**CAPITULO 7**

Having finished all the preliminaries, we now come to the layer where all the applications are found. The layers below the application layer are there to provide transport services, but they do not do real work for users. However, even in the application layer there is a need for support protocols, to allow the applications to function.

**7.1 DNS – The Domain Name System**

The essence of DNS is the invention of a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping host names to IP addresses but can also be used for other purposes.

Very briefly, the way DNS is used is as follows. To map a name onto an IP address, an application program calls a library procedure called the **resolver**, passing it the name as a parameter. The resolver sends a query containing the name to a local DNS server, which looks up the name and returns a response containing the IP address to the resolver, which then returns it to the caller. The query and response messages are sent as UDP packets. Armed with the IP address, the program can then establish a TCP connection with the host or send it UDP packets.

**7.1.1 The DNS name space**

Managing a large and constantly changing set of names is a nontrivial problem. Getting a second-level domain, such as name-of-company.com, is easy. The top-level domains are run by registrars appointed by ICANN. Getting a name merely requires going to a corresponding registrar (for com in this case) to check if the desired name is available and not somebody else’s trademark. If there are no problems, the requester pays the registrar a small annual fee and gets the name.

**7.1.3 Name Servers**

In theory at least, a single name server could contain the entire DNS database and respond to all queries about it. In practice, this server would be so overloaded as to be useless. Furthermore, if it ever went down, the entire Internet would be crippled.

To avoid the problems associated with having only a single source of information, the DNS name space is divided into nonoverlapping **zones**.

Diagram

Description automatically generated

Where the zone boundaries are placed within a zone is up to that zone’s administrator. This decision is made in large part based on how many name servers are desired, and where. Each zone is also associated with one or more name servers. These are hosts that hold the database for the zone. Normally, a zone will have one primary name server, which gets its information from a file on its disk, and one or more secondary name servers, which get their information from the primary name server.

The process of looking up a name and finding an address is called **name resolution**. When a resolver has a query about a domain name, it passes the query to a local name server. If the domain being sought falls under the jurisdiction of the name server, it returns the authoritative resource records. An **authoritative record** is one that comes from the authority that manages the record and is thus always correct.

Diagram

Description automatically generatedWhat happens when a remote domain wants to find the IP address of another domain? In this case, and if there is no cached information about the domain available locally, the name server begins a remote query.

The query is sent to the local name server. The query contains the domain name sought, the type (A), and the class(IN). The next step is to start at the top of the name hierarchy by asking one of the **root name servers**. These name servers have information about each top-level domain. To contact a root server, each name server must have information about one or more root name servers. This information is normally present in a system configuration file that is loaded into the DNS cache when the DNS server is started. It is simply a list of NS records for the root and the corresponding A records.

Each root server could logically be a single computer. However, since the entire Internet depends on the root servers, they are powerful and heavily replicated computers. Most of the servers are present in multiple geographical locations and reached using anycast routing, in which a packet is delivered to the nearest instance of a destination address.

The root name server (and each subsequent name server) does not recursively continue the query for the local name server. It just returns a partial answer and moves on to the next query. The local name server is responsible for continuing the resolution by issuing further queries. This mechanism is called an **iterative query**.

A recursive query may always seem preferable, but many name servers (especially the root) will not handle them. They are too busy. Iterative queries put the burden on the originator. Those hosts do not have to be configured to run a full name server, just to reach the local one.

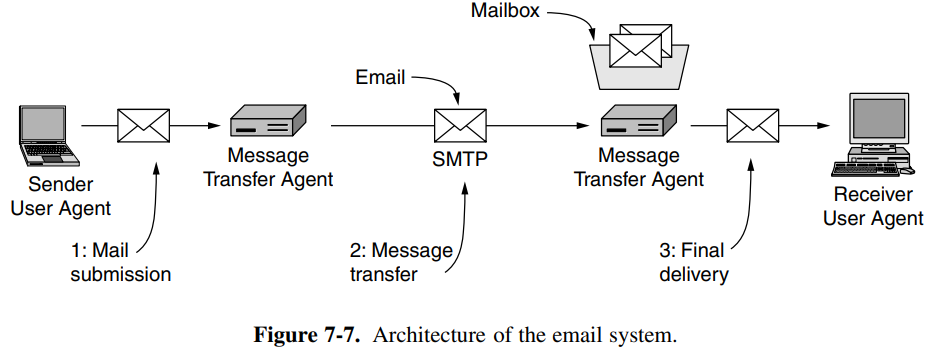
All the answers, including all the partial answers returned, are cached. This way, if a host in domain 1 queries for another host in domain 2 the answer will already be known. Even better, if a host queries for a different host in the same domain, the query can be sent directly to the authoritative name server. Using cached answers greatly reduces the steps in a query and improves performance.

