Chapter 4. Application layer services: DNS, WWW, E-mail

Many protocols have been defined for networked applications. In this chapter there are described some of the important applications that are used on the Internet. It is first explained the Domain Name System (DNS) that enables hosts to be identified by human-friendly names instead of the IPv4 or IPv6 addresses that are used by the network. Then, there are described the protocols used on World Wide Web and the operation of electronic mail, one of the first killer applications on the global Internet.

4.1 The Domain Name System (DNS)

In the early days of the Internet, there were only a few number of hosts (mainly minicomputers) connected to the network. The most popular applications were remote login and file transfer. By 1983, there were already five hundred hosts attached to the Internet. Each of these hosts was identified by a unique IPv4 address. Forcing human users to remember the IPv4 addresses of the remote hosts that they want to use was not user-friendly. Human users prefer to remember names, and use them when needed. Using names as aliases for addresses is a common technique in Computer Science. It simplifies the development of applications and allows the developer to ignore the low level details. For example, by using a programming language instead of writing machine code, a developer can write software without knowing whether the variables that it uses are stored in memory or inside registers.

Because names are at a higher level than addresses, they allow (both in the example of programming above, and on the Internet) to treat addresses as mere technical identifiers, which can change at will. Only the names are stable. On today's Internet, where switching to another ISP means changing your IP addresses, the user-friendliness of domain names is less important (they are not often typed by users) but their stability remains a very important, may be their most important property.

The first solution that allowed applications to use names was the hosts.txt file. This file is similar to the symbol table found in compiled code. It contains the mapping between the name of each Internet host and its associated IP address 2. It was maintained by Stanford Research Institute (SRI) International that coordinated the Network Information Center (NIC). When a new host was connected to the network, the system administrator had to register its name and IP address at the NIC. The NIC updated the hosts.txt file on its server. All Internet hosts regularly retrieved the updated hosts.txt file from the server maintained by SRI. This file was stored at a well-known location on each Internet host and networked applications could use it to find the IP address corresponding to a name.

A hosts.txt file can be used when there are up to a few hundred hosts on the network. However, it is clearly not suitable for a network containing thousands or millions of hosts. A key issue in a large network is to define a suitable naming scheme. The ARPANet initially used a flat naming space, i.e. each host was assigned a unique name. To limit collisions between names, these names usually contained the name of the institution and a suffix to identify the host inside the institution (a kind of poor man's hierarchical naming scheme). On the ARPANet few institutions had several hosts connected to the network.

4.1.1 The DNS Name Space

The set of all names used with DNS constitutes the DNS name space. However, the limitations of a flat naming scheme became clear before the end of the ARPANet and RFC 819 proposed a hierarchical naming scheme. While RFC 819 discussed the possibility of organising the names as a directed graph, the Internet opted eventually for a tree structure capable of containing all names. In this tree, the top-level domains are those that are directly attached to the root. The first top-level domain was ".arpa 3". This top-level name was initially added as a suffix to the names of the hosts attached to the ARPANet and listed in the hosts.txt file. In 1984, the .gov, .edu, .com, .mil and .org generic top-level domain names were added and RFC 1032 proposed the utilisation of the two letter ISO-3166 country codes as top-level domain names. Since ISO-3166 defines a two letter code for each country recognised by the United Nations, this allowed all countries to automatically have a top-level domain. These domains include .be for Belgium, .fr for France, .us for the USA, .ie for Ireland or .ro for Romania. Today, the set of top-level domain-names is managed by the Internet Corporation for Assigned Names and Numbers (ICANN). Recently, ICANN added a dozen of generic top-level domains that are not related to a country and the .cat top-level domain has been registered for the Catalan language. There are ongoing discussions within ICANN to increase the number of top-level domains.

Each top-level domain is managed by an organisation that decides how sub-domain names can be registered. Most top-level domain names use a first-come first served system, and allow anyone to register domain names, but there are some exceptions. For example, .gov is reserved for the US government, .int is reserved for international organisations and names in the .ca are mainly reserved for companies or users who are present in Canada.

The set of all names used with DNS constitutes the DNS name space. This space is partitioned hierarchically and is case insensitive, similar to computer file system folders (directories) and files. The current DNS name space is a tree of domains with an unnamed root at the top. The top echelons of the tree are the so-called top-level domains (TLDs), which include generic TLDs (gTLDs), country-code TLDs (ccTLDs), and internationalized country-code TLDs (IDN ccTLDs), plus a special infrastructure TLD called, for historical reasons, ARPA. These form the top levels of a naming tree with the form shown in figure 4.1

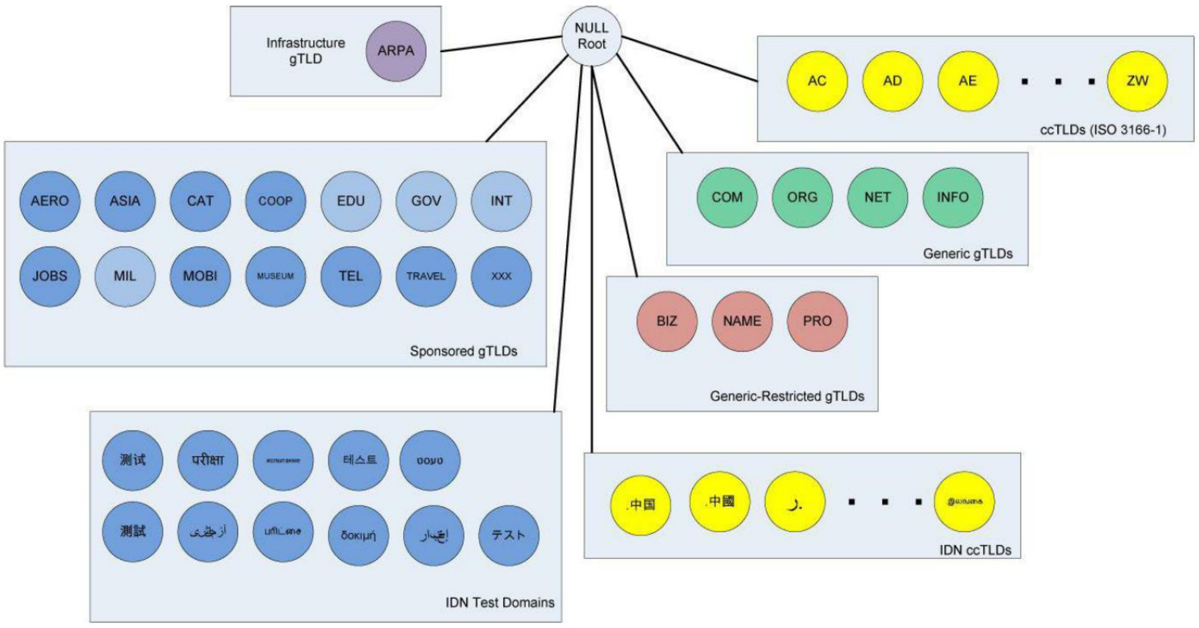

Figure 4.1 Top-level Domains

There are five commonly used groups of TLDs, and one group of specialized domains being used for Internationalized Domain Names (IDNs). The history of IDNs, one piece of the “internationalization” or “i18n” of the Internet, is long and somewhat complicated. Across the world, there are multiple languages, and each uses one or more written scripts. While the Unicode standard aims to capture the entire set of characters, many characters look the same but have different Unicode values. Furthermore, characters written as text may flow from right to left, left to right, or (when combining certain texts with others) in both directions.

