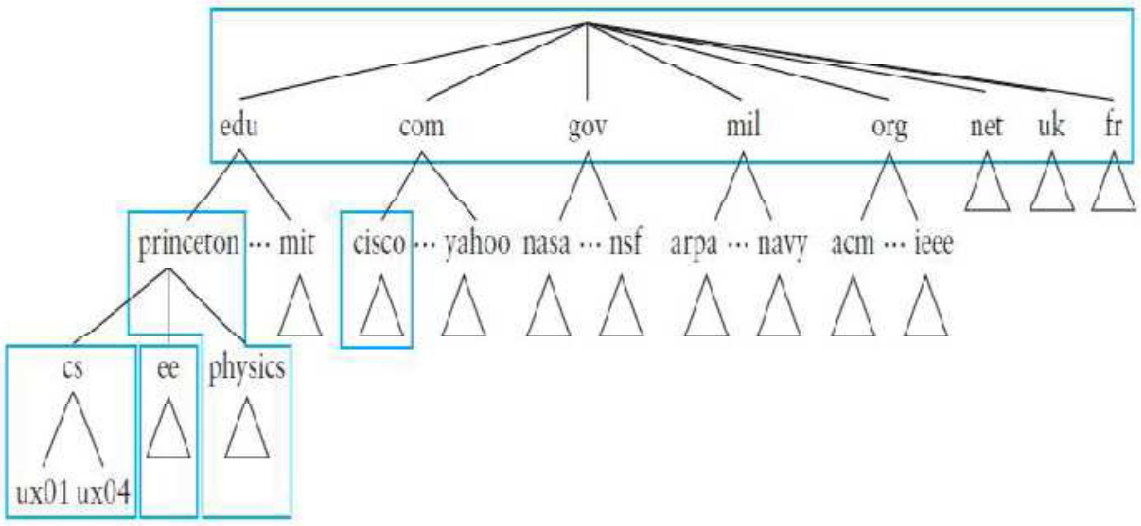


FIGURE 9.16 Domain hierarchy partitioned into zones.

- 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.
- 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.
- Each zone is implemented in two or more name servers for the sake of redundancy; that is, the information is still available even if one name server fails.
- Each name server implements the zone information as a collection of *resource records*.
- 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 hostname and address
- The Type field specifies how the Value should be interpreted.
- Type =A indicates that the Value is an IP address. Thus, A records implement the name-to-address mapping
- 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.
- The time-to-live (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.
- A root name server contains an NS record for each top-level domain (TLD) name server. This identifies a server that can resolve queries for this part of the DNS hierarchy (.edu and .com in this example).
- It also has A records that translates these names into the corresponding IP addresses. Taken together, these two records effectively implement a pointer from the root name server to one of the TLD servers.

(edu,a3.nstld.com,NS,IN)
(a3.nstld.com,192.5.6.32,A,IN)
(com,a.gtld-servers.net,NS,IN)
(a.gtld-servers.net, 192.5.6.30,A,IN)
...

- A third-level name server, such as the one managed by domain cs.princeton.edu, contains A records for all of its hosts. It might also define a set of aliases (CNAME records) for each of those hosts. Aliases are sometimes just convenient (e.g., shorter) names for machines
- The mail exchange (MX) records serve the same purpose for the email application—they allow an administrator to change which host receives mail on behalf of the domain without having to change everyone's email address.

