CHAPTER 23

Internet Applications—Internet Directory Service and the World Wide Web

Life in the modern world is coming to depend more and more upon technical means of communication. Without such technical aids the modern city-state could not exist, for it is only by means of them that trade and business can proceed; that goods and services can be distributed where needed; that railways can run on schedule; that law and order are maintained; that education is possible. Communication renders true social life practicable, for communication means organization.

—*On Human Communication*, Colin Cherry

KEY TOPICS

- The rapid growth in the use of the Web is due to the standardization of all the elements that support Web applications. A key element is HTTP, which is the protocol for the exchange of Web-based information between Web browsers and Web servers.
- Three types of intermediate devices can be used in an HTTP networks: proxies, gateways, and tunnels.
- HTTP uses a request/response style of communication.
- The Domain Name System (DNS) is a directory lookup service that provides a mapping between the name of a host on the Internet and its numerical address.
- DNS makes use of a distributed, hierarchical database to maintain a mapping from names to addresses and to provide related information about hosts on the Internet.

This chapter looks at two of the most widely used and more advanced Internet application areas. This chapter and the next should give the reader a feel for the range and diversity of applications supported by a communications architecture. The chapter begins with DNS, which is an essential name/address directory lookup service for the Internet. Then we look at HTTP, which is the support protocol on which the World Wide Web (WWW) operates.

23.1 INTERNET DIRECTORY SERVICE: DNS

The Domain Name System (DNS) is a directory lookup service that provides a mapping between the name of a host on the Internet and its numerical address. DNS is essential to the functioning of the Internet. It is defined in RFCs 1034 and 1035.

Four elements comprise the DNS:

- **Domain name space:** DNS uses a tree-structured name space to identify resources on the Internet.
- **DNS database:** Conceptually, each node and leaf in the name space tree structure names a set of information (e.g., IP address, type of resource) that is contained in a resource record (RR). The collection of all RRs is organized into a distributed database.
- **Name servers:** These are server programs that hold information about a portion of the domain name tree structure and the associated RRs.
- **Resolvers:** These are programs that extract information from name servers in response to client requests. A typical client request is for an IP address corresponding to a given domain name.

In the next two sections, we examine domain names and the DNS database, respectively. We then describe the operation of DNS, which includes a discussion of name servers and resolvers.

Domain Names

The 32-bit IP address provides a way of uniquely identifying devices attached to the Internet. This address is interpreted as having two components: a network number, which identifies a network on the Internet, and a host address, which identifies a unique host on that network. The practical use of IP addresses presents two problems:

1. Routers devise a path through the Internet on the basis of the network number. If each router needed to keep a master table that listed every network and the preferred path to that network, the management of the tables would be cumbersome and time consuming. It would be better to group the networks in such a way as to simplify the routing function.
2. The 32-bit address is usually written as four decimal numbers, corresponding to the four octets of the address. This number scheme is effective for computer processing but is not convenient for users, who can more easily remember names than numerical addresses.

These problems are addressed by the concept of **domain**. In general terms, a domain refers to a group of hosts that are under the administrative control of a single entity, such as a company or government agency. Domains are organized hierarchically, so that a given domain may consist of a number of subordinate domains. Names are assigned to domains and reflect this hierarchical organization.

Figure 23.1 shows a portion of the domain naming tree. At the very top level are a small number of domains that encompass the entire Internet. Table 23.1 lists currently defined top-level domains. Each subordinate level is named by prefixing a subordinate name to the name at the next highest level. For example,

- edu is the domain of college-level U.S. educational institutions.
- mit.edu is the domain for M.I.T. (the Massachusetts Institute of Technology).
- lcs.mit.edu is the domain for the Laboratory for Computer Science at M.I.T.

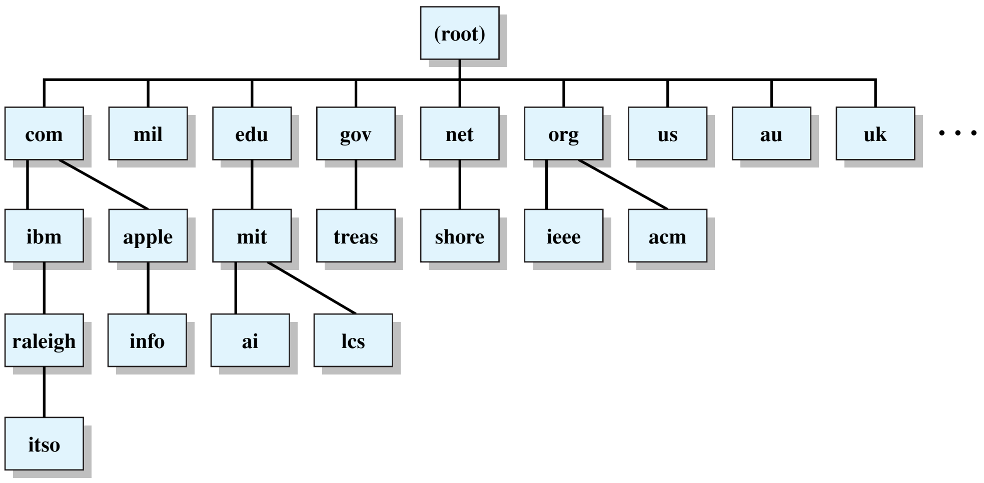


Figure 23.1 Portion of Internet Domain Tree

Table 23.1 Top-Level Internet Domains

Domain	Contents
com	Commercial organizations
edu	Educational institutions
gov	U.S. federal government agencies
mil	U.S. military
net	Network support centers, Internet service providers, and other network-related organizations
org	Nonprofit organizations
us	U.S. state and local government agencies, schools, libraries, and museums
country code	ISO standard 2-letter identifier for country-specific domains (e.g., au, ca, uk)
biz	Dedicated exclusively for private businesses
info	Unrestricted use
name	Individuals, for email addresses and personalized domain names.
museum	restricted to museums, museum organizations, and individual members of the museum profession
coop	Member-owned cooperative organizations, such as credit unions
aero	Aviation community
pro	Medical, legal, and accounting professions
arpa	Temporary ARPA domain (still used)
int	International organizations

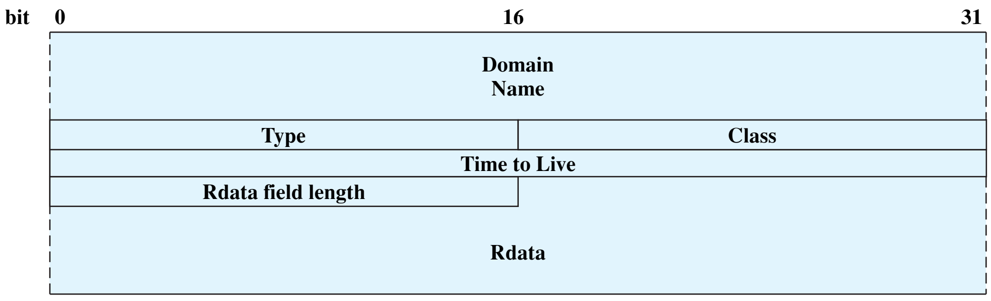


Figure 23.2 DNS Resource Record Format

As you move down the naming tree, you eventually get to leaf nodes that identify specific hosts on the Internet. These hosts are assigned Internet addresses. An Internet-wide organization is responsible for assigning domain names so that every domain name is unique. The actual assignment of addresses is delegated down the hierarchy. Thus, the mil domain is assigned a large group of addresses. The U.S. Department of Defense (DoD) then allocates portions of this address space to various DoD organizations for eventual assignment to hosts.