The gTLDs are grouped into categories: generic, generic-restricted, and sponsored. The generic gTLDs (generic appears twice) are open for unrestricted use. The others (generic-restricted and sponsored) are limited to various sorts of uses or are constrained as to what entity may assign names from the domain.

The ccTLDs include the two-letter country codes specified by the ISO 3166 standard, plus five that are not: uk, su, ac, eu, and tp (the last one is being phased out). Because some of these two-letter codes are suggestive of other uses and meanings, various countries have been able to find commercial windfalls from selling names within their ccTLDs. For example, the domain name cnn.tv is really a registration in the Pacific island of Tuvalu, which has been selling domain names associated with the television entertainment industry. Creating a name in such an unconventional way is sometimes called a domain hack.

4.1.2 DNS Naming Syntax

The names below a TLD in the DNS name tree are further partitioned into groups known as subdomains. This is very common practice, especially for the ccTLDs. For example, most educational sites in England use the suffix “.ac.uk”, whereas names for most for-profit companies there end in the suffix “.co.uk”.

The example names we have seen so far are known as fully qualified domain names (FQDNs). They are sometimes written more formally with a trailing period (e.g., ucv.ro.). This trailing period indicates that the name is complete; no additional information should be added to the name when performing a name resolution. In contrast to the FQDN, an unqualified domain name, which is used in combination with a default domain or domain search list set during system configuration, has one or more strings appended to the end. When a system is configured, it is typically assigned a default domain extension and search list using DHCP. For example, the default domain dcti.ucv.ro might be configured in systems at the Department of Computers and Information Technology at University of Craiova. If a user on one of these machines types in the name www, the local resolver software converts this name to the FQDN www.dcti.ucv.ro. before invoking a resolver to determine www's IP address.

A domain name consists of a sequence of labels separated by periods. The name represents a location in the name hierarchy, where the period is the hierarchy delimiter and descending down the tree takes place from right to left in the name. For example, the FQDN www.dcti.ucv.ro. contains a host name label (www) in a three-level-deep domain (dcti.ucv.ro). Starting from the root, and working from right to left in the name, the TLD is "ro" (the ccTLD for Romania), ucv is shorthand for University of Craiova, dcti is shorthand for "Departamentul de Calculatoare si Tehnologia Informatiei", and finally www is name of the host suggesting that the host is a website. Labels are case insensitive for matching purposes, so the name WWW.UCV.RO is equivalent to www.ucv.ro or Www.UCv.Ro. Each label can be up to 63 characters long, and an entire FQDN is limited to at most 255 (1-byte) characters.

The hierarchical structure of the DNS name space allows different administrative authorities to manage different parts of the name space. For example, creating a new DNS name of the form "research.dcti.ucv.ro" would likely require dealing with the owner of the "dcti.ucv.ro" subdomain only. The "ucv.ro" and "ro" portions of the name space would not require alteration, so the owners of those would not need to be bothered. This feature of DNS is one key aspect of its scalability. That is, no single entity is required to administer all the changes for the entire DNS name space. Indeed, creating a hierarchical structure for names was one of the first responses in the Internet community to the pressures of scaling and a major motivator for the structure used today. The original Internet naming scheme was flat (i.e., no hierarchy), and a single entity was responsible for assigning, maintaining, and distributing the list of non-conflicting names. Over time, as more names were required and more changes were being made, this approach became unworkable.

4.1.3 Name Servers and Zones

Management responsibility for portions of the DNS name space is assigned to individuals or organizations. A person given responsibility for managing part of the active DNS name space (one or more domains) is supposed to arrange for at least two name servers or DNS servers to hold information about the name space so that users of the Internet can perform queries on the names. The collection of servers forms the DNS (service) itself, a distributed system whose primary job is to provide name-toaddress mappings. However, it can also provide a wide array of additional information.

The unit of administrative delegation, in the language of DNS servers, is called a zone. A zone is a subtree of the DNS name space that can be administered separately from other zones. Every domain name exists within some zone, even the TLDs that exist in the root zone. Whenever a new record is added to a zone, the DNS administrator for the zone allocates a name and additional information (usually an IP address) for the new entry and enters these into the name server's database. At a small campus, for example, one person could do this each time a new server is added to the network, but in a large enterprise the responsibility would have to be delegated (probably by departments or other organizational units), as one person likely could not keep up with the work.

A DNS server can contain information for more than one zone. At any hierarchical change point in a domain name (i.e., wherever a period appears), a different zone and containing server may be accessed to provide information for the name. This is called a delegation. A common delegation approach uses a zone for implementing a second-level domain name, such as "ucv.ro". In this domain, there may be individual hosts (e.g., www.ucv.ro) or other domains (e.g., "dcti.ucv.ro"). Each zone has a designated owner or responsible party who is given authority to manage the names, addresses, and subordinate zones within the zone. Often this person manages not only the contents of the zone but also the name servers that contain the zone's database(s). Zone information is supposed to exist in at least two places, implying that there should be at least two servers containing information for each zone. This is for redundancy; if one server is not functioning properly, at least one other server is available. All of these servers contain identical information about a zone. Typically, among the servers, a primary server contains the zone database in a disk file, and one or more secondary servers obtain copies of the database in its entirety from the primary using a process called a zone transfer. DNS has a special protocol for performing zone transfers, but copies of a zone's contents can also be obtained using other means.

4.1.4 Caching

Name servers contain information such as name-to-IP-address mappings that may be obtained from three sources. The name server obtains the information directly from the zone database, as the result of a zone transfer (e.g., for a slave server), or from another server in the course of processing a resolution. In the first case, the server is said to contain authoritative information about the zone and may be called an authoritative server for the zone. Such servers are identified by name within the zone information.

Most name servers (except some of the root and TLD servers) also cache zone information they learn, up to a time limit called the time to live (TTL). They use this cached information to answer queries. Doing so can greatly decrease the amount of DNS message traffic that would otherwise be carried on the Internet. When answering a query, a server indicates whether the information it is returning has been derived from its cache or from its authoritative copy of the zone. When cached information is returned, it is common for a server to also include the domain names of the name servers that can be contacted to retrieve authoritative information about the corresponding zone.

Each DNS record (e.g., name-to-IP-address mapping) has its own TTL that controls how long it can be cached. These values are set and altered by the zone administrator when necessary. The TTL dictates how long a mapping can be cached anywhere within DNS, so if a zone changes, there still may exist cached data within the network, potentially leading to incorrect DNS resolution behaviour until expiry of the TTL. For this reason, some zone administrators, anticipating a change to the zone contents, first reduce the TTL before implementing the change. Doing so reduces the window for incorrect cached data to be present in the network.

It is worth mentioning that caching is applied both for successful resolutions and for unsuccessful resolutions (called negative caching). If a request for a particular domain name fails to return a record, this fact is also cached. Doing so can help to reduce Internet traffic when errant applications repeatedly make requests for names that do not exist. Negative caching was changed from optional to mandatory by RFC2308.

In some network configurations (e.g., those using older UNIX-compatible systems), the cache is maintained in a nearby name server, not in the resolvers resident in the clients. Placing the cache in the server allows any hosts on the LAN that use the nearby server to benefit from the server's cache but implies a small delay in accessing the cache over the local network. In Windows and more recent systems (e.g., Linux), the client can maintain a cache, and it is made available to all applications running on the same system. In Windows, this happens by default, and in Linux, it is a service that can be enabled or disabled.