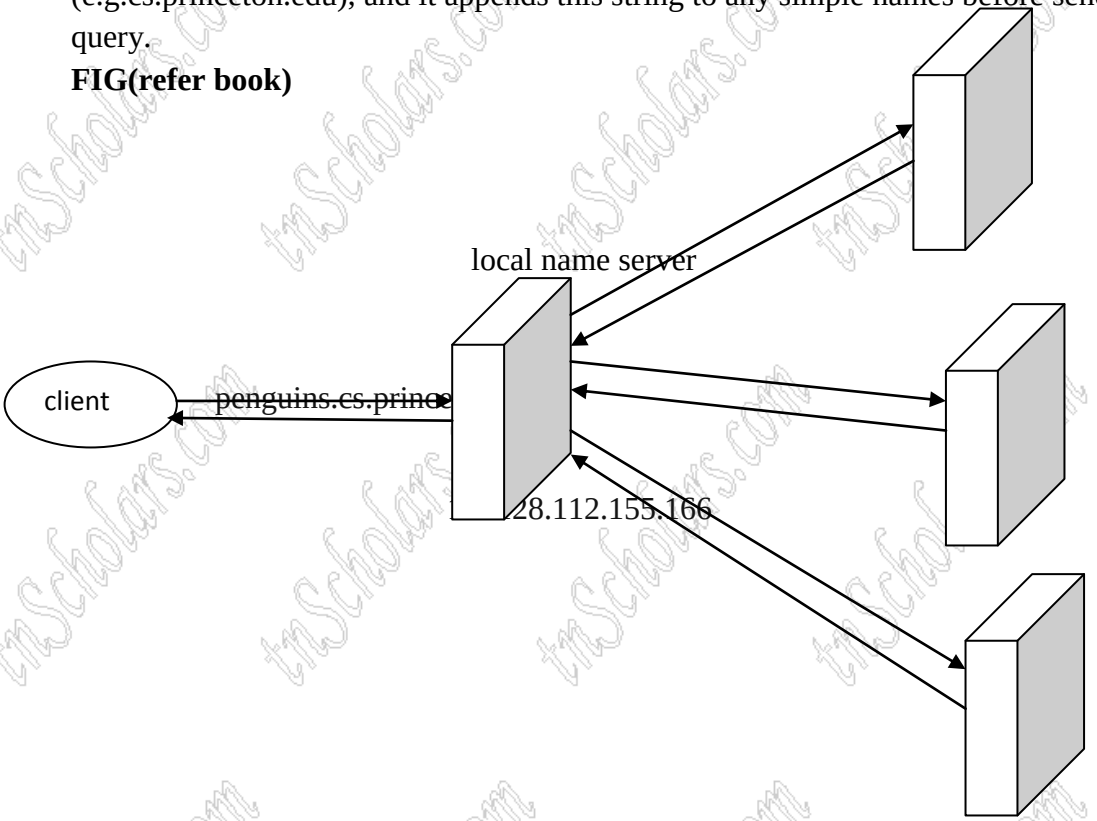
  (penguins.cs.princeton.edu,128.112.155.166,A,IN)
  (www.cs.princeton.edu,coreweb.cs.princeton.edu,CNAME,IN)
  (coreweb.cs.princeton.edu,128.112.136.35,A,IN)
  (cs.princeton.edu, mail.cs.princeton.edu,MX,IN)
  (mail.cs.princeton.edu,128.112.136.72,A,IN)
  ...
- DNS is typically used to name hosts (including servers) and sites. It is not used to name individual people or other objects like files or directories

Name Resolution

- To resolve the name The client could first send a query containing this name to one of the root servers.
- The root server, unable to match the entire name but returns the best match it has
- The name-to-address mapping for one or more root servers is published through some means outside the naming system itself.
- Not all clients know about the root servers. Instead, the client program running on each Internet host is initialized with the address of a *local* name server
- Resolving a name 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.
- This results in the client/server interactions illustrated in Figure 9.18.
- One advantage of this model is that 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.
- A second advantage is that 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.
- The TTL field in the resource records returned by remote servers indicates how long each record can be safely cached.
- This caching mechanism can be used further up the hierarchy as well, reducing the load on the root and TLD servers.

- The system works when a user submits a partial name (e.g., penguins) rather than a complete domain name
- The client program is configured with the local domain in which the host resides (e.g.cs.princeton.edu), and it appends this string to any simple names before sending out a query.

FIG(refer book)



WORLD WIDE WEB(HTTP)

- The original goal of the Web was to find a way to organize and retrieve information, drawing on ideas about hypertext—interlinked documents
- The core idea of hypertext is that one document can link to another document, and the protocol (HTTP) and document language (HTML) were designed to meet that goal.
- Web is as a set of cooperating clients and servers, all of whom speak the same language: HTTP.
- people are exposed to the Web through a graphical client program, or Web browser, like Safari, Chrome, Firefox or Internet Explorer.