For example, the main host at MIT, with a domain name of mit.edu, has four IP addresses: 18.7.21.77, 18.7.21.69, 18.7.21.70, and 18.7.21.110. The subordinate domain lcs.mit.edu has the IP address 18.26.0.36.[1]

The DNS Database

DNS is based on a hierarchical database containing **resource records (RRs)** that include the name, IP address, and other information about hosts. The key features of the database are as follows:

- **Variable-depth hierarchy for names:** DNS allows essentially unlimited levels and uses the period (.) as the level delimiter in printed names, as described earlier.
- **Distributed database:** The database resides in DNS servers scattered throughout the Internet and private intranets.
- **Distribution controlled by the database:** The DNS database is divided into thousands of separately managed zones, which are managed by separate administrators. The database software controls distribution and update of records.

Using this database, DNS servers provide a name-to-address directory service for network applications that need to locate specific servers. For example, every time an e-mail message is sent or a Web page is accessed, there must be a DNS name lookup to determine the IP address of the e-mail server or Web server.

Figure 23.2 shows the structure of a RR. It consists of the following elements:

- **Domain Name:** Although the syntax of domain names in messages, described subsequently, is precisely defined, the form of the domain name in a RR is

[1]You should be able to demonstrate the name/address function by connecting your Web browser to your local ISP. The ISP should provide a ping or nslookup tool that allows you to enter a domain name and retrieve an IP address. Such a tool is typically available on user operating systems as well.

Table 23.2 Resource Record Types

Type	Description
A	A host address. This RR type maps the name of a system to its IP address. Some systems (e.g., routers) have multiple addresses, and there is a separate RR for each.
CNAME	Canonical name. Specifies an alias name for a host and maps this to the canonical (true) name.
HINFO	Host information. Designates the processor and operating system used by the host.
MINFO	Mailbox or mail list information. Maps a mailbox or mail list name to a host name.
MX	Mail exchange. Identifies the systems that relay mail into the organization.
NS	Authoritative name server for this domain.
PTR	Domain name pointer. Points to another part of the domain name space.
SOA	Start of a zone of authority (which part of naming hierarchy is implemented). Includes parameters related to this zone.
SRV	For a given service provides name of server or servers in domain that provide that service.
TXT	Arbitrary text. Provides a way to add text comments to the database.
WKS	Well-known services. May list the application services available at this host.

described in general terms. In essence, the domain name in a RR must correspond to the human readable form, which consists of a series of labels of alphanumeric characters or hyphens, with each pair of labels separated by a period.

- **Type:** Identifies the type of resource in this RR. The various types are listed in Table 23.2.[2]
- **Class:** Identifies the protocol family. The only commonly used value is IN, for the Internet.
- **Time to Live:** Typically, when a RR is retrieved from a name server, the retriever will cache the RR so that it need not query the name server repeatedly. This field specifies the time interval that the resource record may be cached before the source of the information should again be consulted. A zero value is interpreted to mean that the RR can only be used for the transaction in progress and should not be cached.
- **Rdata Field Length:** Length of the Rdata field in octets.
- **Rdata:** A variable-length string of octets that describes the resource. The format of this information varies according to the type of the RR. For example, for the A type, the Rdata is a 32-bit IP address, and for the CNAME type, the Rdata is a domain name.

[2]The SRV RR type is defined in RFC 2782.

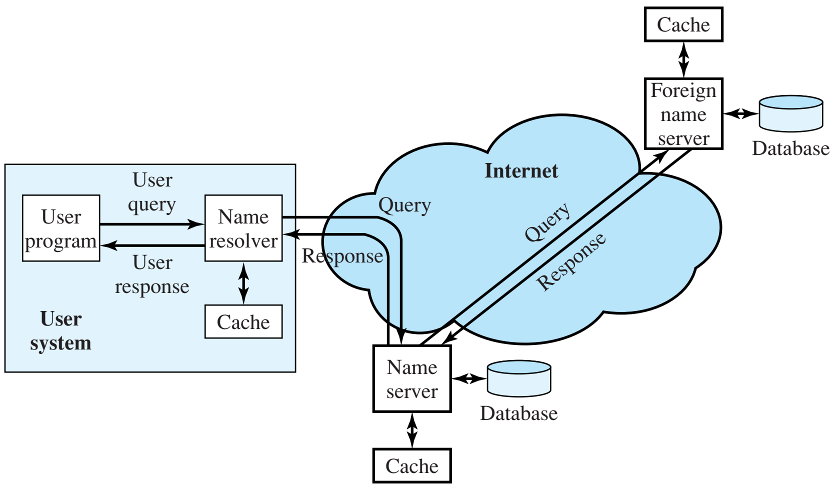


Figure 23.3 DNS Name Resolution

DNS Operation

DNS operation typically includes the following steps (Figure 23.3):

1. A user program requests an IP address for a domain name.
2. A resolver module in the local host or local ISP formulates a query for a local name server in the same domain as the resolver.
3. The local name server checks to see if the name is in its local database or cache, and, if so, returns the IP address to the requestor. Otherwise, the name server queries other available name servers, starting down from the root of the DNS tree or as high up the tree as possible.
4. When a response is received at the local name server, it stores the name/address mapping in its local cache and may maintain this entry for the amount of time specified in the time to live field of the retrieved RR.
5. The user program is given the IP address or an error message.

The results of these behind-the-scenes activities are seen by the user in a way illustrated in Figure 23.4. Here, a user issues a Telnet connection request to locis.loc.gov. This is resolved by DNS to the IP address of 140.147.254.3.

The distributed DNS database that supports the DNS functionality must be updated frequently because of the rapid and continued growth of the Internet. Accordingly, dynamic updating functions for DNS have been defined. In essence, DNS name servers automatically send out updates to other relevant name servers as conditions warrant.