4.1.5 The DNS Protocol

The DNS protocol consists of two main parts: a query/response protocol used for performing queries against the DNS for particular names, and another protocol for name servers to exchange database records (zone transfers). It also has a way to notify secondary servers that the zone database has evolved and a zone transfer is necessary (DNS Notify), and a way to dynamically update the zone (dynamic updates). By far, the most typical usage is a simple query/response to look up the IPv4 address that corresponds to a domain name. Most often, DNS name resolution is the process of mapping a domain name to an IPv4 address, although IPv6 addresses mappings work in essentially the same way. DNS query/response operations are supported over the distributed DNS infrastructure consisting of servers deployed locally at each site or ISP, and a special set of root servers. There is also a special set of generic top-level domain servers used for scaling some of the larger gTLDs, including COM and NET. As of mid-2011, there are 13 root servers named by the letters A through M (see [ROOTS] for more information about them); 9 of them have IPv6 addresses. There are also 13 gTLD servers, also labeled A through M; 2 of them have IPv6 addresses. By contacting a root server and possibly a gTLD server, the name server for any TLD in the Internet can be discovered. These servers are mutually coordinated to provide the same information.

Some of them are not a single physical server but instead a group of servers (over 50 for the J root server) that use the same IP address (i.e., using IP anycast addressing; see Chapter 2).

A full resolution that is unable to benefit from pre-existing cached entries takes place among several entities, as shown in Figure 4.2.

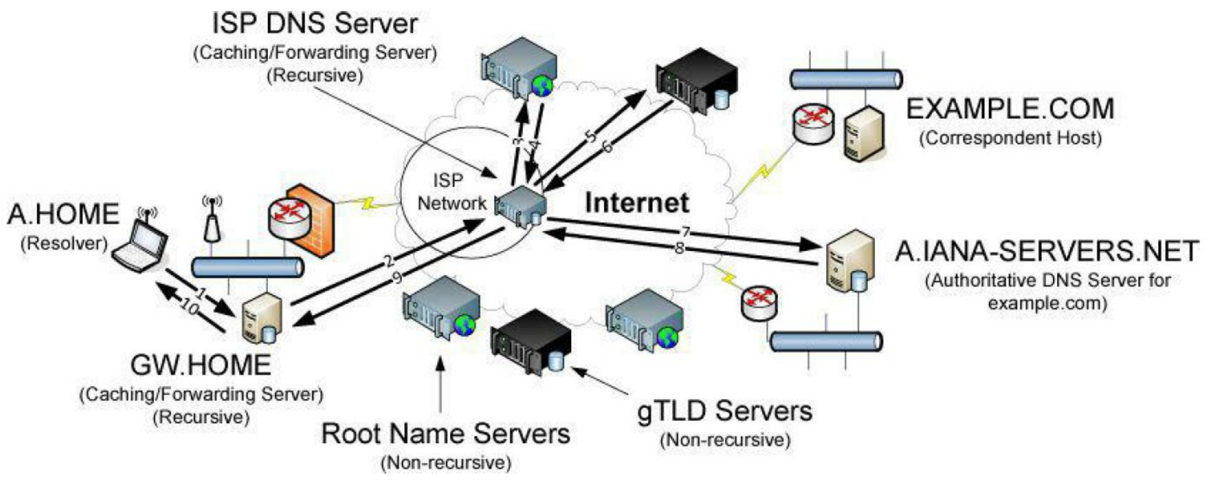

Figure 4.2. A typical recursive DNS query

Here, we have a laptop called A.HOME residing nearby the DNS server GW.HOME. The domain HOME is private, so it is not known to the Internet—only locally at the user's residence. When a user on A.HOME wishes to connect to the host EXAMPLE.COM (e.g., because a Web browser has been instructed to access the page http://EXAMPLE.COM), A.HOME must determine the IP address for the server EXAMPLE.COM. Assuming it does not know this address already (it might if it has accessed the host recently), the resolver software on A.HOME first makes a request to its local name server, GW.HOME. This is a request to convert the name EXAMPLE.COM into an address and constitutes message 1 (labelled on an arrow in Figure 4.2).

If GW.HOME does not already know the IP address for EXAMPLE.COM or the name servers for either the EXAMPLE.COM domain or the COM TLD, it forwards the request to another DNS server (called recursion). In this case, a request (message 2) goes to an ISP-provided DNS server. Assuming that this server also does not know the required address or other information, it contacts one of the root name servers (message 3). The root servers are not recursive, so they do not process the request further but instead return the information required to contact a name server for the COM TLD. For example, it might return the name A.GTLD-SERVERS.NET and one or more of its IP addresses (message 4). With this information, the ISP-provided server contacts the gTLD server (message 5) and discovers the name and IP addresses of the name servers for the domain EXAMPLE.COM (message 6). In this case, one of the servers is A.IANASERVERS.NET.

Given the correct server for the domain, the ISP-provided server contacts the appropriate server (message 7), which responds with the requested IP address (message 8). At this point, the ISP-provided server can respond to GW.HOME with the required information (message 9). GW.HOME is now able to complete the initial query and responds to the client with the desired IPv4 and/or IPv6 address(es) (message 10).

From the perspective of A.HOME, the local name server was able to perform the request. However, what really happened is a recursive query, where the GW.HOME and ISP-provided servers in turn made additional DNS requests to satisfy A.HOME's query. In general, most name servers perform recursive queries such as this. The notable exceptions are the root servers and other TLD servers that do not perform recursive queries. These servers are a relatively precious resource, so encumbering them with recursive queries for every machine that performs a DNS query would lead to poor global Internet performance.

4.1.6 UDP or TCP

The well-known port number for DNS is 53, for both UDP and TCP. The most common format uses the UDP/IPv4 datagram structure.

When a resolver issues a query and the response comes back with the TC bit field set ("truncated"), the size of the true response exceeded 512 bytes, so only the first 512 bytes are returned by the server. The resolver may issue the request again, using TCP, which now must be a supported configuration. This allows more than 512 bytes to be returned because TCP breaks up large messages into multiple segments.

When a secondary name server for a zone starts up, it normally performs a zone transfer from the primary name server for the zone. Zone transfers can also be initiated by a timer or as a result of a DNS NOTIFY message. Full zone transfers use TCP as they can be large. Incremental zone transfers, where only the updated entries are transferred, may use UDP at first but switch to TCP if the response is too large, just like a conventional query. When UDP is used, both the resolver and the server application software must perform their own timeout and retransmission.

4.1.7 Address (A, AAAA) and Name Server (NS) Records

The most important records within DNS are the address (A, AAAA) and name server (NS) records. The A records contain 32-bit IPv4 addresses, and AAAA (called "quad-A") records contain IPv6 addresses. An NS record contains the name of an authoritative DNS server that contains information for a particular zone. Because the name of a DNS server alone is not sufficient to perform a query, the IP address(es) of these servers is also typically provided as a so-called glue record in the additional information section of DNS responses. Indeed, such glue records are required to avoid loops whenever the names of the authoritative name servers use the same domain name for which they are authoritative. (Consider how ns1.example.com would be resolved if the name server for example.com was ns1.example.com.). The structure of A, AAAA, and NS records is depicted in figure 4.3.

We can see the structure of A, AAAA, and NS records using the dig tool provided on most Linux/UNIX-like systems or nslookup provided by Windows based systems.

```
roedu.net           IN   NS     ns.nren.ro.
roedu.net           IN   NS     ns1.roedu.net.
roedu.net           IN   NS     pub.pub.ro.
alpha.roedu.net     IN   A      81.180.250.131
alpha.roedu.net     IN   AAAA   2001:b30:1::131
relay.roedu.net     IN   A      81.180.250.138
proxy1.roedu.net    IN   A      37.128.225.250
ns.nren.ro          IN   A      81.180.250.190
ns.nren.ro          IN   AAAA   2001:b30:1::190
```

Figure 4.3 DNS records

4.2 World Wide Web

In the early days of the Internet was mainly used for remote terminal access with telnet, email and file transfer. The default file transfer protocol, FTP, was widely used and FTP clients and servers are still included in most operating systems.

