Mr. Ali bin Tahir Email: ali@biit.edu.pk

Week # 15 – Lecture # 29 – Application Layer

Recommended Reading:

Book: Computer Networking, A Top-Down Approach 6th edition, Authors: James F. Kurose, K W. Ross

Application Layer

In last lecture, we have discussed web application and hyper text transfer protocol (HTTP). In this lecture we will discuss another related concept of web caching. After that, we will discuss more network applications such as file transfer and domain name system (DNS).

Here is the outline for today's lecture:

- **Web Caching (Proxy Server)**
- **File Transfer Protocol (FTP)**
- **FTP Commands**
- **Domain Name System (DNS)**

Web Caching (Proxy Server)

A Web cache—also called a proxy server—is a network entity that satisfies HTTP requests on the behalf of an origin Web server. The Web cache has its own disk storage and **keeps copies of recently requested objects** in this storage. As shown in Figure 2.11, a user's browser can be configured so that all of the user's HTTP requests are first directed to the Web cache. Once a browser is configured, each browser request for an object is first directed to the Web cache. As an example, suppose a browser is requesting the object **http://www.someschool.edu/campus.gif. Here is what happens:**

1. The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
2. The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.
3. If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server, that is, to www.someschool.edu. The Web

cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends the object within an HTTP response to the Web cache.

4. When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser (over the existing TCP connection between the client browser and the Web cache).

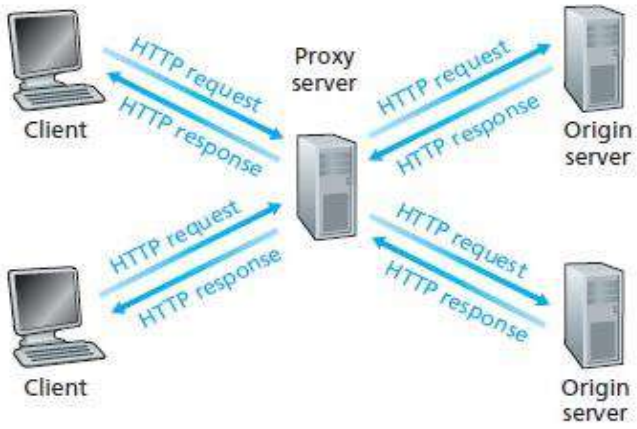

Figure 2.11 ♦ Clients requesting objects through a Web cache

Typically, a Web cache is purchased and installed by an institution or ISP to serve its customers. For example, a university might install a cache on its campus network and configure all of the campus browsers to point to the cache. **Note that a cache is both a server and a client at the same time**. When it receives requests from and sends responses to a browser, it is a server. When it sends requests to and receives responses from an origin server, it is a client.

Advantages of adding a Proxy Server

Web caching has seen deployment in the Internet for two reasons.

- First, a Web cache can substantially reduce the **response time** for a client request when required object is available on local proxy server (see figure 2.13).
- Second, Web caches can substantially **reduce traffic (cost)** on an institution's access link to the Internet. By reducing traffic, the institution (for example, a company or a university) does not have to upgrade bandwidth as quickly, thereby reducing costs. Furthermore, Web caches can substantially reduce Web traffic in the Internet as a whole, thereby improving performance for all applications.

- Third, Security.

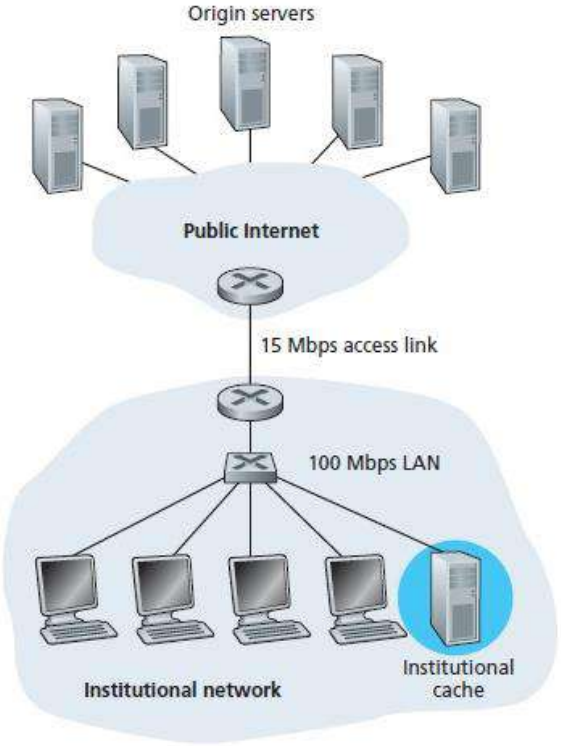

Figure 2.13 ♦ Adding a cache to the institutional network

File Transfer: FTP

In a typical FTP session, the user is sitting in front of one host (the local host) and wants to transfer files to or from a remote host. In order for the user to access the remote account, the user must provide a user identification and a password. After providing this authorization information, the user can transfer files from the local file system to the remote file system and vice versa.

As shown in Figure 2.14, the user interacts with FTP through an FTP user agent. The user first provides the hostname of the remote host, causing the FTP client process in the local host to establish a TCP connection with the FTP server process in the remote host. The user then provides the user identification and password, which are sent over the TCP connection as part of FTP commands. Once the server has authorized the user, the user copies one or more files stored in the local file system into the remote file system (or vice versa).

Port numbers **21** and **20** are **used for** FTP. **Port 21** is **used to** establish the connection between the 2 computers (or hosts) and **port 20** to transfer data (via the Data channel).