The third issue is the transport protocol that is used for the queries and responses. It is UDP. DNS messages are sent in UDP packets with a simple format for queries, answers, and name servers that can be used to continue the resolution. If no response arrives within a short time, the DNS client repeats the query, trying another server for the domain after a small number of retries.

This process is designed to handle the case of the server being down as well as the query or response packet getting lost. A 16-bit identifier is included in each query and copied to the response so that a name server can match answers to the corresponding query, even if multiple queries are outstanding at the same time.

**7.2 Electronic Mail**

**7.2.1 Architecture and Services**

The architecture of the email system consists of two kinds of subsystems: the **user agents**, which allow people to read and send email, and the **message transfer agents**, which move the messages from the source to the destination. We will also refer to message transfer agents informally as **mail servers**.

The user agent is a program that provides a graphical interface, or sometimes a text- and command-based interface that lets users interact with the email system. The act of sending new messages into the mail system for delivery is called **mail submission**.

The message transfer agents are typically system processes. They run in the background on mail server machines and are intended to be always available. Their job is to automatically move email through the system from the originator to the recipient with **SMTP (Simple Mail Transfer Protocol)**. This is the message transfer step.

SMTP sends mail over connections and reports back the delivery status and any errors. Message transfer agents also implement **mailing lists**, in which an identical copy of a message is delivered to everyone on a list of email addresses. Mail is sent between message transfer agents in a standard format.

A key idea in the message format is the distinction between the envelope and its contents. The envelope encapsulates the message. It contains all the information needed for transporting the message, such as the destination address, priority, and security level, all of which are distinct from the message itself. The message transport agents use the envelope for routing, just as the post office does.

The message inside the envelope consists of two separate parts: the **header** and the **body**. The header contains control information for the user agents. The body is entirely for the human recipient.

Diagram

Description automatically generated with low confidence

**7.2.4 Message Transfer**

**------------- SMTP (Simple Mail Transfer Protocol) -------------**

Within the Internet, email is delivered by having the sending computer establish a TCP connection to port 25 of the receiving computer. Listening to this port is a mail server that speaks SMTP (Simple Mail Transfer Protocol). This server accepts incoming connections, subject to some security checks, and accepts messages for delivery. If a message cannot be delivered, an error report containing the first part of the undeliverable message is returned to the sender. SMTP is a simple ASCII protocol.

After establishing the TCP connection to port 25, the sending machine, operating as the client, waits for the receiving machine, operating as the server, to talk first. The server starts by sending a line of text giving its identity and telling whether it is prepared to receive mail (phase 1 - greeting). If it is not, the client releases the connection and tries again later.

If the server is willing to accept email, the client announces whom the email is coming from and whom it is going to. If such a recipient exists at the destination, the server gives the client the go-ahead to send the message. Then the client sends the message and the server acknowledges it (phase 2 – transfer of messages). No checksums are needed because TCP provides a reliable byte stream. When all the email has been exchanged in both directions, the connection is released (phase 3 - closure).

**7.2.5 Final Delivery**

All that remains is to transfer a copy of the message to Bob’s user agent for display. This is step 3 in the architecture of Fig. 7-7.

**------------- IMAP—The Internet Message Access Protocol -------------**

One of the main protocols that is used for final delivery is **IMAP (Internet Message Access Protocol)**. To use IMAP, the mail server runs an IMAP server that listens to port 143. The user agent runs an IMAP client. First, the client will start a secure transport if one is to be used, and then log in or otherwise authenticate itself to the server. One user thus has many IMAP mailboxes, each of which is typically presented to the user as a folder.

IMAP has the ability to address mail not by message number, but by using attributes. Searches can be performed on the server to find the messages that satisfy certain criteria so that only those messages are fetched by the client. IMAP is an improvement over an earlier final delivery protocol, **POP3 (Post Office Protocol, version 3)**.

POP3 is a simpler protocol but supports fewer features and is less secure in typical usage. Mail is usually downloaded to the user agent computer, instead of remaining on the mail server. This makes life easier on the server, but harder on the user. It is not easy to read mail on multiple computers, plus if the user agent computer breaks, all email may be lost permanently.

Proprietary protocols can also be used because the protocol runs between a mail server and user agent that can be supplied by the same company. Microsoft Exchange is a mail system with a proprietary protocol.

**7.3 The World Wide Web**

**7.3.1 Cookies**

Navigating the Web as we have described it so far involves a series of independent page fetches. There is no concept of a login session. The browser sends a request to a server and gets back a file. Then the server forgets that it has ever seen that particular client.

