UNIT – V

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

Introduction :

- The first is that 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, Mosaic, and Netscape—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.
- This section focuses on four application protocols:
    1. **SMTP:** Simple Mail Transfer Protocol is used to exchange electronic mail.
    2. **HTTP:** HyperText Transport Protocol is used to communicate between Web browsers and Web servers.
    3. **DNS :** Domain Name System Protocol is used to query name servers and send the responses.
    4. **SNMP:** Simple Network Management Protocol is used to query (and sometimes modify) the state of remote network nodes.
- All these protocols except DNS have a **companion protocol** that specifies the format of the data that can be exchanged.
    - For example, **SMTP** is a protocol for exchanging electronic mail messages, but **RFC 822 and MIME (Multipurpose Internet Mail Extensions)** define the format of email messages.
    - Similarly, **HTTP** is a protocol for fetching Web pages, but **HTML (HyperText Markup Language)** is a companion specification that defines the form of those pages.
    - Finally, **SNMP** is a protocol for querying a network node, but **MIB (management information base)** defines the variables that can be queried.

Notesengine.com

5. 1 Electronic Mail (SMTP, MIME, IMAP)

- Email is one of the oldest network applications.
- To learn how email works we need to distinguish between
    - the **user interface** (i.e., your mail reader) and the underlying **message transfer protocol** (in this case, SMTP),
    - the **transfer protocol** and a **companion protocol** (RFC 822 and MIME) that defines the format of the messages being exchanged.

Message Format

RFC 822

- RFC 822 defines **messages** to have two parts:
  a header and a body.

- Both 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,
  - **To:** header identifies the message recipient
  - **Subject:** header says something about the purpose of the message.
- Other headers are filled in by the underlying mail delivery system.
- For example,
  - **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.
- RFC 822 has been augmented by MIME to allow the message body to carry all sorts of data.

Notesengine.com

MIME (Multipurpose Internet Mail Extension)

- 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),
  - **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**

- As another example, **text/plain** refers to simple text,while **text/richtext** denotes a message that contains “marked up” text .

- 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.**

Base 64 Encoding Scheme

1. The idea is to map every three bytes of the original binary data into four ASCII characters.
2. This is done by grouping the binary data into 24-bit units.
3. Breaking each such unit into four 6-bit pieces.
4. Each 6-bit piece maps onto one of 64 valid ASCII characters.
5. The first 64 valid ASCII characters are the 52 upper- and lowercase letters, the 10 digits 0 through 9, and the special characters + and /.

- Putting this all together, a message that contains some plain text, a JPEG image,
- and a PostScript file would look something like this:

```
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
```

Notesengine.com

. . . readable encoding of a PostScript document

Message Transfer

- **SMTP (Simple Mail Transfer Protocol)**—the protocol used to transfer messages from one host to another.
- The users interact with a mail reader when they compose, file, search, and read their email.
- Most Web browsers now include a mail reader.
- There is a mail daemon (or process) running on each host. The mail daemon plays the e 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.
- The daemon puts incoming messages into the user's mailbox.
- Figure 5.1.1 illustrates a two-hop path from the sender to the receiver.

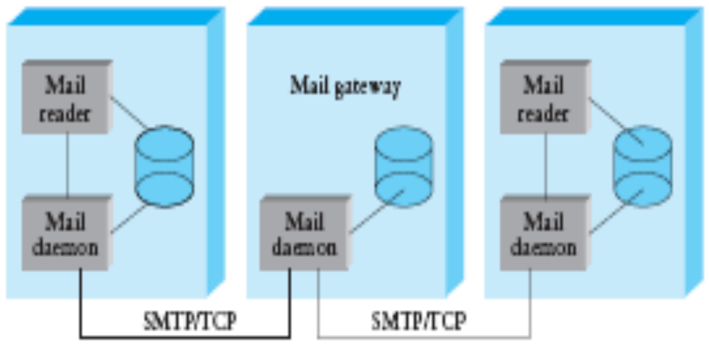


Figure 5.1.1 Sequence of mail gateways store and forward email messages.

- **For example,**
  - Mail delivered to Bob@cs.princeton.edu is first sent to a mail gateway in the CS Department at Princeton (that is, to the host named cs.princeton.edu), and then forwarded—involving a second SMTP/TCP connection—to the specific machine on which Bob happens to be reading his email today.
  - 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. (The list of Received: header lines in the message will help you trace the mail gateways that a given message traversed.)
  - 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.