The Server Hierarchy The DNS database is distributed hierarchically, residing in DNS name servers scattered throughout the Internet. Name servers can be operated by any organization that owns a domain; that is, any organization that has responsibility for a subtree of the hierarchical domain name space. Each name server is configured with a subset of the domain name space, known as a **zone**, which

```
telnet locis.loc.gov
Trying 140.147.254.3...
Connected to locis.loc.gov.
Escape character is '^]'.
          L O C I S:  LIBRARY OF CONGRESS INFORMATION SYSTEM

          To make a choice: type a number, then press ENTER

  1   Copyright Information    -- files available and up-to-date

  2   Braille and Audio        -- files frozen mid-August 1999

  3   Federal Legislation      -- files frozen December 1998

*    *    *    *    *    *    *    *    *    *    *    *    *    *    *

                The LC Catalog Files are available at:
                     http://lcweb.loc.gov/catalog/

*    *    *    *    *    *    *    *    *    *    *    *    *    *    *

  8   Searching Hours and Basic Search Commands
  9   Library of Congress General Information
 10   Library of Congress Fast Facts

 12   Comments and Logoff
      Choice:
 9
                 LIBRARY OF CONGRESS GENERAL INFORMATION

 LC is a research library serving Congress, the federal government, the
 library community world-wide, the US creative community, and any researchers
 beyond high school level or age.  On-site researchers request materials by
 filling out request slips in LC's reading rooms; requesters must present a
 photo i.d.  Staff are available for assistance in all public reading rooms.

 -----------------------------------------------------------------------------
 The following phone numbers offer information about hours and other services:

 General Research Info:     202-707-6500     Reading Room Hours:    202-707-6400
 Exhibits/Tours/Gift Shop: 202-707-8000     Location/Parking:      202-707-4700
 Copyright Information:     202-707-3000     Cataloging Products:   202-707-6100
 Copyright Forms:           202-707-9100          "       "   fax:  202-707-1334

 -----------------------------------------------------------------------------
 For information on interlibrary loan, see:  http://lcweb.loc.gov/rr/loan/


 12  Return to LOCIS MENU screen

 Choice:
```

Figure 23.4 A Telnet Session

is a collection of one or more (or all) subdomains within a domain, along with the associated RRs. This set of data is called authoritative, because this name server is responsible for maintaining an accurate set or RRs for this portion of the domain name hierarchy. The hierarchical structure can extend to any depth. Thus, a portion of the name space assigned to an authoritative name server can be delegated to a subordinate name server in a way that corresponds to the structure of the domain

Table 23.3 Internet Root Servers

Server	Operator	Cities	IP Addr
A	VeriSign Global Registry Services	Herndon VA, US	198.41.0.4
B	Information Sciences Institute	Marina Del Rey CA, US	128.9.0.107
C	Cogent Communications	Herndon VA, US	192.33.4.12
D	University of Maryland	College Park MD, US	128.8.10.90
E	NASA Ames Research Center	Mountain View CA, US	192.203.230.10
F	Internet Software Consortium San Francisco CA, US	Palo Alto CA, US; IPv6: 2001:500::1035	IPv4: 192.5.5.241
G	U.S. DOD Network Information Center	Vienna VA, US	192.112.36.4
H	U.S. Army Research Lab	Aberdeen MD, US	128.63.2.53
I	Autonomica	Stockholm, SE	192.36.148.17
J	VeriSign Global Registry Services	Herndon VA, US	192.58.128.30
K	Reseaux IP Europeens—Network Coordination Centre	London, UK	193.0.14.129
L	Internet Corporation for Assigned Names and Numbers	Los Angeles CA, US	198.32.64.12
M	WIDE Project	Tokyo, JP	202.12.27.33

name tree. For example, a name server corresponds to the domain **ibm.com**. A portion of that domain is defined by the name **watson.ibm.com**, which corresponds to the node **watson.ibm.com** and all of the branches and leaf nodes underneath the node **watson.ibm.com**.

At the top of the server hierarchy are 13 **root name servers** that share responsibility for the top level zones (Table 23.3). This replication is to prevent the root server from becoming a bottleneck. Even so, each individual root server is quite busy. For example, the Internet Software Consortium reports that its server (F) answers almost 300 million DNS requests daily (**www.isc.org/services/public/F-root-server.html**).

Consider a query by a program on a user host for watson.ibm.com. This query is sent to the local server and the following steps occur:

1. If the local server already has the IP address for **watson.ibm.com** in its local cache, it returns the IP address.
2. If the name is not in the local name server's cache, it sends the query to a root server. The root server in turn forwards the request to a server with an NS record for **ibm.com**. If this server has the information for **watson.ibm.com**, it returns the IP address.
3. If there is a delegated name server just for **watson.ibm.com**, then the **ibm.com** name server forwards the request to the watson.ibm.com name server, which returns the IP address.

Typically, single queries are carried over UDP. Queries for a group of names are carried over TCP.

Name Resolution As Figure 23.3 indicates, each query begins at a name resolver located in the user host system (e.g., gethostbyname in UNIX). Each resolver is configured to know the name and address of a local DNS name server. If the resolver does not have the requested name in its cache, it sends a DNS query to the local DNS server, which either returns an address immediately or does so after querying one or more other servers. Again, resolvers use UDP for single queries and TCP for group queries.

There are two methods by which queries are forwarded and results returned. Suppose a name server (A) forwards a DNS request to another name server (B). If B has the name/address in its local cache or local database, it can return the IP address to A. If not, then B can do either of the following:

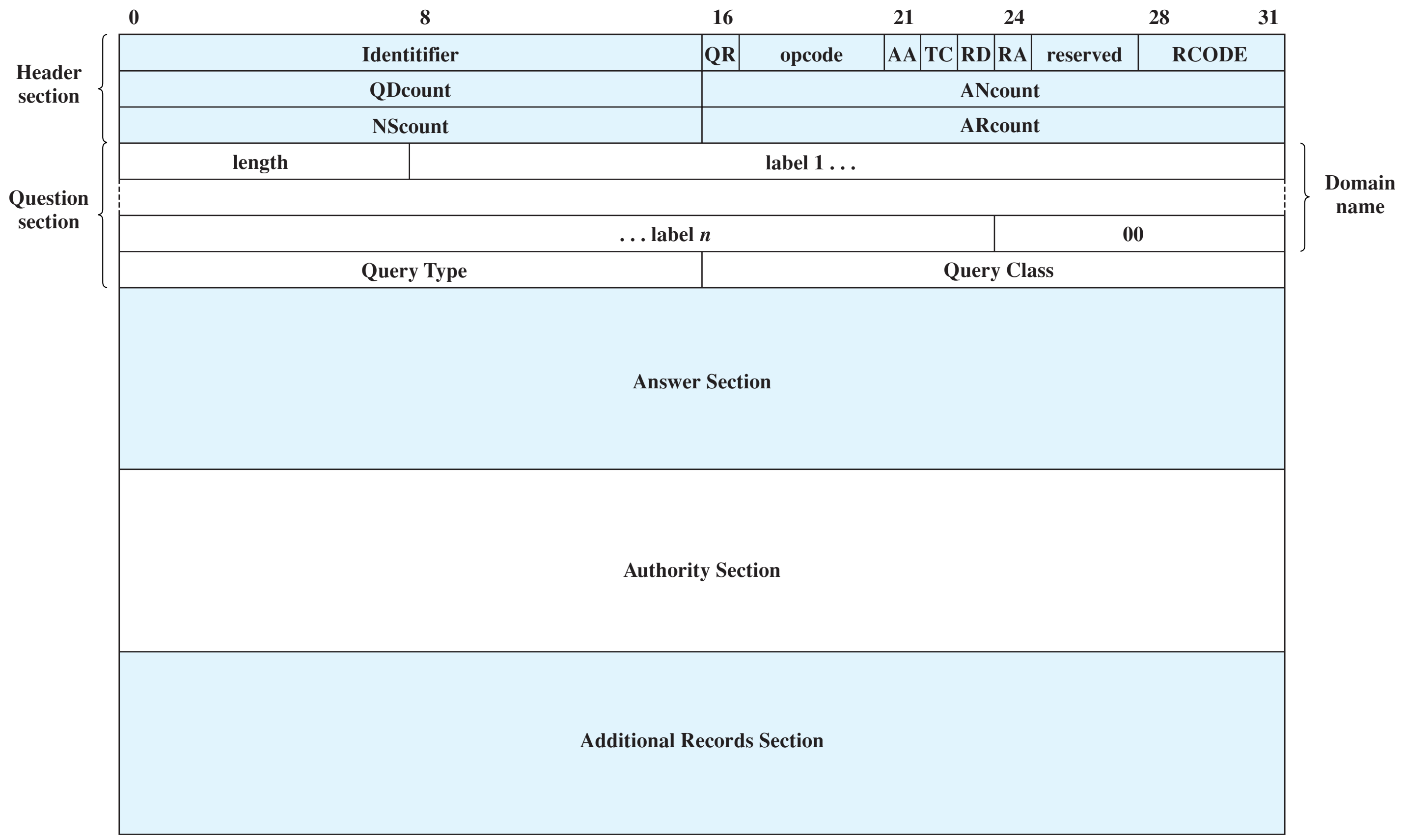
1. Query another name server for the desired result and then send the result back to A. This is known as a **recursive** technique.
2. Return to A the address of the next server (C) to whom the request should be sent. A then sends out a new DNS request to C. This is known as the **iterative** technique.