This model is not suited for returning different pages to different users depending on what they have already done with the server. This problem is solved with an oft-criticized mechanism called **cookies**.

When a client requests a Web page, the server can supply additional information in the form of a cookie along with the requested page. The cookie is a rather small, named string (of at most 4 KB) that the server can associate with a browser. In principle, a cookie could contain a virus, but since cookies are treated as data, there is no official way for the virus to actually run and do damage.

A cookie may contain up to five fields:

* The *Domain* tells where the cookie came from.
* The *Path* is a path in the server’s directory structure that identifies which parts of the server’s file tree may use the cookie. It is often /, which means the whole tree.
* The *Content* field takes the form name = value. Both name and value can be anything the server wants. This field is where the cookie’s content is stored.
* The *Expires* field specifies when the cookie expires. If this field is absent, the browser discards the cookie when it exits.
* The *Secure* field can be set to indicate that the browser may only return the cookie to a server using a secure transport, namely SSL/TLS.

**7.3.4 HTTP—The HyperText Transfer Protocol**

The protocol that is used to transport all this information between Web servers and clients. It is **HTTP (HyperText Transfer Protocol)**. HTTP is a simple request-response protocol that normally runs over TCP. The request and response headers are given in ASCII and the contents are given in a MIME-like format. This simple model was partly responsible for the early success of the Web because it made development and deployment straightforward.

HTTP is an application layer protocol because it runs on top of TCP and is closely associated with the Web. However, in another sense HTTP is becoming more like a transport protocol that provides a way for processes to communicate content across the boundaries of different networks.

**------------- Connections -------------**

HTTP 1.1 supports **persistent connections**. With them, it is possible to establish a TCP connection, send a request and get a response, and then send additional requests and get additional responses. This strategy is also called **connection reuse**. By amortizing the TCP setup, startup, and release costs over multiple requests, the relative overhead due to TCP is reduced per request. It is also possible to pipeline requests, that is, send request 2 before the response to request 1 has arrived.

Part (a) shows three requests, one after the other and each in a separate connection. The URLs of the images are determined as the main page is fetched, so they are fetched after the main page.

Diagram

Description automatically generatedIn part (c) there is one persistent connection and the requests are pipelined. Specifically, the second and third requests are sent in rapid succession as soon as enough of the main page has been retrieved to identify that the images must be fetched. The responses for these requests follow eventually. This method cuts down the time that the server is idle, so it further improves performance.

Persistent connections do not come for free, however. A new issue that they raise is when to close the connection. A connection to a server should stay open while the page loads. If the connection remains open, the next request can be sent immediately. However, there is no guarantee that the client will make another request of the server any time soon. In practice, clients and servers usually keep persistent connections open until they have been idle for a short time or they have a large number of open connections and need to close some.

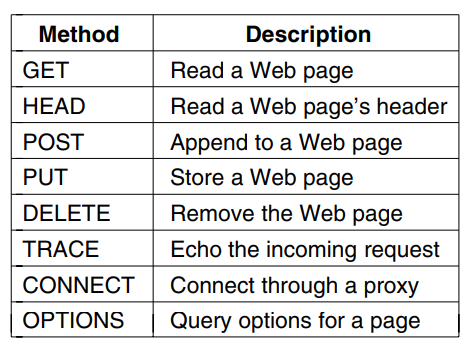
It is also possible to send one request per TCP connection, but run multiple TCP connections in parallel. This **parallel connection** method was widely used by browsers before persistent connections. It has the same disadvantage as sequential connections—extra overhead—but much better performance.

However, running many TCP connections to the same server is discouraged. The reason is that TCP performs congestion control for each connection independently. As a consequence, the connections compete against each other, causing added packet loss, and in aggregate are more aggressive users of the network than an individual connection. Persistent connections are superior and used in preference to parallel connections because they avoid overhead and do not suffer from congestion problems.

Table

Description automatically generated

**------------- Methods -------------**

Operations, called **methods**, other than just requesting a Web page are supported. Each request consists of one or more lines of ASCII text, with the first word on the first line being the name of the method requested.

Every request gets a response consisting of a status line, and possibly additional information. The status line contains a three-digit status code telling whether the request was satisfied and, if not, why not.

**------------- Caching -------------**

People often return to Web pages that they have viewed before, and related Web pages often have the same embedded resources. It would be very wasteful to fetch all of these resources for these pages each time they are displayed because the browser already has a copy.

Squirreling away pages that are fetched for subsequent use is called caching. The advantage is that when a cached page can be reused, it is not necessary to repeat the transfer. HTTP has built-in support to help clients identify when they can safely reuse pages.

The pages are usually kept on disk so that they can be used when the browser is run at a later date.