- **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
```

Notesengine.com

Explanation of the SMTP Example

- 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) and 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).
- In this particular example, the client first identifies itself to the server with the HELO command. It gives its domain name as an argument.
- The server verifies that this name corresponds to the IP address being used by the TCP connection; you'll notice the server states this IP address back to the client.
- The client then asks the server if it is willing to accept mail for two different users; the server responds by saying "yes" to one and "no" to the other.

- Then the client sends the message, which is terminated by a line with a single period (“.”) on it.
- Finally, the client terminates the connection.

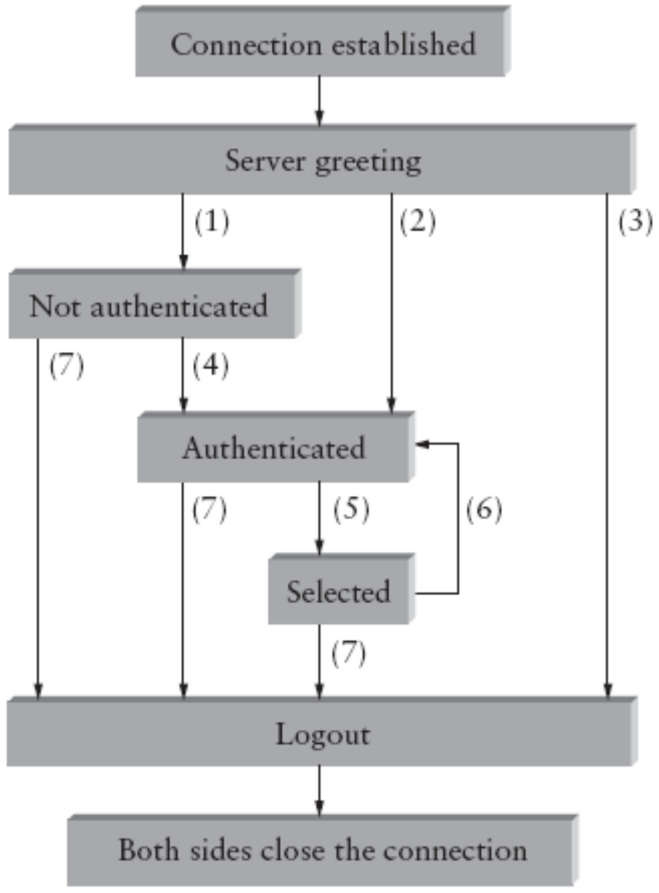
Mail Reader

- 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).**

IMAP

- It 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 he or she wants to access.
- This can be represented by the simple state transition diagram shown in Figure 5.1.2.

Notesengine.com



(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

Figure 5.1.2 IMAP state transition diagram.

5.2 World Wide Web (HTTP)

- The World Wide Web has been so successful and has made the Internet accessible to so many people that sometimes it seems to be synonymous with the Internet.
- Any web browser has a function that allows the users 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 ask yourWeb browser to open one of these embedded URLs (e.g., by pointing and clicking on it with a mouse), it will open a new connection and retrieve and display a new file.
- This is called **"following a link."**
- It thus becomes very easy to hop from one machine to another around the network, following links to all sorts of information.
- 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**.
- Each HTTP message has the general form