- if you want to organize information into a system of linked documents or objects, you need to be able to retrieve one document to get started.
- Hence, any Web browser has a function that allows the user to obtain an object by "opening a URL."
- URLs (Uniform Resource Locators) are so familiar to most of us by now that it's easy to forget that they haven't been around forever.
- They provide information that allows objects on the Web to be located, and they look like the following:

  http://www.cs.princeton.edu/index.html
- If particular URL is opened , 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 many have other objects such as audio and video clips, pieces of code, etc.
- They also frequently include URLs that point to other files that may be located on other machines, which is the core of the "hypertext" part of HTTP and HTML.
- To view a page, browser (the client) fetches the page from the server using HTTP running over TCP.
- Like SMTP, HTTP is a text oriented protocol.
- At its core, HTTP is a request/response protocol, where every 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.
- 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.
- Although HTTP defines a wide assortment of possible request operations—including "write" operations that allow a Web page to be posted on a server—the two most common operations are GET (fetch the specified Web page) and HEAD (fetch status information about the specified Web page).

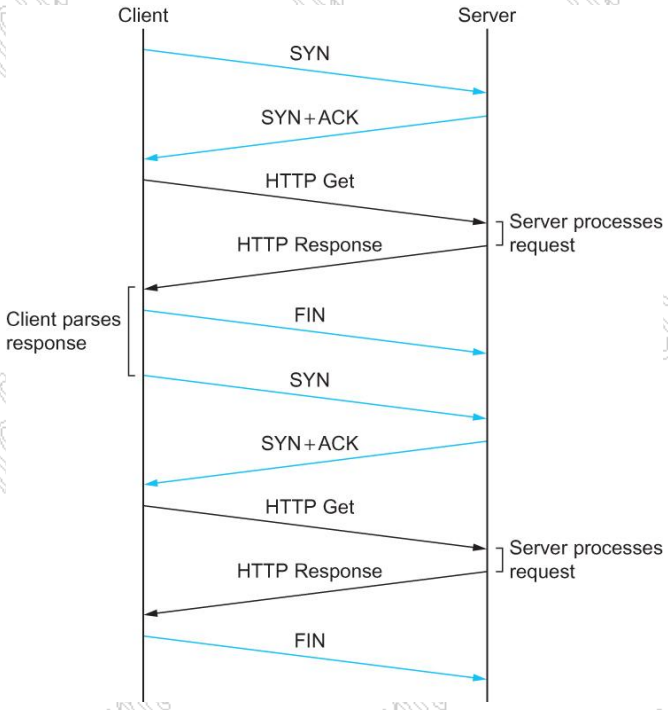
Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

- 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.

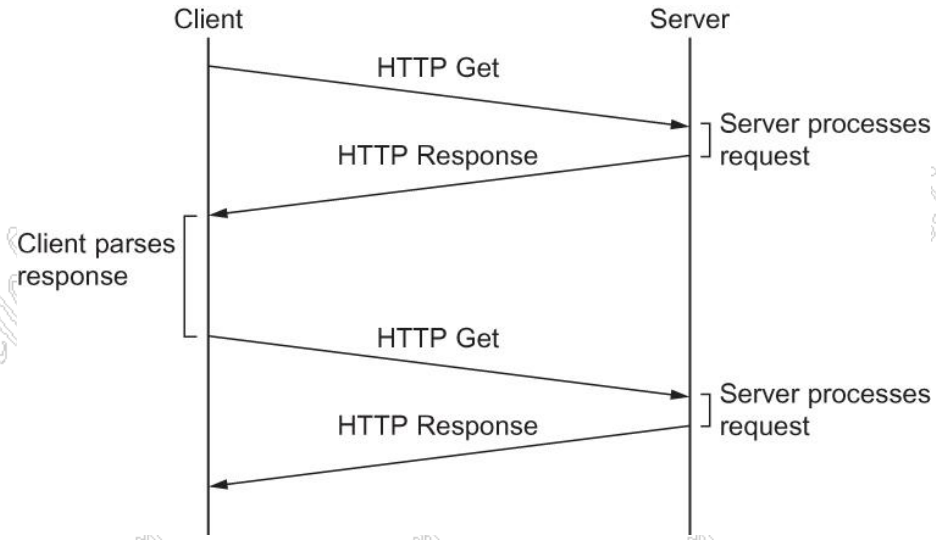
Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error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