In the late 1980s, high energy physicists working at CERN had to efficiently exchange documents about their ongoing and planned experiments. Tim Berners-Lee evaluated several of the documents sharing techniques that were available at that time. As none of the existing solutions met CERN's requirements, they choose to develop a completely new document sharing system. This system was initially called the mesh, but was quickly renamed the World Wide Web. The starting point for the World Wide Web is hypertext documents. A hypertext document is a document that contains references (hyperlinks) to other documents that the reader can immediately access. Hypertext was not invented for the World Wide Web. The idea of hypertext documents was proposed in 1945 and the first experiments were done during the 1960s. Compared to the hypertext documents that were used in the late 1980s, the main innovation introduced by the World Wide Web was to allow hyperlinks to reference documents stored on remote machines.

In 1993, the Web started to grow rapidly, which was mainly due to the National Center for Supercomputing Applications (NCSA) developing a Web browser program called Mosaic, an X Window System-based application. This application provided the first graphical user interface to the Web and made browsing more convenient. Today, there are Web browsers and servers available for nearly all platforms. The rapid growth in popularity of the Web is due to the flexible way people can navigate through worldwide resources in the Internet and retrieve them.

The number of Web servers is also increasing rapidly, and the traffic over port 80 on the NSF backbone has had a phenomenal rate of growth, too

4.2.1 Web browsers

Generally, a browser is referred to as an application that provides access to a Web server. Depending on the implementation, browser capabilities and thus structures vary. A Web browser, at a minimum, consists of an Hypertext Markup Language (HTML) interpreter and HTTP client that is used to retrieve HTML Web pages. Besides this basic requirement, many browsers also support FTP, NNTP, e-mail (POP and SMTP clients), among other features, with an easy-to-manage graphical interface. Figure 4.4 illustrates a basic Web browser structure.

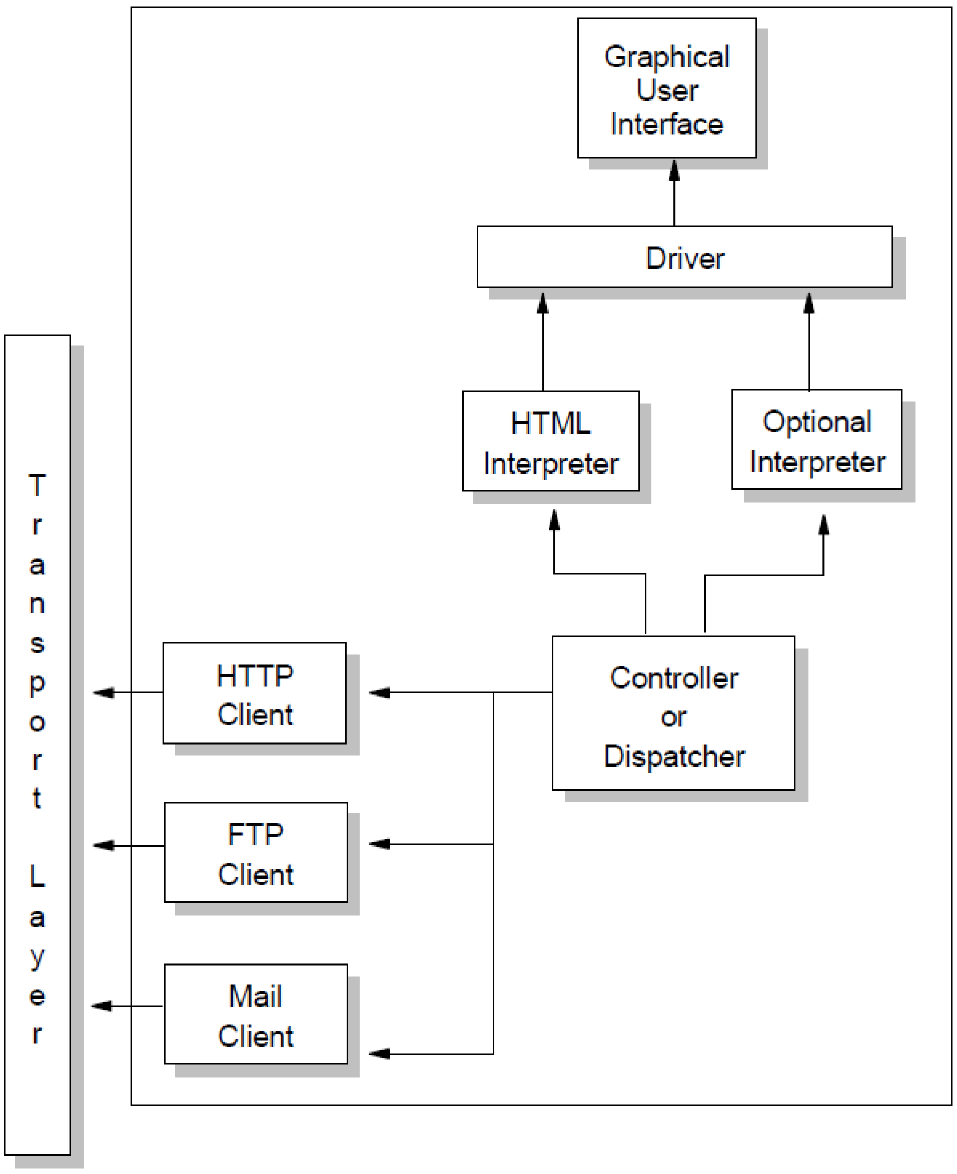

Figure 4.4 Structure of a web browser

As with many other Internet facilities, the Web uses a client/server processing model. The Web browser is the client component. Examples of Web browsers include Mozilla Firefox, Google Chrome, Safari, Opera, and Microsoft Internet Explorer.

Web browsers are responsible for formatting and displaying information, interacting with the user, and invoking external functions, such as Telnet, or external viewers for data types that Web browsers do not directly support. Web browsers have become the “universal client” for the GUI workstation environment.

4.2.2 Web servers

Web servers are responsible for servicing requests for information from Web browsers. The information can be a file retrieved from the server's local disk, or it can be generated by a program called by the server to perform a specific application function.

There are a number of public-domain Web servers available for a variety of platforms, including most UNIX variants, as well as personal computer environments such as Microsoft Windows. Some well-known public domain servers are CERN, NCSA httpd, and Apache servers.

4.2.3 WWW components

A document sharing system such as the World Wide Web is composed of three important parts.

1. A standardised addressing scheme that allows unambiguous identification of documents
2. A standard document format : the HyperText Markup Language
3. A standardised protocol that facilitates efficient retrieval of documents stored on a server

The first components of the World Wide Web are the Uniform Resource Identifiers (URI). A URI is a character string that unambiguously identifies a resource on the World Wide Web.

The first component of a URI is its scheme. A scheme can be seen as a selector, indicating the meaning of the fields after it. In practice, the scheme often identifies the application-layer protocol that must be used by the client to retrieve the document, but it is not always the case. Some schemes do not imply a protocol at all and some do not indicate a retrievable document. The most frequent scheme is http that will be described later. The characters ‘: and // follow the scheme of any URI (e.g. http://).