```
START_LINE <CRLF>
MESSAGE_HEADER <CRLF>
<CRLF>
MESSAGE_BODY <CRLF>
```

- Where <CRLF> stands for carriage-return-line-feed.
- The first line (START LINE) indicates whether this is a request message or a response message.
- The next set of lines specifies a collection of options and parameters that qualify the request or response.
- 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.
- 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:
  1. the operation to be performed
  2. the Web page the operation should be performed on
  3. the version of HTTP being used.

Notesengine.com

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

Table 5.2.1 HTTP Request Operations

- The full set of HTTP request operations is summarized in Table 5.2.1

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

Table 5.2.2 Five types of HTTP result codes.

Uniform Resource Identifiers

- A URI is a character string that identifies a resource ,where a resource can be anything that has an identity,such as a document,image or service.
- The first part of URI is a **scheme** that names a particular way of identifying a certain kind of resource.
- The second part is the **scheme-specific part.**

- The scheme is separated from the scheme-specific part by a colon.

Example

mailto:santa@northpole.org

file:///C:/foo.html

TCP Connections

- The original version of HTTP (1.0) established a separate TCP connection for each data
- item retrieved from the server.
- 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.**
  1. They 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.
  2. TCP's congestion window mechanism is able to operate more efficiently because a client can send multiple request messages down a single TCP connection.
- **Persistent connections have a disadvantage.**
  - The problem is that neither the client nor server necessarily knows how long to keep a particular TCP connection open. This is especially critical on the server, which might be asked to keep connections open on behalf of thousands of clients. The solution is that the server must time out and close a connection if it has received no requests on the connection for a period of time. Also, both the client and server must watch to see if the other side has elected to close the connection, and they must use that information as a signal that they should close their side of the connection as well. (Recall that both sides must close a TCP connection before it is fully terminated.)

Caching

- 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 can be implemented in many different places.

Notesengine.com

- 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.
- 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.