In exchanges between name servers, either the iterative or recursive technique may be used. For requests sent by a name resolver, the recursive technique is used.

DNS Messages DNS messages use a single format, shown in Figure 23.5. There are five possible sections to a DNS message: header, question, answer, authority, and additional records.

The **header section** is always present and consists of the following fields:

- **Identifier:** Assigned by the program that generates any kind of query. The same identifier is used in any response, enabling the sender to match queries and responses.
- **Query Response:** Indicates whether this message is a query or response.
- **Opcode:** Indicates whether this is a standard query, an inverse query (address to name), or a server status request. This value is set by the originator and copied into the response.
- **Authoritative Answer:** Valid in a response, and indicates whether the responding name server is an authority for the domain name in question.
- **Truncated:** Indicates whether the response message was truncated due to length greater then permitted on the transmission channel. If so, the requestor will use a TCP connection to resend the query.
- **Recursion Desired:** If set, directs the server to pursue the query recursively.
- **Recursion Available:** Set or cleared in a response to denote whether recursive query support is available in the name server.
- **Response Code:** Possible values are: no error, format error (server unable to interpret query), server failure, name error (domain name does not exist), not implemented (this kind of query not supported), and refused (for policy reasons).



QR = query/response bit	RCODE = response code
AA = authoritative answer	QDcount = number of entries in question section
TC = truncated	ANcount = number of resource records in answer section
RD = recursion desired	NScount = number of name server resource records in authority section
RA = recursion available	ARcount = number of resource records in additional records section

Figure 23.5 DNS Message Format

- **QDcount:** Number of entries in question section (zero or more).
- **ANcount:** Number of RRs in answer section (zero or more).
- **NScount:** Number of RRs in authority section (zero or more).
- **ARcount:** Number of RRs in additional records section (zero or more).

The **question section** contains the queries for the name server. If present, it typically contains only one entry. Each entry contains the following:

- **Domain Name:** A domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root.
- **Query Type:** Indicates type of query. The values for this field include all values valid for the Type field in the RR format (Figure 23.2), together with some more general codes that match more than one type of RR.
- **Query Class:** Specifies the class of query, typically Internet.

The **answer section** contains RRs that answer the question; the **authority section** contains RRs that point toward an authoritative name server; the **additional records section** contains RRs that relate to the query but are not strictly answers for the question.

23.2 WEB ACCESS—HTTP

The Hypertext Transfer Protocol (HTTP) is the foundation protocol of the World Wide Web (WWW) and can be used in any client/server application involving hypertext. The name is somewhat misleading in that HTTP is not a protocol for transferring hypertext; rather it is a protocol for transmitting information with the efficiency necessary for making hypertext jumps. The data transferred by the protocol can be plaintext, hypertext, audio, images, or any Internet-accessible information.

We begin with an overview of HTTP concepts and operation and then look at some of the details, basing our discussion on the most recent version to be put on the Internet standards track, HTTP 1.1 (RFC 2616). A number of important terms defined in the HTTP specification are summarized in Table 23.4; these will be introduced as the discussion proceeds.

HTTP Overview

HTTP is a transaction-oriented client/server protocol. The most typical use of HTTP is between a Web browser and a Web server. To provide reliability, HTTP makes use of TCP. Nevertheless, HTTP is a "stateless" protocol: Each transaction is treated independently. Accordingly, a typical implementation will create a new TCP connection between client and server for each transaction and then terminate the connection as soon as the transaction completes, although the specification does not dictate this one-to-one relationship between transaction and connection lifetimes.

The stateless nature of HTTP is well suited to its typical application. A normal session of a user with a Web browser involves retrieving a sequence of Web pages

Table 23.4 Key Terms Related to HTTP

Cache	A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server while it is acting as a tunnel.
Client	An application program that establishes connections for the purpose of sending requests.
Connection	A transport layer virtual circuit established between two application programs for the purposes of communication.
Entity	A particular representation or rendition of a data resource, or reply from a service resource, that may be enclosed within a request or response message. An entity consists of entity headers and an entity body.
Gateway	A server that acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the original server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. Gateways are often used as server-side portals through network firewalls and as protocol translators for access to resources stored on non-HTTP systems.
Message	The basic unit of HTTP communication, consisting of a structured sequence of octets transmitted via the connection.
Origin Server	The server on which a given resource resides or is to be created.
Proxy	An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them, with possible translation, on to other servers. A proxy must interpret and, if necessary, rewrite a request message before forwarding it. Proxies are often used as client-side portals through network firewalls and as helper applications for handling requests via protocols not implemented by the user agent.
Resource	A network data object or service which can be identified by a URI.
Server	An application program that accepts connections in order to service requests by sending back responses.
Tunnel	An intermediary program that is acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication, though the tunnel may have been initiated by an HTTP request. A tunnel ceases to exist when both ends of the relayed connections are closed. Tunnels are used when a portal is necessary and the intermediary cannot, or should not, interpret the relayed communication.
User Agent	The client that initiates a request. These are often browsers, editors, spiders, or other end-user tools.

and documents. The sequence is, ideally, performed rapidly, and the locations of the various pages and documents may be a number of widely distributed servers.

Another important feature of HTTP is that it is flexible in the formats that it can handle. When a client issues a request to a server, it may include a prioritized list of formats that it can handle, and the server replies with the appropriate format. For example, a Lynx browser cannot handle images, so a Web server need not transmit any images on Web pages. This arrangement prevents the transmission of unnecessary information and provides the basis for extending the set of formats with new standardized and proprietary specifications.