The second part of the URI is the authority. With retrievable URI, this includes the DNS name or the IP address of the server where the document can be retrieved using the protocol specified via the scheme. This name can be preceded by some information about the user (e.g. a user name) who is requesting the information. Earlier definitions of the URI allowed the specification of a user name and a password before the @ character, but this is now deprecated as placing a password inside a URI is insecure. The host name can be followed by the semicolon character and a port number. A default port number is defined for some protocols and the port number should only be included in the URI if a non-default port number is used (for other protocols, techniques like service DNS records are used).

The third part of the URI is the path to the document. This path is structured as filenames on a Unix host (but it does not imply that the files are indeed stored this way on the server). If the path is not specified, the server will return a default document. The last two optional parts of the URI are used to provide a query and indicate a specific part (e.g. a section in an article) of the requested document. Sample URIs are shown below.

http://tools.ietf.org/html/rfc3986.html
mailto:infobot@example.com?subject=current-issue
http://docs.python.org/library/basehttpserver.html?highlight=http#BaseHTTPServer.BaseHTTPRequestHandler
telnet://[2001:6a8:3080:3::2]:80/
ftp://cnn.example.com&story=breaking_news@10.0.0.1/top_story.htm

4.2.4 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol is a protocol designed to allow the transfer of Hypertext Markup Language (HTML) documents. HTML is a tag language used to create hypertext documents. Hypertext documents include links to other documents that contain additional information about the highlighted term or subject. Such documents can contain other elements apart from text, such as graphic images, audio and video clips, Java applets, and even virtual reality worlds (which are described in VRML, a scripting language for that kind of elements). See "Hypertext Markup Language (HTML)" on page 615 for more information about HTML.

Nowadays, both HTTP 1.0 and HTTP 1.1 are stable specifications; therefore, W3C has closed the HTTP activity after its goal of creating a stable and weakness-free HTTP standard has been achieved.

HTTP is based on request-response activity. A client, running an application called a browser, establishes a connection with a server and sends a request to the server in the form of a request method. The server responds with a status line, including the message's protocol version and a success or error code, followed by a message containing server information, entity information, and possible body content.

An HTTP transaction is divided into four steps:

1. The browser opens a connection.
2. The browser sends a request to the server.
3. The server sends a response to the browser.
4. The connection is closed.

On the Internet, HTTP communication generally takes place over TCP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used. Except for experimental applications, current practice requires that the connection be established by the client prior to each request and closed by the server after sending the response. Both clients and servers should be aware that either party can close the connection prematurely, due to user action, automated timeout, or program failure, and should handle such closing in a predictable and desirable fashion. In any case, the closing of the connection by either or both parties always terminates the current request, regardless of its status. In simple terms, HTTP is a stateless protocol because it does not keep track of the connections. To load a page including two graphics, for example, a graphic-enabled browser will open three TCP connections: one for the page and two for the graphics. Most browsers, however, are able to handle several of these connections simultaneously.

If a request depends on the information exchanged during a previous connection, this information has to be kept outside the protocol. One way of tracking such persistent information is the use of cookies. A cookie is a set of information that is exchanged between a client Web browser and a Web server during an HTTP transaction. The maximum size of a cookie is 4 KB. All these pieces of information, or cookies, are then stored in one single file and placed in the directory of the Web browser. If cookies are disabled, that file is automatically deleted. A cookie can be retrieved and checked by the server at any subsequent connection. Because cookies are regarded as a potential privacy exposure, a Web browser should allow the user to decide whether or not he or she will accept cookies from a particular server. While cookies merely serve the purpose of keeping some kind of state for HTTP connections, secure client and server authentication is provided by the Secure Sockets Layer (SSL).

4.2.4.1 HTTP operation

HTTP was initially designed to share self-contained text documents. For this reason, and to ease the implementation of clients and servers, the designers of HTTP chose to open a TCP connection for each HTTP request. This implies that a client must open one TCP connection for each URI that it wants to retrieve from a server as illustrated in the figure 4.5. For a web page containing only text documents this was a reasonable design choice as the client usually remains idle while the (human) user is reading the retrieved document.

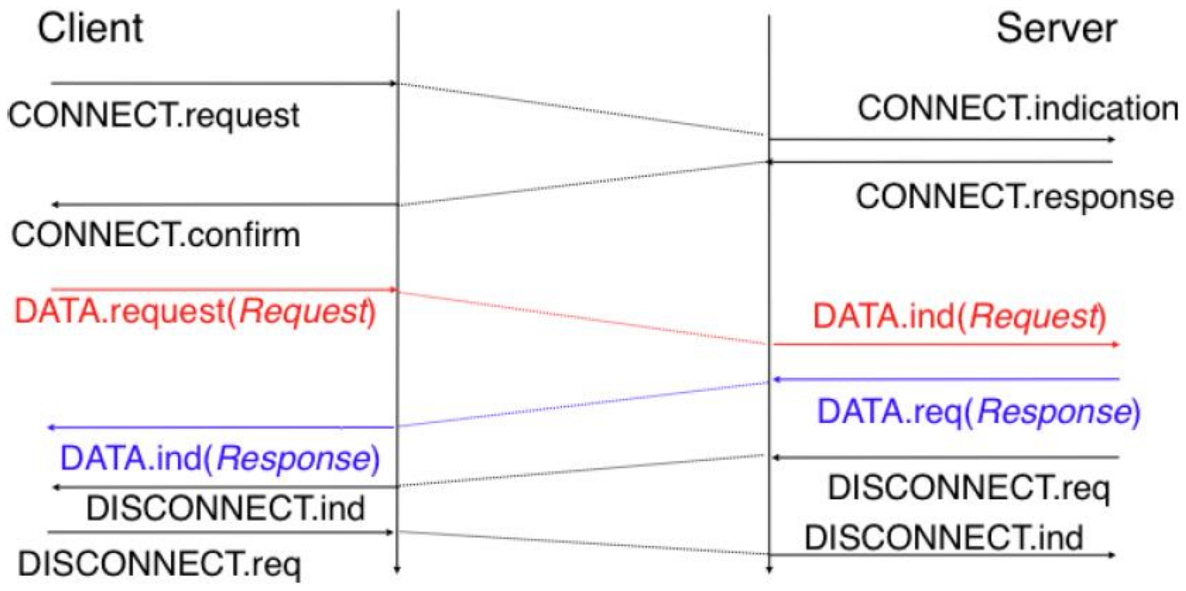

Figure 4.5 HTTP 1.0 and the underlying TCP connection

However, as the web evolved to support richer documents containing images, opening a TCP connection for each URI became a performance problem. Indeed, besides its HTML part, a web page may include dozens of images or more. Forcing the client to open a TCP connection for each component of a web page has two important drawbacks. First, the client and the server must exchange packets to open and close a TCP connection as we will see later. This increases the network overhead and the total delay of completely retrieving all the components of a web page. Second, a large number of established TCP connections may be a performance bottleneck on servers.

This problem was solved by extending HTTP to support persistent TCP connections. A persistent connection is a TCP connection over which a client may send several HTTP requests. This is illustrated in the figure 4.6.

