Week 2: Application Layer

This week will clarify many interesting facts about services provided by the application layer. The application layer is the top most layer in the Internet protocol stack, which helps network applications resident in different computers to communicate. The application layer contains protocols that are used by network applications to facilitate mutual understanding. Among these protocols are: the HTTP, FTP, SMTP, and etc.

A distributed (also called **client/server)** application consists of two parts: one part runs at the client's (user's) site, another at the server's site. Both parts exchange messages during the process of interaction. There is no direct connection, however, between clients and servers. A client-side program (or better called a *process*) sends service requests destined to the server through a software gate, called a *socket*. The *socket* is the interface point between the user's application and the top most layer of the network system, the application layer. The application layer takes the responsibility of delivering the service request to the *server.* Delivering a service request is a complex job done with the aid of many layers. However, a user's process need not worry about all of this. The end result is a clean interaction between the server and client.

In client/server mode, applications intercommunicate with each other through message exchanges. To exchange messages, an application should provide the host and process addresses of the desired partner. For example, process A running at computer X that desires to send a message to process B running at computer Y, should provide the addresses for Y and B. Address of computer Y is known as Ys IP address and has a special format that we will discuss later on; process Bs address is given as a port number. Thus, if process B is a web server, then its address is port 80. Port numbers of popular applications are known to network programmers. Agents are programs that serve as an interface between users and network applications. For example: Netscape/Explorer web browser is an agent for the Web network application and Eudura/Netscape Messenger/Outlook is an agent for the e-mail application. In most applications, agents provide a convenient graphical program that helps users take advantage of network applications in a friendly manner.

A user's process can set parameters for the application layer to define the data loss tolerance, bandwidth, timing, and other necessary qualities. Figure 2.4 lists example applications and service requirements for each. For example, some applications, such as Internet TV, do not mind loosing small portions of transmitted data; it appears as a small glitch in the received picture. However other applications are sensitive to data loss, such as data in a financial transaction. Also, some applications demand a minimum bandwidth (amount of information sent per unit of time), such as Internet telephony. If this minimum is not provided, then the voice will not sound natural. Finally, some applications require receiving an answer within a preset time limit; a delayed result is a bad result. Accordingly, the application layers in both client and server interact through a set of common protocols to achieve the desired requirements.

The application layer does not carry the whole responsibility of achieving the communication task. In fact, the application layer does a good part of the communication process, and relies on other layers to do the rest of work. The application layer uses the TCP and UDP layer services to communicate with peer application layers, residing at remote sites. The choice between TCP and UDP depends on the desired quality of service. For example, TCP has tools to insure safe delivery (via retransmissions) and congestion control (through speed control), to insure complete delivery. UDP, on the other hand, serves those applications that tolerate a degree of data loss, but demand a constant data speed, such as streamed videos. Next seminar will delve into details of these transport layer protocols.

Next will have a look at a few of the most widely used applications and application layer protocols, namely: the HTTP, FTP, and SMTP.

The World Wide Web: HTTP

The Hypertext Transfer Protocol (HTTP) is an application layer protocol that facilitates understanding between web servers and browsers. A browser sends a request for a document to the server. The server receives the request in HTTP format; analyzes the lines of the HTTP message; prepares the document (sometimes by executing a script or a program); and sends the answer back in HTTP format. The browser gets back the server's response, then it displays the contents of the document according to the received HTTP format.

There are two main versions of HTTP protocols: HTTP/1.0 and the later HTTP/1.1. Both versions prepare an application's message and pass it, through a socket, to the TCP layer. A single HTTP transaction requires three exchanges between TCP peer layers. HTTP/1.0 sends *n* independent messages in a row to request *n* documents from a server. Meanwhile, HTTP/1.1 sends a single message to request *n* consecutive documents. Hence, TCP performs *3n* transactions to deliver the HTTP/1.0 document, and only *n+2* transactions to deliver the same amount of documents with HTTP/1.1.

HTTP uses TCP as an underlying transport protocol. Unlike UDP, TCP provides an error-free connection-oriented link between servers and clients. Client-side TCP does a three-way handshake with the server-side TCP to establish a connection, prior to any exchange of requests. This connection remains open until explicitly terminated or until timed-out after a lengthy lack of use. TCP also has measures for congestion control by controlling the speed of data transmission between applications. To simplify congestion control, TCP starts at a slow rate (*slow start*) then it keeps increasing the sending rate until it senses signs of imminent congestion, where it drops speed sharply and repeats the slow-start cycle again. By using TCP, HTTP needs to meet the additional overhead required by this reliable service, as we will see next.
HTTP uses two forms of connections to connect browsers and servers:

1. **nonpersistent connection** (used by HTTP 1.0). Under this connection a multi-object browser's request to the server results in multi-TCP connections. For example, if a browser requests a web page that has 10 images, each of the basic web page and the other 10 images would take a separate complete TCP connection that includes: browserside TCP connection request, server-side TCP connection acknowledgment, object transfer, and TCP connection termination. In addition, with every connection, the server needs to start TCP buffers and status variables, which is a great deal of work for a busy browser.
2. **persistent connection** (used by HTTP 1.1). This connection results in one connection for a multi-object browser request. Upon requesting a multi-object web page, a TCP connection is started and kept for the duration of transfer. Performance of persistent connections can be further improved by **pipelining** multiple requests. Instead of waiting for one request to be serviced, a browser sends back-to-back requests to the server. The server also sends back requested objects back-to-back.

HTTP messages are written in a readable ASCII format. Lines of the client message contain: the requested document, the host address, the connection type, the browser's version, the desired language, and entity body. The entity body of a client message is used only whenever a user fills in a form and sends it back to the server. An HTTP message generated by the server's site contains: a status line, header lines (containing date, server type, date of modification, length, and content's type), and the contents of the document. See specifics of HTTP formats in the text section 2.2.3.

Some documents require a password. Therefore, the server's first response to such requests is a message indicating to the browser the need of a password, with a 401 Authorization Required status code. The browser resends the message with the password obtained from the user. One might expect that an interceptor of these messages might be able to crack the password, an issue that will be dealt with later.

Cookies are labels attached to HTTP messages indicating to the server the identity of a certain frequent requester. The Web server uses this identity to act according to the requester's needs. Cookies are stored in the client's computer. Therefore, to get the benefit of cookies, the user must use the same computer.

Caching of Web pages is an important strategy to reduce the traffic in the network. A copy requested by a user is kept in the user's computer for probable future reuse. Whenever a user requests a Web page from a remote server, the local cache is checked first for the availability of this page. Organizations use **proxy servers** to represent member computers to the outside world. This proxy server acts as another cache for the whole organization. The more caches that exist on different levels the shorter the distance the Web page has to travel. An important cache issue is to keep cached copies up-to-date. Browsers may make a request to update a page based on examining the date of the cached page.

File Transfer: FTP

FTP is an old communication protocol that is still in use today. With FTP, you can download or upload files from and to a remote system. Like HTTP, FTP works on top of TCP to accomplish communication with remote hosts. However, unlike HTTP, the FTP server maintains **state** information about users. By maintaining a *state* about a user, the FTP server keeps track of all transactions done by a user during a session (an act HTTP does not do).

FTP commands are 4 ASCII characters each. FTP commands convey requests from clients to servers, such as requests for uploading, downloading, or directory changing. Servers respond with a three-digit status code indicating the server's response, such as information about successful data transfer, or error writing to a file, etc.

Electronic mail

Electronic mail has been around for a while. In recent years e-mail has become widely used by many people, including children. Nowadays, e-mail is being ported to cellular phones and small hand-held instruments.

An E-mail system is made up of three components: 1) user agent, to compose and read mail messages, 2) mail server, to queue mail from different users and feed them into the network, and 3) SMTP protocol that works in the application layer to establish communication with remote systems.

Although SMTP is a file transfer protocol, it has some differences with HTTP. First, all mail messages have to be converted into ASCII code, which is an inefficient way to send images or audio files. And, SMTP is a push protocol, where clients initiate file transfer.

Mail messages are formatted according to a known standard, RFC 822. The message format contains information about the sender, the recipient, and the subject. The MIME extension to the standard format contains information about non-ASCII messages and methods of decompression, or application programs to view delivered messages.

Mail access protocols, POP and IMAP are used to transfer mail messages from the mailboxes in the mail servers to users local PCs. POP downloads all messages received by the mail server and addressed to a user. The user can then organize his mail locally using his own PC. There is also a provision for users to keep a copy of each mail message on the mail server. On the other hand IMAP downloads only mail headers to the user PC; the user then selects certain messages to download. Selection between POP and IMAP depends on the user's method of operation. If a user accesses the mail using the same PC, then he is better off using POP, else (if the user is nomadic) IMAP is a better choice.

DNS- Domain Name Servers

Machines are identified by names that are easy to remember by humans. Internally, however, computers are identified with IP addresses that are made up of numbers. We have already studied the structure of an IP address with all its fields. It is easy to observe that the number of fields in the IP address is not equal to the number of fields of the host identification name.
DNS is an application layer protocol, which is used by another application layer's protocols, such as HTTP, FTP, and SMTP, to produce the hostname-to-IP address mapping.