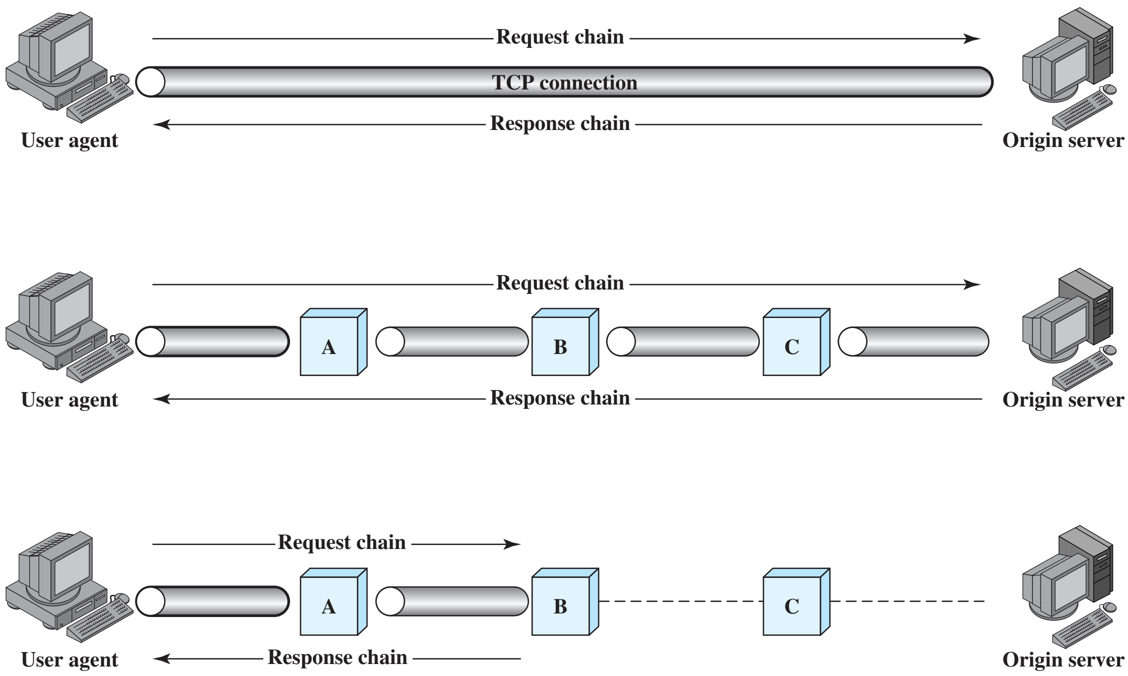


Figure 23.6 Examples of HTTP Operation

Figure 23.6 illustrates three examples of HTTP operation. The simplest case is one in which a user agent establishes a direct connection with an origin server. The *user agent* is the client that initiates the request, such as a Web browser being run on behalf of an end user. The *origin server* is the server on which a resource of interest resides; an example is a Web server at which a desired Web home page resides. For this case, the client opens a TCP connection that is end-to-end between the client and the server. The client then issues an HTTP request. The request consists of a specific command, referred to as a method, an address [referred to as a Uniform Resource Locator (URL)],[3] and a MIME-like message containing request parameters, information about the client, and perhaps some additional content information.

When the server receives the request, it attempts to perform the requested action and then returns an HTTP response. The response includes status information, a success/error code, and a MIME-like message containing information about the server, information about the response itself, and possible body content. The TCP connection is then closed.

The middle part of Figure 23.6 shows a case in which there is not an end-to-end TCP connection between the user agent and the origin server. Instead, there are one or more intermediate systems with TCP connections between logically adjacent systems. Each intermediate system acts as a relay, so that a request initiated by the

[3]Appendix M contains a discussion of URLs.

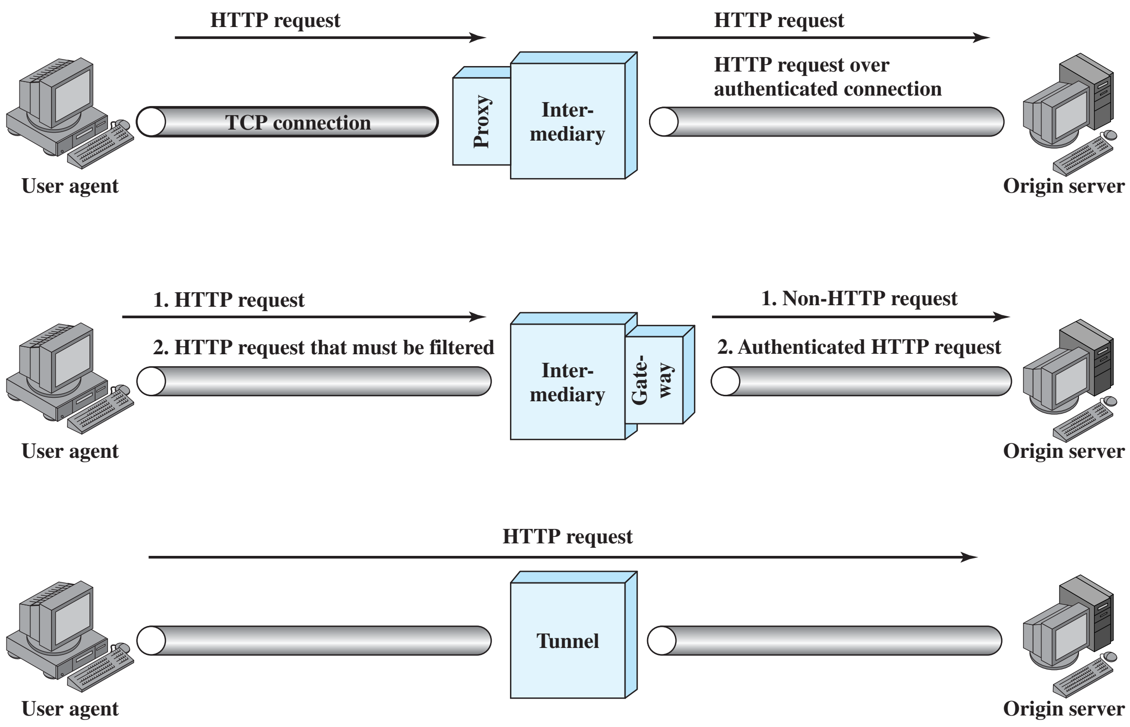


Figure 23.7 Intermediate HTTP Systems

client is relayed through the intermediate systems to the server, and the response from the server is relayed back to the client.

Three forms of intermediate system are defined in the HTTP specification: proxy, gateway, and tunnel, all of which are illustrated in Figure 23.7.

Proxy A proxy acts on behalf of other clients and presents requests from other clients to a server. The proxy acts as a server in interacting with a client and as a client in interacting with a server. There are two scenarios that call for the use of a proxy:

- **Security intermediary:** The client and server may be separated by a security intermediary such as a firewall, with the proxy on the client side of the firewall. Typically, the client is part of a network secured by a firewall and the server is external to the secured network. In this case, the server must authenticate itself to the firewall to set up a connection with the proxy. The proxy accepts responses after they have passed through the firewall.
- **Different versions of HTTP:** If the client and server are running different versions of HTTP, then the proxy can implement both versions and perform the required mapping.