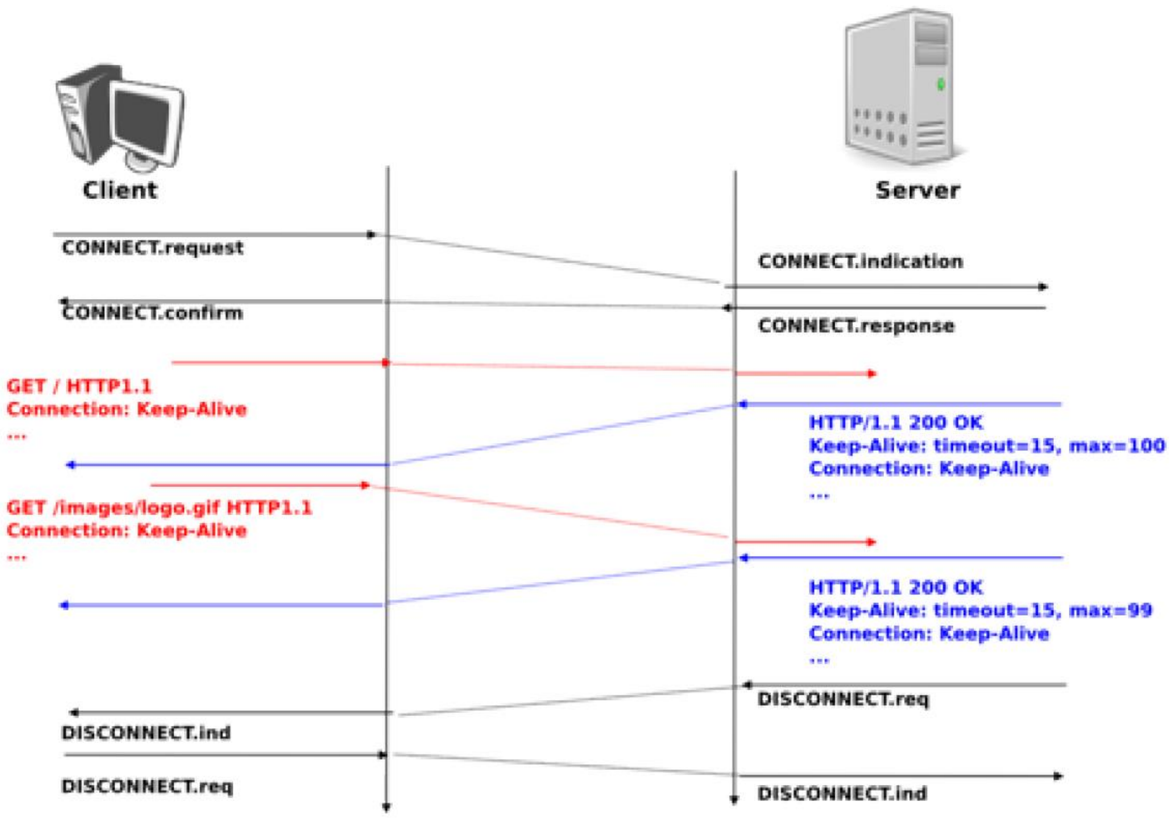

Figure 4.6 HTTP 1.1 persistent connections

In most cases, the HTTP communication is initiated by the user agent requesting a resource on the origin server. In the simplest case, the connection is established through a single connection between the user agent and the origin server, as shown in Figure 4.7.

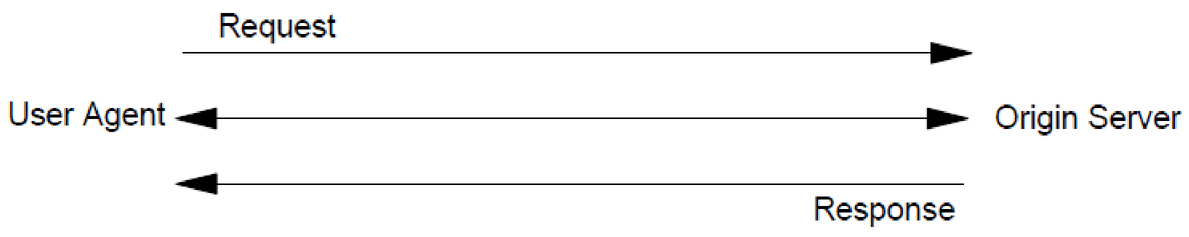

Figure 4.7 Single client/server connection

In some cases, there is no direct connection between the user agent and the origin server. There is one (or more) intermediary between the user agent and origin server, such as a proxy, gateway, or tunnel. Requests and responses are evaluated by the intermediaries and forwarded to the destination or another intermediary in the request-response chain, as shown in Figure 4.8.

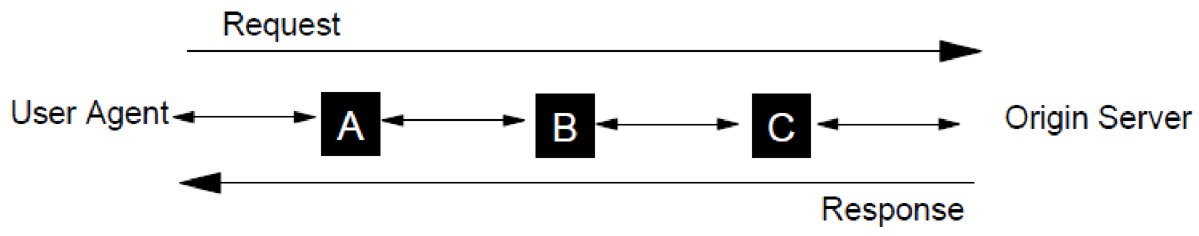

Figure 4.8 Client/server connection with intermediaries in between

A proxy can handle the content of the data and therefore modify the data accordingly. When a request comes to a proxy, it rewrites all or part of the message and forwards the message to the next destination. A gateway receives the message and sends the message to the underlying protocols with an appropriate format. A tunnel does not deal with the content of the message; therefore, it simply forwards the message as it is.

Proxies and gateways in general, can handle the caching of HTTP messages. This can dramatically reduce the response time and IP traffic in the network. Because tunnels cannot understand the message content, they cannot store cached data of HTTP messages. In figure 4.6, if one of the intermediaries (A, B, and C) employs an internal cache for HTTP messages, the user agent can get a response from the intermediary if it is previously cached from the origin server in the response chain. Figure 4.9 illustrates that A has a cached copy of an earlier response from the origin server in the response chain. Therefore, if the server response for the request is not already cached in the user agent's internal cache, it can directly be obtained from A.

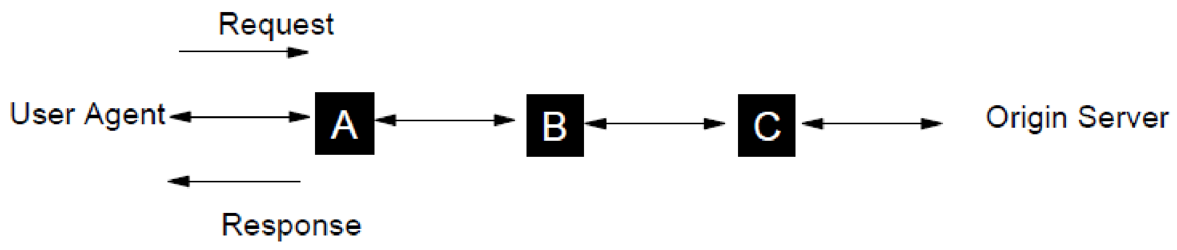

Figure 4.9 Cached server response

Caching is not applicable to all server responses. Caching behaviour can be modified by special requests to determine which server responses can or cannot be cached. For this purpose, server responses can be marked as non-cachable, public, or private (cannot be cached in a public cache).

4.2.4.2 HTTP parameters

Here are provided some of the HTTP protocol parameters.