5.3 Network Management (SNMP)

- A **network is a complex system**, both in terms of the number of nodes that are involved and in terms of the suite of protocols that can be running on any one node.
- All the state that is maintained and manipulated on any one of these nodes—for example, address translation tables, routing tables, TCP connection state, and so on—then it becomes tedious to manage all of this information.
- This is the **problem of network management.**
- Since the nodes we want to keep track of are distributed, our only real option is to **use the network to manage the network.**
- 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** that 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**.
- SNMP also supports a third operation—**GET-NEXT.**

Notesengine.com

How does SNMP work?

- A system administrator interacts with a client program that displays information about the network.
- This client program usually has a graphical interface.
- You can think of this interface as playing the same role as a Web browser.
- 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.
- **How does the client indicate which piece of information it wants to retrieve?**
- **How does the server know which variable in memory to read to satisfy the request?**
    - The answer is that 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.
- Some examples of the groups are:

  1. **System:** general parameters of the system (node) as a whole, including where the node is located, how long it has been up, and the system's name.
  2. **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.
  3. **Address translation:** information about the Address Resolution Protocol (ARP), and in particular, the contents of its address translation table.
  4. **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 .
  5. **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.
  6. **UDP:** information about UDP traffic, including the total number of UDP datagrams that have been sent and received.
- There are also groups for ICMP, EGP, and SNMP itself. The 10th group is used by different media.

Notesengine.com

5.4 Name Service (DNS)

- We have been using addresses to identify hosts.
- Addresses are not exactly user friendly. It is for this reason that a **unique *name*** is also typically assigned to each host in a network.
- We now describe how a **naming service** can be developed to **map user-friendly names into router-friendly addresses.**
- Such a service is often the first application program implemented in a network since it frees other applications to identify hosts by name rather than by address.
- **Name services** are sometimes called ***middleware*** because they fill a gap between applications and the underlying network.