In summary, a proxy is a forwarding agent, receiving a request for a URL object, modifying the request, and forwarding the request toward the server identified in the URL.

Gateway A gateway is a server that appears to the client as if it were an origin server. It acts on behalf of other servers that may not be able to communicate directly with a client. There are two scenarios in which gateways can be used.

- **Security intermediary:** The client and server may be separated by a security intermediary such as a firewall, with the gateway on the server side of the firewall. Typically, the server is connected to a network protected by a firewall, with the client external to the network. In this case the client must authenticate itself to the gateway, which can then pass the request on to the server.
- **Non-HTTP server:** Web browsers have built into them the capability to contact servers for protocols other than HTTP, such as FTP and Gopher servers. This capability can also be provided by a gateway. The client makes an HTTP request to a gateway server. The gateway server then contacts the relevant FTP or Gopher server to obtain the desired result. This result is then converted into a form suitable for HTTP and transmitted back to the client.

Tunnel Unlike the proxy and the gateway, the tunnel performs no operations on HTTP requests and responses. Instead, a tunnel is simply a relay point between two TCP connections, and the HTTP messages are passed unchanged as if there were a single HTTP connection between user agent and origin server. Tunnels are used when there must be an intermediary system between client and server but it is not necessary for that system to understand the contents of messages. An example is a firewall in which a client or server external to a protected network can establish an authenticated connection and then maintain that connection for purposes of HTTP transactions.

Cache Returning to Figure 23.6, the lowest portion of the figure shows an example of a cache. A cache is a facility that may store previous requests and responses for handling new requests. If a new request arrives that is the same as a stored request, then the cache can supply the stored response rather than accessing the resource indicated in the URL. The cache can operate on a client or server or on an intermediate system other than a tunnel. In the figure, intermediary B has cached a request/response transaction, so that a corresponding new request from the client need not travel the entire chain to the origin server, but is handled by B.

Not all transactions can be cached, and a client or server can dictate that a certain transaction may be cached only for a given time limit.

Messages

The best way to describe the functionality of HTTP is to describe the individual elements of the HTTP message. HTTP consists of two types of messages: requests from clients to servers, and responses from servers to clients. The general structure of such messages is shown in Figure 23.8. More formally, using enhanced BNF (Backus-Naur Form) notation[4] (Table 23.5), we have

[4]A description of BNF is contained in Appendix N.

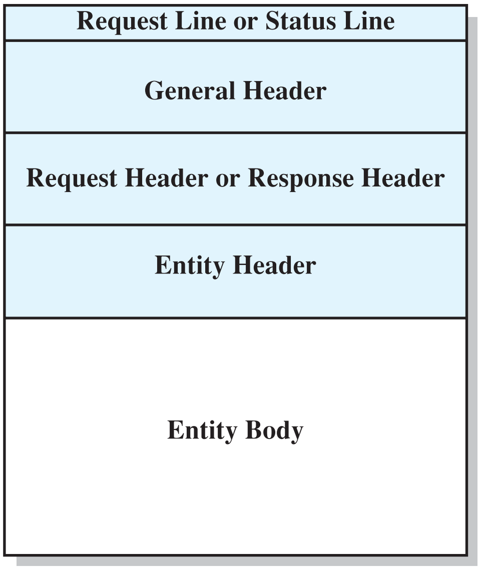

Request Line or Status Line
General Header
Request Header or Response Header
Entity Header
Entity Body

Figure 23.8 General Structure of HTTP Messages

Table 23.5 Augmented BNF Notation Used in URL and HTTP Specifications

- Words in lowercase represent variables or names of rules.
- A rule has the form

  name = definition

- DIGIT is any decimal digit; CRLF is carriage return, line feed; SP is one or more spaces.
- Quotation marks enclose literal text.
- Angle brackets, "<" ">", may be used within a definition to enclose a rule name when their presence will facilitate clarity.
- Elements separated by bar ("|") are alternatives.
- Ordinary parentheses are used simply for grouping.
- The character "*" preceding an element indicates repetition. The full form is:

  <I>*<J>element

  indicating at least I and at most J occurrences of element. *element allows any number, including 0; 1*element requires at least one element; and 1*2element allows 1 or 2 elements; <N>element means exactly N elements.
- Square brackets, "[" "]", enclose optional elements.
- The construct "#" is used to define, with the following form:

  <I>#<J>element

  indicating at least I and at most J elements, each separated by a comma and optional linear white space.
- A semicolon at the right of a rule starts a comment that continues to the end of the line.

```
HTTP-Message = Simple-Request | Simple-Response | Full-Request |
Full-Response

Full-Request =     Request-Line
    *( General-Header | Request-Header | Entity-Header )
    CRLF
    [ Entity-Body ]

Full-Response = Status-Line
    *( General-Header | Response-Header | Entity-Header )
    CRLF
    [ Entity-Body ]

Simple-Request  = "GET" SP Request-URL CRLF

Simple-Response = [ Entity-Body ]
```

The Simple-Request and Simple-Response messages were defined in HTTP/0.9. The request is a simple GET command with the requested URL; the response is simply a block containing the information identified in the URL. In HTTP/1.1, the use of these simple forms is discouraged because it prevents the client from using content negotiation and the server from identifying the media type of the returned entity.

With full requests and responses, the following fields are used:

- **Request-Line:** Identifies the message type and the requested resource
- **Status-Line:** Provides status information about this response
- **General-Header:** Contains fields that are applicable to both request and response messages but that do not apply to the entity being transferred
- **Request-Header:** Contains information about the request and the client
- **Response-Header:** Contains information about the response
- **Entity-Header:** Contains information about the resource identified by the request and information about the entity body
- **Entity-Body:** The body of the message

All of the HTTP headers consist of a sequence of fields, following the same generic format as RFC 822 (described in Chapter 22). Each field begins on a new line and consists of the field name followed by a colon and the field value.

Although the basic transaction mechanism is simple, there is a large number of fields and parameters defined in HTTP. In the remainder of this section, we look at the general header fields. Following sections describe request headers, response headers, and entities.

General Header Fields General header fields can be used in both request and response messages. These fields are applicable in both types of messages and contain information that does not directly apply to the entity being transferred. The fields are as follows:

- **Cache-Control:** Specifies directives that must be obeyed by any caching mechanisms along the request/response chain. The purpose is to prevent a cache from adversely interfering with this particular request or response.
- **Connection:** Contains a list of keywords and header field names that only apply to this TCP connection between the sender and the nearest nontunnel recipient.
- **Date:** Date and time at which the message originated.
- **Forwarded:** Used by gateways and proxies to indicate intermediate steps along a request or response chain. Each gateway or proxy that handles a message may attach a Forwarded field that gives its URL.
- **Keep-Alive:** May be present if the keep-alive keyword is present in an incoming Connection field, to provide information to the requester of the persistent connection. This field may indicate a maximum time that the sender will keep the connection open waiting for the next request or the maximum number of additional requests that will be allowed on the current persistent connection.
- **MIME-Version:** Indicates that the message complies with the indicated version of MIME.
- **Pragma:** Contains implementation-specific directives that may apply to any recipient along the request/response chain.
- **Upgrade:** Used in a request to specify what additional protocols the client supports and would like to use; used in a response to indicate which protocol will be used.