- HTTP version
  HTTP uses a <major>.<minor> numbering scheme to indicate the versions of the protocol. The furthermost connection is performed according to the protocol versioning policy. The <major> number is incremented when there are significant changes in protocol, such as changing a message format. The <minor> number is incremented when the changes do not affect the message format. The version of HTTP messages is sent by an HTTP-Version field in the first line of the message. The HTTP-Version field is in the following format (refer to RFC 2822 for augmented Backus-Naur Form): HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT
- Uniform Resource Identifiers (URIs)
  Uniform Resource Identifiers are generally referred to as WWW addresses and a combination of Uniform Resource Locators (URLs) and Uniform Resource Names (URNs). In fact, URIs are strings that indicate the location and name of the source on the server. See RFC 2616 and RFC 3986 for more details about the URI and URL syntax.
- HTTP URL
  The HTTP URL scheme enables you to locate network resources through the HTTP protocol. It is based on the URI Generic Syntax. The general syntax of a URL scheme is: HTTP_URL = "http" "//" host [":" port] [abs_path] The port number is optional. If it is not specified, the default value is 80.

4.2.4.3 HTTP message

HTTP messages consist of the following fields:

- Message types
  A HTTP message can be either a client request or a server response. The following string indicates the HTTP message type: HTTP-message = Request | Response
- Message header
  The HTTP message header field can be one of the following:
  - General header
  - Request header
  - Response header
  - Entity header
- Message body
  Message body can be referred to as entity body if there is no transfer coding has been applied. Message body simply carries the entity body of the relevant request or response.
- Message length
  Message length indicates the length of the message body if it is included.
- General header field
  General header fields can apply both request and response messages. Currently defined general header field options are as follows:
  - Cache-Control
  - Connection
  - Date

- Pragma
- Transfer-Encoding
- Upgrade
- Via

4.3 Electronic mail

4.3.1 Overview

Electronic mail, or email, is a very popular application in computer networks such as the Internet. E-mail appeared in the early 1970s and allows users to exchange text based messages. Initially, it was mainly used to exchange short messages, but over the years its usage has grown. It is now not only used to exchange small, but also long messages that can be composed of several parts as we will see later.

Before looking at the details of Internet email, let us consider a simple scenario illustrated in the figure 4.10, where Alice sends an email to Bob. Alice prepares her email by using an email client and sends it to her email server. Alice's email server extracts Bob's address from the email and delivers the message to Bob's server. Bob retrieves Alice's message on his server and reads it by using his favourite email client or through his webmail interface.

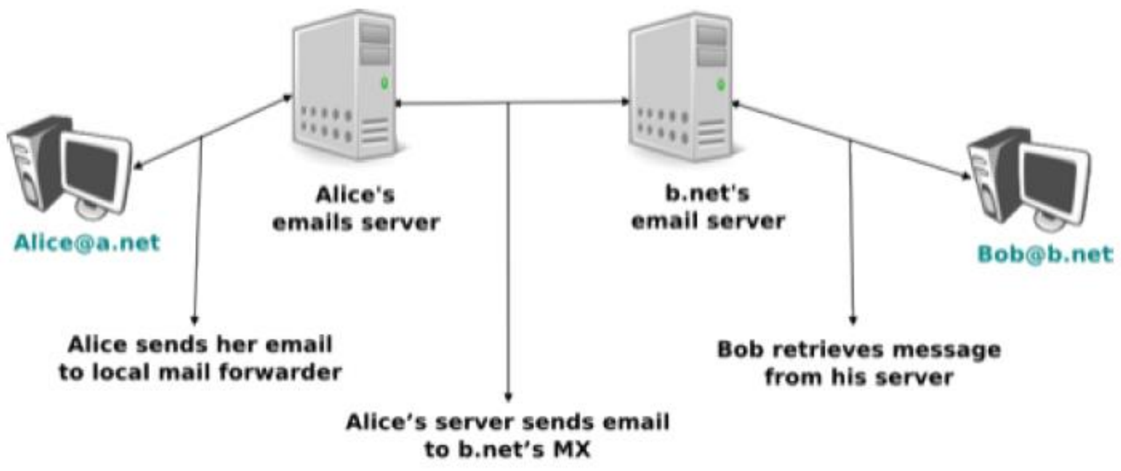
Alice@a.net
Alice's emails server
b.net's email server
Bob@b.net
Alice sends her email to local mail forwarder
Alice's server sends email to b.net's MX
Bob retrieves message from his server

Figure 4.10 Simplified architecture of the Internet email

The email system is composed of four components:

- a message format, that defines how valid email messages are encoded
- protocols, that allow hosts and servers to exchange email messages
- client software, that allows users to easily create and read email messages
- software, that allows servers to efficiently exchange email messages

Email messages, like postal mail, are composed of two parts:
• a header that plays the same role as the letterhead in regular mail. It contains metadata about the message.

• the body that contains the message itself.
Email messages are entirely composed of lines of ASCII characters. Each line can contain up to 998 characters and is terminated by the CR and LF control characters. The lines that compose the header appear before the message body. An empty line, containing only the CR and LF characters, marks the end of the header.

The email header contains several lines that all begin with a keyword followed by a colon and additional information. Two of these header lines are mandatory and must appear in all email messages:

- The sender address. This header line starts with From:. This contains the (optional) name of the sender followed by its email address between < and >. Email addresses are always composed of a username followed by the @ sign and a domain name.
- The date. This header line starts with Date:. RFC 5322 precisely defines the format used to encode a date.

Other header lines appear in most email messages. The Subject: header line allows the sender to indicate the topic discussed in the email. Three types of header lines can be used to specify the recipients of a message:

- the To: header line contains the email addresses of the primary recipients of the message 11 . Several addresses can be separated by using commas.
- the cc: header line is used by the sender to provide a list of email addresses that must receive a carbon copy of the message. Several addresses can be listed in this header line, separated by commas. All recipients of the email message receive the To: and cc: header lines.
- the bcc: header line is used by the sender to provide a list of comma separated email addresses that must receive a blind carbon copy of the message. The bcc: header line is not delivered to the recipients of the email message.

Initially, email was used to exchange small messages of ASCII text between computer scientists. However, with the growth of the Internet, supporting only ASCII text became a severe limitation for two reasons. First of all, non-English speakers wanted to write emails in their native language that often required more characters than those of the ASCII character table. Second, many users wanted to send other content than just ASCII text by email such as binary files, images or sound.

To solve this problem, the IETF developed the Multipurpose Internet Mail Extensions (MIME). These extensions were carefully designed to allow Internet email to carry non-ASCII characters and binary files without breaking the email servers that were deployed at that time. This requirement for backward compatibility forced the MIME designers to develop extensions to the existing email message format instead of defining a completely new format that would have been better suited to support the new types of emails.

4.3.2 Simple Mail Transfer Protocol SMTP

Now that we have explained the format of the email messages, we can discuss how these messages can be exchanged through the Internet. The figure 4.11 illustrates the protocols that are used when Alice sends an email message to Bob. Alice prepares her email with an email client or on a webmail interface.