- **Difference between host names and host addresses**
- Host names are usually of variable length and mnemonic, thereby making them easier for humans to remember. Host addresses are fixed-length numeric addresses are easier for routers to process.
- Host names typically contain no information that helps the network locate the host. Addresses, in contrast, sometimes have routing information embedded in them.
- 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).
- 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).**

Birth of DNS

- The Internet did not always use DNS.
- Earlier, 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.
- 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.
- **The hosts.txt approach** could not be used as the number of hosts in the Internet started to grow.
- Hence DNS was used which 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.
- A user presents a host name to an applicationprogram (possibly embedded in a compound name such as an email address or URL),
- The naming system is used 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 5.4.1.

Notesengine.com

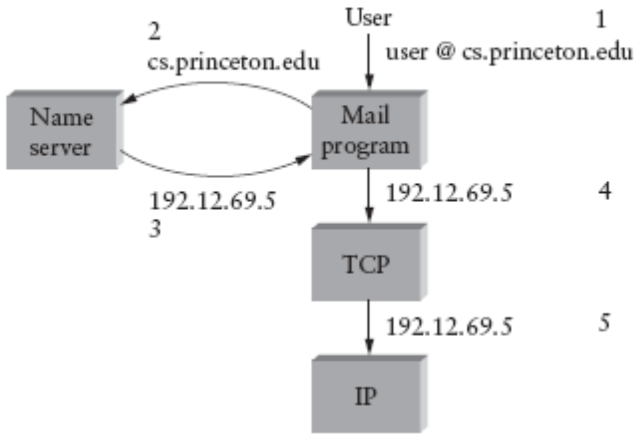


Figure 5.4.1 Names translated into addresses, where the numbers 1–5 show the sequence of steps in the process.

esengine.com

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.
- An **example domain name for a host** is **cicada.cs.princeton.edu.**

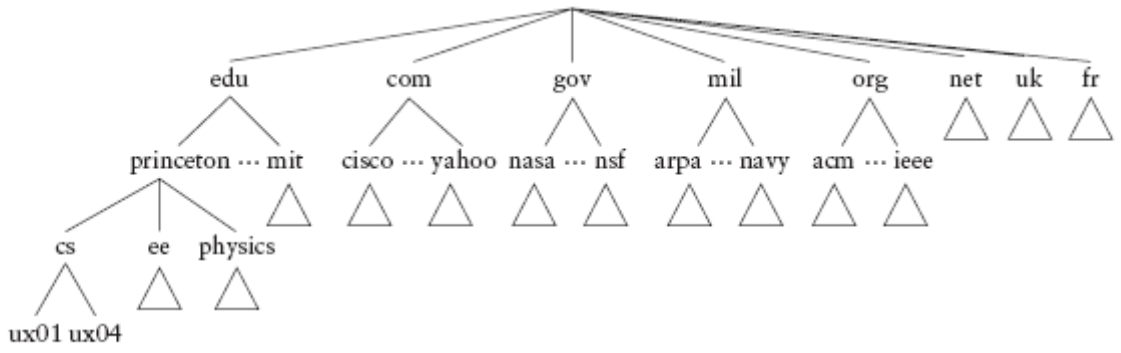


Figure 5.4.2 Example of a domain hierarchy.

- The DNS hierarchy can be visualized as a tree,
  - Each node corresponds to a domain
  - Leaves correspond to the hosts being named.
- Figure 5.4.2 gives an example of a domain hierarchy.

- There are domains for each country, plus the "big six" domains: edu, com, gov, mil, org, and net.
- These six domains are all based in the United States; the only domain names that don't explicitly specify a country are those in the United States.
- This is because the development of DNS was originally funded by ARPA, the major
- research arm of the U.S. Department of Defense.

Name Servers

- The hierarchy is partitioned into subtrees called *zones*.
- For example, Figure 5.4.3 shows how the hierarchy given in Figure 5.4.2 might be divided into zones.
- Each zone can be thought of as corresponding to some administrative authority that is responsible for that portion of the hierarchy.

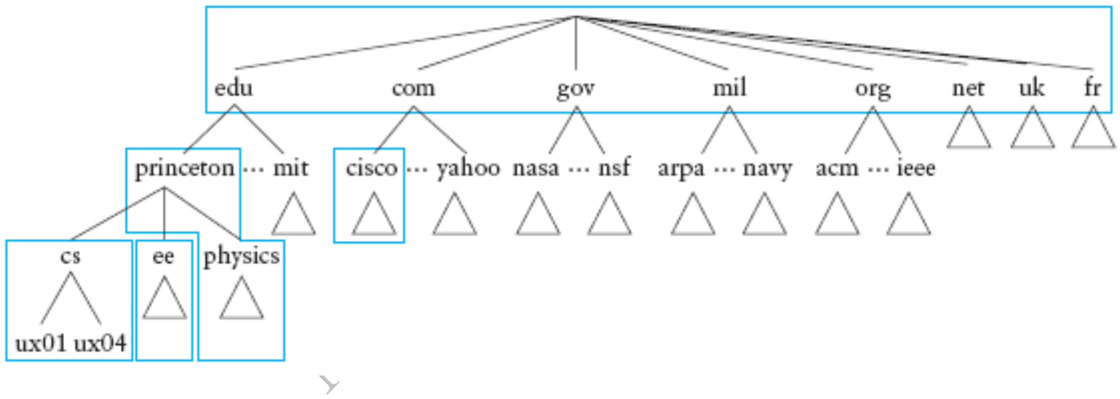


Figure 5.4.3 Domain hierarchy partitioned into zones.

- For example, the top level of the hierarchy forms a zone that is managed by the NIC. Below this is a zone that corresponds to Princeton University.