- Uniform Resource Identifiers
  - The URLs that HTTP uses as addresses are one type of *Uniform Resource Identifier (URI).*
  - A URI is a character string that identifies a resource, where a resource can be anything that has identity, such as a document, an image, or a service.
  - The format of URIs allows various more-specialized kinds of resource identifiers to be incorporated into the URI space of identifiers.
  - The first part of a URI is a *scheme* that names a particular way of identifying a certain kind of resource, such as mailto for email addresses or file for file names.
  - The second part of a URI, separated from the first part by a colon, is the *scheme-specific part.*
- 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.

- To overcome this situation, HTTP version 1.1 introduced *persistent connections*— the client and server can exchange multiple request/response messages over the same TCP connection.
- Persistent connections have many 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.
    - This is because it's not necessary to go through the slow start phase for each page.



HTTP 1.0 behavior



HTTP 1.1 behavior with persistent connections

- Caching
  - One of the most active areas of research (and entrepreneurship) in the Internet today is how to effectively cache Web pages.
  - Caching has many benefits. From 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
  - Caching can be implemented in many different places. For example, a user's browser can cache recently accessed pages, and simply display the cached copy if the user visits the same page again.
  - As another example, a site can support a single site-wide cache.
  - This allows users to take advantage of pages previously downloaded by other users.
  - Closer to the middle of the Internet, ISPs can cache pages.
- In the second case, the users within the site most likely know what 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*

WEB SERVICES

- Much of the motivation for enabling direct application-to-application communication comes from the business world.
- Historically, interactions between enterprises—businesses or other organizations—have involved some manual steps such as filling out an order form or making a phone call to determine whether some product is in stock.
- Even within a single enterprise it is common to have manual steps between software systems that cannot interact directly because they were developed independently.
- Increasingly such manual interactions are being replaced with direct application-to application interaction.
- An ordering application at enterprise A would send a message to an order fulfillment application at enterprise B, which would respond immediately indicating whether the order can be filled.
- Perhaps, if the order cannot be filled by B, the application at A would immediately order from another supplier, or solicit bids from a collection of suppliers.
- Two architectures have been advocated as solutions to this problem.
- Both architectures are called *Web Services,* taking their name from the term for the individual applications that offer a remotely-accessible service to client applications to form network applications.
- The terms used as informal shorthand to distinguish the two Web Services architectures are *SOAP and REST (as in, "the SOAP vs. REST debate").*
- The SOAP architecture's approach to the problem is to make it feasible, at least in theory, to generate protocols that are customized to each network application.
- The key elements of the approach are a framework for protocol specification, software toolkits for automatically generating protocol implementations from the specifications, and modular partial specifications that can be reused across protocols.
- The REST architecture's approach to the problem is to regard individual Web Services as World Wide Web resources—identified by URIs and accessed via HTTP.

- Essentially, the REST architecture is just the Web architecture.
- The Web architecture's strengths include stability and a demonstrated scalability (in the network-size sense).

  Custom Application Protocols (WSDL, SOAP)
- The architecture informally referred to as SOAP is based on *Web Services Description Language (WSDL) and SOAP.4*
- Both of these standards are issued by the World Wide Web Consortium (W3C).
- This is the architecture that people usually mean when they use the term Web Services without any preceding qualifier.

TEXT BOOK:

1. 1. Larry L. Peterson, Bruce S. Davie, ―Computer Networks: A Systems Approach‖, Fifth Edition, Morgan Kaufmann Publishers, 2011.

REFERENCES:

1. James F. Kurose, Keith W. Ross, ―Computer Networking - A Top-Down Approach Featuring the Internet‖, Fifth Edition, Pearson Education, 2009.
2. Nader. F. Mir, ―Computer and Communication Networks‖, Pearson Prentice Hall Publishers, 2010.
3. Ying-Dar Lin, Ren-Hung Hwang, Fred Baker, ―Computer Networks: An Open Source Approach‖, Mc Graw Hill Publisher, 2011.
4. Behrouz A. Forouzan, ―Data communication and Networking‖, Fourth Edition, Tata McGraw – Hill, 2011.