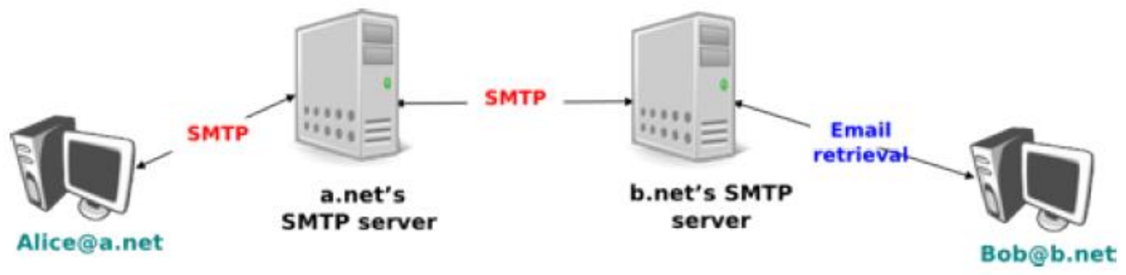

Figure 4.11: Email delivery protocols

To send her email to Bob, Alice‘s client will use the Simple Mail Transfer Protocol (SMTP) to deliver her message to her SMTP server. Alice‘s email client is configured with the name of the default SMTP server for her domain. There is usually at least one SMTP server per domain. To deliver the message, Alice‘s SMTP server must find the SMTP server that contains Bob‘s mailbox. This can be done by using the Mail eXchange (MX) records of the DNS. A set of MX records can be associated to each domain. Each MX record contains a numerical preference and the fully qualified domain name of a SMTP server that is able to deliver email messages destined to all valid email addresses of this domain. The DNS can return several MX records for a given domain. In this case, the server with the lowest preference is used first. If this server is not reachable, the second most preferred server is used etc. Bob‘s SMTP server will store the message sent by Alice until Bob retrieves it using a webmail interface or protocols such as the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP).

The Simple Mail Transfer Protocol (SMTP) defined in RFC 5321 is a client-server protocol. The SMTP specification distinguishes between five types of processes involved in the delivery of email messages. Email messages are composed on a Mail User Agent (MUA). The MUA is usually either an email client or a webmail. The MUA sends the email message to a Mail Submission Agent (MSA). The MSA processes the received email and forwards it to the Mail Transmission Agent (MTA). The MTA is responsible for the transmission of the email, directly or via intermediate MTAs to the MTA of the destination domain. This destination MTA will then forward the message to the Mail Delivery Agent (MDA) where it will be accessed by the recipient’s MUA. SMTP is used for the interactions between MUA and MSA 13, MSA-MTA and MTA-MTA.

SMTP is a text-based protocol like many other application-layer protocols on the Internet. It relies on the bytestream service. Servers listen on port 25. Clients send commands that are each composed of one line of ASCII text terminated by CR+LF. Servers reply by sending ASCII lines that contain a three digit numerical error/success code and optional comments.

The transfer of an email message is performed in three phases. During the first phase, the client opens a transport connection with the server. Once the connection has been established, the client and the server exchange greetings messages (EHLO command). Most servers insist on receiving valid greeting messages and some of them drop the underlying transport connection if they do not receive a valid greeting. Once the greetings have been exchanged, the email transfer phase can start. During this phase, the client transfers one or more email messages by indicating the email address of the sender (MAIL FROM: command), the email address of the recipient (RCPT TO: command) followed by the headers and the body of the email message (DATA command). Once the client has finished sending all its queued email messages to the SMTP server, it terminates the SMTP association (QUIT command).

A successful transfer of an email message is shown below:

```
S: 220 smtp.example.com ESMTP MTA information
C: EHLO mta.example.org
S: 250 Hello mta.example.org, glad to meet you
C: MAIL FROM:<alice@example.org>
S: 250 Ok
C: RCPT TO:<bob@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Alice Doe" <alice@example.org>
C: To: Bob Smith <bob@example.com>
C: Date: Mon, 9 Mar 2010 18:22:32 +0100
C: Subject: Hello
C:
C: Hello Bob
C: This is a small message containing 4 lines of text.
C: Best regards,
C: Alice
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
```

In the example above, the MTA running on mta.example.org opens a TCP connection to the SMTP server on host smtp.example.com. The lines prefixed with S: (resp. C:) are the responses sent by the server (resp. the commands sent by the client). The server sends its greetings as soon as the TCP connection has been established. The client then sends the EHLO command with its fully qualified domain name. The server replies with reply-code 250 and sends its greetings. The SMTP association can now be used to exchange an email.

To send an email, the client must first provide the address of the recipient with RCPT TO:. Then it uses the MAIL FROM: with the address of the sender. Both the recipient and the sender are accepted by the server. The client can now issue the DATA command to start the transfer of the email message. After having received the 354 reply code, the client sends the headers and the body of its email message. The client indicates the end of the message by sending a line containing only the . (dot) character. The server confirms that the email message has been queued for delivery or transmission with a reply code of 250. The client issues the QUIT command to close the session and the server confirms with reply-code 221, before closing the TCP connection.

4.3.3 The Post Office Protocol (POP)

When the first versions of SMTP were designed, the Internet was composed of minicomputers that were used by an entire university department or research lab. These minicomputers were used by many users at the same time. Email was mainly used to send messages from a user on a given host to another user on a remote host. At that time, SMTP was the only protocol involved in the delivery of the emails as all hosts attached to the network were running an SMTP server. On such hosts, an email destined to local users was delivered by placing the email in a special directory or file owned by the user. However, the introduction of personal computers in the 1980s, changed this environment. Initially, users of these personal computers used applications such as telnet to open a remote session on the local minicomputer to read their email. This was not user-friendly. A better solution appeared with the development of user friendly email client applications on personal computers. Several protocols were designed to allow these client applications to retrieve the email messages destined to a user from his/her server. Two of these protocols became popular and are still used today. The Post Office Protocol (POP) is the simplest one. It allows a client to download all the messages destined to a given user from his/her email server. We describe POP briefly in this section. The second protocol is the Internet Message Access Protocol (IMAP). IMAP is more powerful, but also more complex than POP. IMAP was designed to allow client applications to efficiently access in real-time to messages stored in various folders on servers. IMAP assumes that all the messages of a given user are stored on a server and provides the functions that are necessary to search, download, delete or filter messages.

POP is another example of a simple line-based protocol. POP runs above the bytestream service. A POP server usually listens to port 110. A POP session is composed of three parts: an authorisation phase during which the server verifies the client's credential, a transaction phase during which the client downloads messages and an update phase that concludes the session. The client sends commands and the server replies are prefixed by +OK to indicate a successful command or by -ERR to indicate errors.

When a client opens a transport connection with the POP server, the latter sends as banner an ASCII-line starting with +OK. The POP session is at that time in the authorisation phase. In this phase, the client can send its username (resp. password) with the USER (resp. PASS) command. The server replies with +OK if the username (resp. password) is valid and -ERR otherwise.

Once the username and password have been validated, the POP session enters in the transaction phase. In this phase, the client can issue several commands. The STAT command is used to retrieve the status of the server. Upon reception of this command, the server replies with a line that contains +OK followed by the number of messages in the mailbox and the total size of the mailbox in bytes. The RETR command, followed by a space and an integer, is used to retrieve the nth message of the mailbox. The DELE command is used to mark for deletion the nth message of the mailbox.

Once the client has retrieved and possibly deleted the emails contained in the mailbox, it must issue the QUIT command. This command terminates the POP session and allows the server to delete all the messages that have been marked for deletion by using the DELE command.

Below is presented a simple POP session. All lines prefixed with C: (resp. S:) are sent by the client (resp. server):

```
S: +OK POP3 server ready
C: USER alice
S: +OK
C PASS 12345pass
S: +OK alice’s maildrop has 2 messages (620 octets)
C: STAT
S: +OK 2 620
C: LIST
S: +OK 2 messages (620 octets)
S: 1 120
S: 2 500
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: QUIT
S: +OK POP3 server signing off (1 message left)
```

In this example, a POP client contacts a POP server on behalf of the user named “alice”. Note that in this example, Alice’s password is sent in clear by the client. This implies that if someone is able to capture the packets sent by Alice, he will know Alice’s password 16. Then Alice’s client issues the STAT command to know the number of messages that are stored in her mailbox. It then retrieves and deletes the first message of the mailbox.