Request Messages

A full request message consists of a status line followed by one or more general, request, and entity headers, followed by an optional entity body.

Request Methods

A full request message always begins with a Request-Line, which has the following format:

Request-Line = Method SP Request-URL SP HTTP-Version CRLF

The Method parameter indicates the actual request command, called a method in HTTP. Request-URL is the URL of the requested resource, and HTTP-Version is the version number of HTTP used by the sender.

The following request methods are defined in HTTP/1.1:

- **OPTIONS:** A request for information about the options available for the request/response chain identified by this URL.
- **GET:** A request to retrieve the information identified in the URL and return it in a entity body. A GET is conditional if the If-Modified-Since header field is included and is partial if a Range header field is included.
- **HEAD:** This request is identical to a GET, except that the server's response must not include an entity body; all of the header fields in the response are the same as if the entity body were present. This enables a client to get information about a resource without transferring the entity body.

- **POST:** A request to accept the attached entity as a new subordinate to the identified URL. The posted entity is subordinate to that URL in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.
- **PUT:** A request to accept the attached entity and store it under the supplied URL. This may be a new resource with a new URL or a replacement of the contents of an existing resource with an existing URL.
- **PATCH:** Similar to a PUT, except that the entity contains a list of differences from the content of the original resource identified in the URL.
- **COPY:** Requests that a copy of the resource identified by the URL in the Request-Line be copied to the location(s) given in the URL-Header field in the Entity-Header of this message.
- **MOVE:** Requests that the resource identified by the URL in the Request-Line be moved to the location(s) given in the URL-Header field in the Entity-Header of this message. Equivalent to a COPY followed by a DELETE.
- **DELETE:** Requests that the origin server delete the resource identified by the URL in the Request-Line.
- **LINK:** Establishes one or more link relationships from the resource identified in the Request-Line. The links are defined in the Link field in the Entity-Header.
- **UNLINK:** Removes one or more link relationships from the resource identified in the Request-Line. The links are defined in the Link field in the Entity-Header.
- **TRACE:** Requests that the server return whatever is received as the entity body of the response. This can be used for testing and diagnostic purposes.
- **WRAPPED:** Allows a client to send one or more encapsulated requests. The requests may be encrypted or otherwise processed. The server must unwrap the requests and process accordingly.
- **Extension-method:** Allows additional methods to be defined without changing the protocol, but these methods cannot be assumed to be recognizable by the recipient.

Request Header Fields Request header fields function as request modifiers, providing additional information and parameters related to the request. The following fields are defined in HTTP/1.1:

- **Accept:** A list of media types and ranges that are acceptable as a response to this request.
- **Accept-Charset:** A list of character sets acceptable for the response.
- **Accept-Encoding:** List of acceptable content encodings for the entity body. Content encodings are primarily used to allow a document to be compressed or encrypted. Typically, the resource is stored in this encoding and only decoded before actual use.
- **Accept-Language:** Restricts the set of natural languages that are preferred for the response.

- **Authorization:** Contains a field value, referred to as *credentials*, used by the client to authenticate itself to the server.
- **From:** The Internet e-mail address for the human user who controls the requesting user agent.
- **Host:** Specifies the Internet host of the resource being requested.
- **If-Modified-Since:** Used with the GET method. This header includes a date/time parameter; the resource is to be transferred only if it has been modified since the date/time specified. This feature allows for efficient cache update. A caching mechanism can periodically issue GET messages to an origin server, and will receive only a small response message unless an update is needed.
- **Proxy-Authorization:** Allows the client to identify itself to a proxy that requires authentication.
- **Range:** For future study. The intent is that, in a GET message, a client can request only a portion of the identified resource.
- **Referrer:** The URL of the resource from which the Request-URL was obtained. This enables a server to generate lists of back-links.
- **Unless:** Similar in function to the If-Modified-Since field, with two differences: (1) It is not restricted to the GET method, and (2) comparison is based on any Entity-Header field value rather than a date/time value.
- **User-Agent:** Contains information about the user agent originating this request. This is used for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations.

Response Messages

A full response message consists of a status line followed by one or more general, response, and entity headers, followed by an optional entity body.

Status Codes A full response message always begins with a Status-Line, which has the following format:

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

The HTTP-Version value is the version number of HTTP used by the sender. The Status-Code is a three-digit integer that indicates the response to a received request, and the Reason-Phrase provides a short textual explanation of the status code.

HTTP/1.1 includes a rather large number of status codes, organized into the following categories:

- **Informational:** The request has been received and processing continues. No entity body accompanies this response.
- **Successful:** The request was successfully received, understood, and accepted. The information returned in the response message depends on the request method, as follows:

—GET: The contents of the entity-body corresponds to the requested resource.

—HEAD: No entity body is returned.

—POST: The entity describes or contains the result of the action.

—TRACE: The entity contains the request message.

—Other methods: The entity describes the result of the action.

- **Redirection:** Further action is required to complete the request.
- **Client Error:** The request contains a syntax error or the request cannot be fulfilled.
- **Server Error:** The server failed to fulfill an apparently valid request.

Response Header Fields Response header fields provide additional information related to the response that cannot be placed in the Status-Line. The following fields are defined in HTTP/1.1:

- **Location:** Defines the exact location of the resource identified by the Request-URL.
- **Proxy-Authenticate:** Included with a response that has a status code of Proxy Authentication Required. This field contains a "challenge" that indicates the authentication scheme and parameters required.
- **Public:** Lists the nonstandard methods supported by this server.
- **Retry-After:** Included with a response that has a status code of Service Unavailable, and indicates how long the service is expected to be unavailable.
- **Server:** Identifies the software product used by the origin server to handle the request.
- **WWW-Authenticate:** Included with a response that has a status code of Unauthorized. This field contains a "challenge" that indicates the authentication scheme and parameters required.

Entities

An entity consists of an entity header and an entity body in a request or response message. An entity may represent a data resource, or it may constitute other information supplied with a request or response.

Entity Header Fields Entity header fields provide optional information about the entity body or, if no body is present, about the resource identified by the request. The following fields are defined in HTTP/1.1:

- **Allow:** Lists methods supported by the resource identified in the Request-URL. This field must be included with a response that has a status code of Method Not Allowed and may be included in other responses.
- **Content-Encoding:** Indicates what content encodings have been applied to the resource. The only encoding currently defined is zip compression.
- **Content-Language:** Identifies the natural language(s) of the intended audience of the enclosed entity.