Hostname-to-IP translation is done in a distributed manner. In other words, no single computer contains the whole database of IP addresses. We will demonstrate the work of DNS through an example. Consider trying to access, *surf.eurecom.fr,* whose IP address is 131.232.10.8. Observe that there is no one-to-one field translation between the hostname and the IP address. DNS works as follows: a local domain name server for the machine that searches for *surf.eurocom.fr* is consulted, hoping that it knows the translation. If the translation information resides in its cache the process is finished and the requesting machine get the IP address. If the request came from the same domain then we assume that the DNS must know about all computers in its domain, including *surf.* If the IP address is for another domain and is not cached at the DNS for *eurecom.fr,* one of the ROOT DNS is consulted with the same request. The ROOT DNS most probably does not know surf.eurecom.fr as it can't hold all of the names and their IP addresses, (a) because there are too many and (b) because a specific IP address can be assigned temporarily to a certain computer by a local service known as DHCP that we will meet in chapter four. The ROOT does not have a clue on the current mapping, but it does know who can answer this question. In a sense it says;" I do not know the answer, but I know someone that knows the answer". It will refer the inquiry to the local DNS for *eurecom.*fr who will provide the mapping that will be stored in the requesting domain DNS for future use, until the item in the cache (there is NO file on disk) is discarded to make place for a new mapping, because it was not used for a long period of time.

Clearly, if an IP address is known, it is faster to address a computer with that number. For example, use (telnet 131.232.10.250), instead of (telnet *urf.eurecom.fr).*
DNS also performs other services such as:

1. Host aliasing: complex (canonical) host names are replaced by simple aliases (For example, use of *enterprise.com* instead of *relay1.west-coast.enterprise.com)*
2. Mail server aliasing: similar to host aliasing, that is, using *hotmail.com* instead of *relay1.west-cost.hotmail.com*.
3. Load distribution: high-demand web servers, such as CNN, are replicated to cope with the large number of requests. Each replica has its own IP address, but still has the same name (cnn.com). When the IP address for this web server (cnn.com) is requested, the set of all IP addresses are sent, yet each time in a different order. DNS always selects the first IP number in the list. If this host is busy, then it goes for the second in the list and so on. Changing IP list order for replicated hosts directs subsequent requests to different hosts.

Content Distribution (Caching)

To reduce traffic and document retrieval time over the Internet, copies of big and frequently requested documents are replicated and stored in several hosts in the WWW. Now comes the question which host should be requested by a node for that document. To lead a node to the closest or the least loaded node, there are several setups.

1. Web caching: All requests to documents are made first to the **proxy server.** The proxy server, in addition to forwarding requests to proper IP addresses, keeps copies of requested pages upon arrival. These copies are used to satisfy near future requests by other nodes in the group. Thus a certain number of requests will result in hits for pages already stored (cached) in the proxy server. Caches can reside anywhere in the web: in

the institution or at the ISP, or anywhere else.
2. Content Distribution Networks (CDN) : Some content providers, such as Yahoo, buy caching services from new form of companies called CDN companies, such as Akami. Whenever a user requests a page from the content provider (Yahoo for example), the content provider responds with an HTTP document with URLs of a CDN distribution node. The browser requesting the [Yahoo let us say] document, then queries the CDN distribution node for that document. The CDN distribution node, in turn, directs the request to the most suitable CDN server. CDNs keep updating their caches with the most recent content provider documents.
3. Peer–to-Peer File sharing (P2P): User computers are used to store documents and share them with other peers whenever requested. This form of caching became widely used with applications such as Napster, Gnutella, Freenet, KaZaA. A user gets IP addresses of other peers storing files in one of several ways:
a. A centralized directory server keeps track of users that are active and of the files kept at (by) each user. All queries go the centralized directory, which provides a requester with an IP of a computer storing that document. If, let us say, the providing peer gets interrupted while in the middle of downloading a file, the centralized computer takes the responsibility of completing the file from other peers.
b. A decentralized directory: in this setup, every group of peers share a one centralized directory called group leader. Group leaders are connected to each other. In this way the directory is not stored in one place. The forming of groups and group leaders is done through a technique called overlay networks. A special sever, which is up all the time, serves to set up this virtual network by adding peers and group leaders on the run.
c. Query flooding: Also, a virtual network is formed of participating peers. However, no directories are present. A query is sent and forwarded by each node until a document is eventually found. In more details, a node sends a request to all its neighbors in the virtual network. Every neighbor who does not have the requested document forwards the request to all its neighbors, excluding the requesting node. Thus requests propagate in all directions. Special measures are utilized to restrict the queries from flooding the network.

Reading:

Read chapter 2 sections 2.1-2.6