- Within this zone, some departments do not want the responsibility of managing the hierarchy (and so they remain in the university-level zone), while others, like the Department of Computer Science, manage their own department-level zone.
- The relevance of a zone is that it corresponds to the fundamental unit of implementation in DNS—the name server.
- 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.
- 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.

- It is more accurate to think of DNS as being represented by a hierarchy of name servers rather than by a hierarchy of domains, as illustrated in Figure 5.4.4.

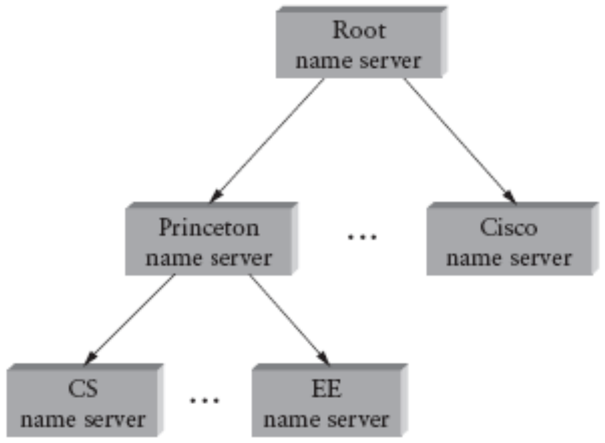


Figure 5.4.4 Hierarchy of name servers.

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.
- This results in the client/server interactions illustrated in Figure 5.4.5.

Notesengine.com

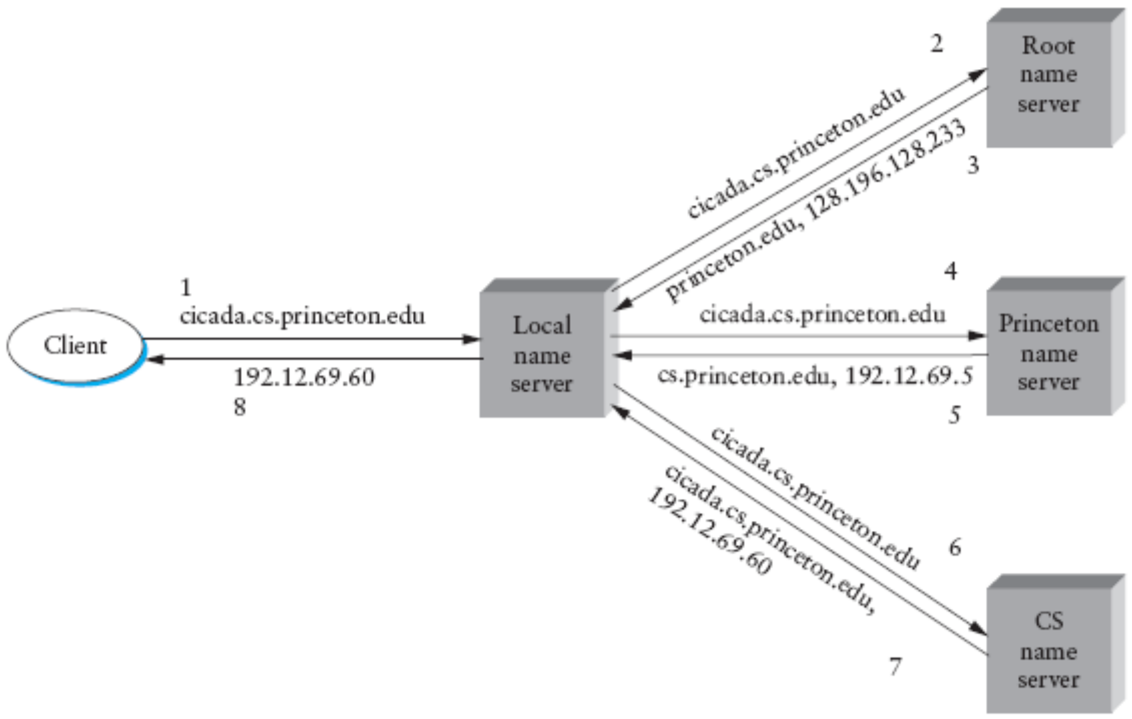


Figure 5.4.5 Name resolution in practice, where the numbers 1–8 show the sequence of steps in the process.

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.

5. 5 Security

Introduction

- Users sometimes want to encrypt the messages they send, with the goal of keeping anyone who is eavesdropping on the channel from being able to read the contents of the message.
- 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.
- 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.
- When a suitable combination of a key and an encryption algorithm is used, it is sufficiently difficult for an eavesdropper to break the ciphertext, and the sender and receiver can rest assured that their communication is secure.
- 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.

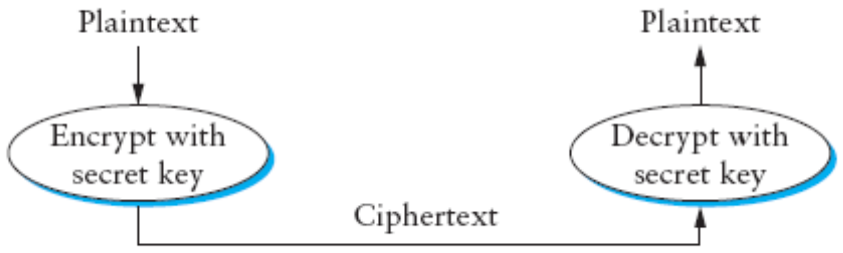


Figure 5.5.1 Secret Key Encryption

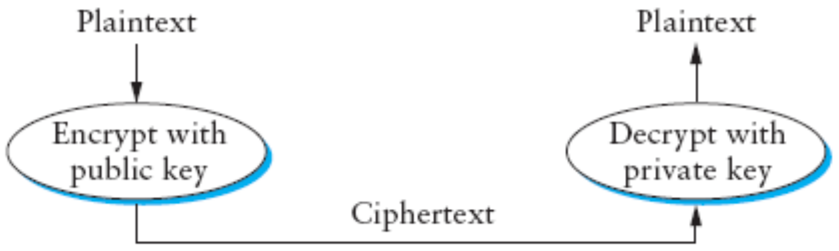


Figure 5.5.2 Public Key Encryption

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 keysigning 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.

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.

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.
- When the users login, both their passwords and all the data they send or receive potentially passes through countless untrusted 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. **SSH-AUTH: an authentication protocol**
    3. **SSH-CONN: a connection protocol**

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.

Notesengine.com

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 8.15, where we see a client on host A indirectly communicating with a server on host B by forwarding its traffic through an SSH connection.

Notesengine.com

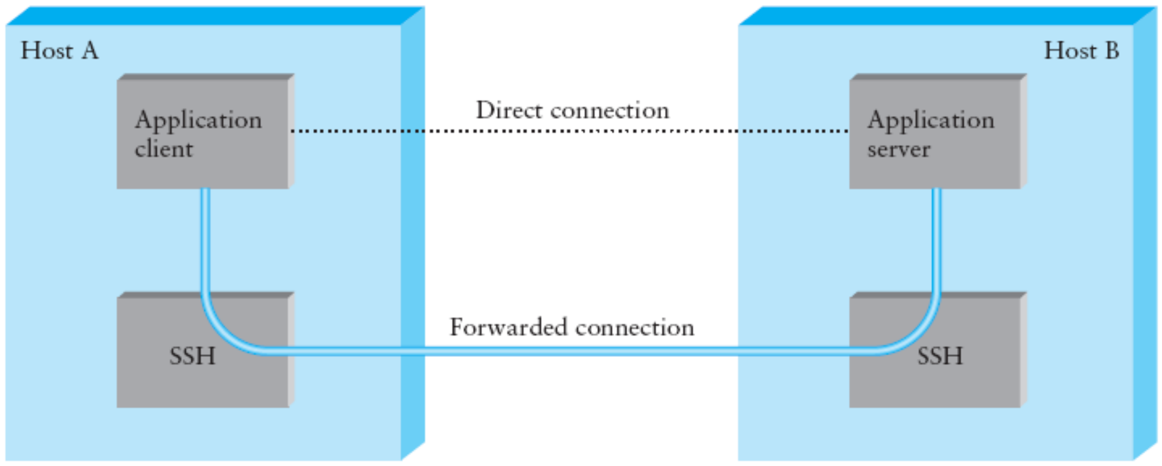


Figure 5.5.3 Using SSH port forwarding to secure other TCP-based applications.

- 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.

Notesengine.com