- **Content-Length:** The size of the entity body in octets.
- **Content-MD5:** For future study. MD5 refers to the MD5 hash code function, described in Chapter 21.
- **Content-Range:** For future study. The intent is that this will indicate a portion of the identified resource that is included in this response.
- **Content-Type:** Indicates the media type of the entity body.
- **Content-Version:** A version tag associated with an evolving entity.
- **Derived-From:** Indicates the version tag of the resource from which this entity was derived before modifications were made by the sender. This field and the Content-Version field can be used to manage multiple updates by a group of users.
- **Expires:** Date/time after which the entity should be considered stale.
- **Last-Modified:** Date/time that the sender believes the resource was last modified.
- **Link:** Defines links to other resources.
- **Title:** A textual title for the entity.
- **Transfer-Encoding:** Indicates what type of transformation has been applied to the message body to transfer it safely between the sender and the recipient. The only encoding defined in the standard is *chunked.* The chunked option defines a procedure for breaking an entity body into labeled chunks that are transmitted separately.
- **URL-Header:** Informs the recipient of other URLs by which the resource can be identified.
- **Extension-Header:** Allows additional fields to be defined without changing the protocol, but these fields cannot be assumed to be recognizable by the recipient.

Entity Body An entity body consists of an arbitrary sequence of octets. HTTP is designed to be able to transfer any type of content, including text, binary data, audio, images, and video. When an entity body is present in a message, the interpretation of the octets in the body is determined by the entity header fields Content-Encoding, Content-Type, and Transfer-Encoding. These define a three-layer, ordered encoding model:

$$\text{entity-body} := \text{Transfer-Encoding}$$
$$(\text{Content-Encoding}(\text{Content-Type}(\text{data})))$$

The data are the content of a resource identified by a URL. The Content-Type field determines the way in which the data are interpreted. A Content-Encoding may be applied to the data and stored at the URL instead of the data. Finally, on transfer, a Transfer-Encoding may be applied to form the entity body of the message.

23.3 RECOMMENDED READING AND WEB SITES

[MOGU02] discusses the design strengths and weaknesses of HTTP. [GOUR02] provides comprehensive coverage of HTTP. Another good treatment is [KRIS01]. [MOCK88] is an overview of DNS.

GOUR02 Gourley, D., et al. *HTTP: The Definitive Guide.* Sebastopol, CA: O'Reilly, 2002.

KRIS01 Krishnamurthy, B., and Rexford, J. *Web Protocols and Practice: HTTP/1.1. Networking Protocols, Caching, and Traffic Measurement.* Upper Saddle River, NJ: Prentice Hall, 2001.

MOCK88 Mockapetris, P., and Dunlap, K. "Development of the Domain Name System." *ACM Computer Communications Review*, August 1988.

MOGU02 Mogul, J. "Clarifying the Fundamentals of HTTP." *Proceedings of the Eleventh International Conference on World Wide Web*, 2002.

Recommended Web Sites:

- **WWW Consortium:** Contains up-to-date information on HTTP and related topics.
- **DNS Extensions Working Group:** Chartered by IETF to develop standards related to DNS. The Web site includes all relevant RFCs and Internet drafts.

23.4 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

Backus-Naur Form (BNF)	HTTP tunnel	Resolver
domain	Hypertext Transfer Protocol (HTTP)	resource record (RR)
domain name	iterative technique	root name server
Domain Name Service (DNS)	name server	Uniform Resource Locator (URL)
HTTP gateway	origin server	zone
HTTP method	recursive technique	
HTTP proxy		

Review Questions

23.1 What is DNS?

23.2 What is the difference between a name server and a resolver in DNS?

23.3 What is a DNS resource record?

23.4 Give a brief description of DNS operation.

23.5 What is the difference between a domain and a zone?

23.6 Explain the difference between the recursive technique and the iterative technique in DNS.

23.7 What is meant by saying that HTTP is a stateless protocol?

23.8 Explain the differences among HTTP proxy, gateway, and tunnel.

23.9 What is the function of the cache in HTTP?

Problems

Note: For some of the problems in this chapter, you will need to consult the relevant RFCs.

23.1 Classify a DNS resolver and a DNS name server as either client, server, or both.

23.2 A DNS resolver typically issues a query using UDP but may also use TCP. Is there a problem using TCP for this purpose? If so, what do you suggest is the solution?
Hint: Consider the TCP and UDP headers.

23.3 What's the main difference between a primary and a secondary name server?

23.4 Name servers can be accessed on UDP port 53 as well as on TCP port 53. When is each protocol used, and why?

23.5 We query an authoritative name server for the 'example.com' zone, in order to get the IP address of **www.example.com**, the Web site of a large company. We get eight A records in response to our query. We repeat this query several times, and note that we continue getting the same eight A records, but in a different order each time. Suggest a reason why.

23.6 The dig tool provides easy interactive access to the DNS. The dig tool is available for UNIX and Windows operating systems. It can also be used from the Web. Here are three sites that, at the time of this writing, provided free access to dig:

http://www.gont.com.ar/tools/dig

http://www.webmaster-toolkit.com/dig.shtml

http://www.webhostselect.com/whs/dig-tool.jsp

Use the dig tool to get the list of root servers.

23.7 Discuss the advantages of using several stub resolvers along with a caching-only name server, instead of several full resolvers.

23.8 Choose a root server, and use the dig tool to send it a query for the IP address of **www.example.com**, with the RD (Recursion Desired) bit set. Does it support recursive lookups? Why or why not?

23.9 Type dig **www.example.com** A in order to get the IP address of **www.example.com**. What's the TTL of the A record returned in the response? Wait a while, and repeat the query. Why has the TTL changed?

23.10 With the widespread use of x-DSL and cable-modem technologies, many home users now host Web sites on their own desktop computers. As their IP addresses are dynamically assigned by their Internet Service Providers (ISPs), users must update their DNS records every time their IP addresses change (it's usually done by some computer software on the user machine that automatically contacts the name server to update the corresponding data whenever the assigned IP address changes). This service is usually called Dynamic DNS. However, in order for these updates to work as expected, there's one field of each resource record that must be set to a quite different value from the typical ones. Which one, and why?

23.11 Secondary name servers periodically query the primary to check whether the zone data has been updated. Regardless of how many resource records the zone data contains, the secondary name servers need to query the primary only one resource record to detect any changes on the zone data. Which resource record will they query? How will they use the requested information to detect changes?

23.12 A user on the host 170.210.17.145 is 'using a Web browser to visit **www.example.com**. In order to resolve the 'www.example.com' domain to an IP address, a query is sent to an authoritative name server for the 'example.com' domain. In response, the name server returns a list of four IP addresses, in the following order {192.168.0.1, 128.0.0.1, 200.47.57.1, 170.210.10.130}. Even though it is the last IP address in the list returned by the name server, the Web browser creates a connection to 170.210.17.130. Why?

23.13 Before the deployment of the Domain Name System, a simple text file (HOSTS.TXT) centrally maintained at the SRI Network Information Center was used to enable mapping between host names and addresses. Each host connected to the Internet had to have an updated local copy of it to be able to use host names instead of having to cope directly with their IP addresses. Discuss the main advantages of the DNS over the old centralized HOSTS.TXT system.

23.14 Prior to persistent connections, one separate TCP connection was used to fetch each URL. Analyze the advantages of persistent connections over the old HTTP paradigm of one connection per data transfer.