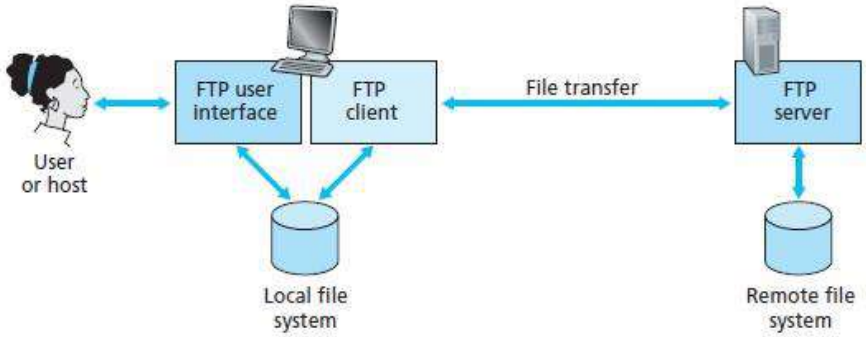

Figure 2.14 ♦ FTP moves files between local and remote file systems

HTTP vs. FTP

HTTP and FTP are both file transfer protocols and have many common characteristics; for example, they both run on top of TCP. However, the two application-layer protocols have some important differences. The most striking difference is that **FTP uses two parallel TCP connections to transfer a file, a control connection and a data connection**. The control connection is used for sending control information between the two hosts—information such as user identification, password, commands to change remote directory, and commands to "put" and "get" files. The data connection is used to actually send a file. Because FTP uses a separate control connection, FTP is said to send its control information out-of-band. **HTTP**, as you recall, **sends request and response header lines into the same TCP connection that carries the transferred file itself**. For this reason, HTTP is said to send its control information in-band The FTP control and data connections are illustrated in Figure 2.15.

Figure 2.15 ♦ Control and data connections

FTP Commands and Replies

Here we discuss some common FTP commands and replies. The commands, from client to server, and replies, from server to client, are sent across the control connection in ASCII format. Thus, like HTTP commands, FTP commands are readable by people. Each command consists of four uppercase ASCII characters, some with optional arguments. Some of the more common commands are given below:

Commands

- **USER username:** Used to send the user identification to the server.
- **PASS password:** Used to send the user password to the server.
- **LIST:** Used to ask the server to send back a list of all the files in the current remote directory. The list of files is sent over a (new and non-persistent) data connection rather than the control TCP connection.
- **RETR filename:** Used to retrieve (that is, get) a file from the current directory of the remote host. This command causes the remote host to initiate a data connection and to send the requested file over the data connection.
- **STOR filename:** Used to store (that is, put) a file into the current directory of the remote host.

Replies

FTP command sent across the TCP control connection. Each command is followed by a reply, sent from server to client. The **replies are three-digit numbers**, with an optional message following the number. This is similar in structure to the status code and phrase in the status line of the HTTP response message. Some typical replies, along with their possible messages, are as follows:

- **331 Username OK, password required**
- **125 Data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

DNS - The Internet's Directory Service

We human beings can be identified in many ways. For example, we can be identified by the names that appear on our birth certificates. We can be identified by our CNIC numbers. We can be identified by our driver's license numbers. Just as humans can be identified in many ways, so too can Internet hosts. One identifier for a host is its hostname. Host names—such as cnn.com, www.yahoo.com, gaia.cs.umass.edu, and cis.poly.edu—are common examples which are appreciated by humans. However, hostnames provide little, if any, information about the location within the Internet of the host. (A hostname such as www.eurecom.fr, which ends with the country code .fr, tells us that the host is probably in France, but doesn't say much more.) Furthermore, because hostnames can consist of variable length alphanumeric characters, they would be difficult to process by routers. For these reasons, hosts are also identified by so-called IP addresses.

An IP address consists of four bytes and has a rigid hierarchical structure. An IP address looks like 121.7.106.83, where each period separates one of the bytes expressed in decimal notation from 0 to 255. An IP address is hierarchical because as we scan the address from left to right, we obtain more and more specific information about where the host is located in the Internet (that is, within which network, in the network of networks). Similarly, when we scan a postal address from bottom to top, we obtain more and more specific information about where the addressee is located.

We have just seen that there are two ways to identify a host—by a hostname and by an IP address. People prefer to use hostname identifier, while routers prefer fixed-length, hierarchically structured IP addresses. In order to reconcile these preferences, we need a directory service that translates hostnames to IP addresses. This is the main task of the Internet's domain name system (DNS). The DNS is

- A distributed database implemented in a hierarchy of DNS servers, and
- An application-layer protocol that allows hosts to query the distributed database.

The DNS protocol runs over UDP and uses port 53. DNS is commonly used by other application-layer protocols—including HTTP, SMTP, and FTP—to translate user-supplied hostnames to IP addresses.

Example

As an example, consider what happens when a browser (that is, an HTTP client), running on some user's host, requests the URL www.someschool.edu/index.html. In order for the user's host to be able to send an HTTP request message to the Web server www.someschool.edu, the user's host must first obtain the IP address of www.someschool.edu. This is done as follows.

1. **The same user machine runs the client side of the DNS application.**
2. **The browser extracts the hostname, www.someschool.edu, from the URL and passes the hostname to the client side of the DNS application.**
3. **The DNS client sends a query containing the hostname to a DNS server.**
4. **The DNS client eventually receives a reply, which includes the IP address for the hostname.**
5. **Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address.**

Week # 15 – Lecture # 30 – Application Layer

Recommended Reading:

Book: Computer Networking, A Top-Down Approach 6th edition, Authors: James F. Kurose, K W. Ross

Application Layer

In previous lectures we have introduced DNS, this lecture is the continuation of the same topic and we will discuss more about the functionality of DNS and reinforce the concepts with the help of examples.

Here is the outline for today's lecture:

- **How DNS Works?**
- **A Distributed, Hierarchical Database**
- **DNS Example**
- **DNS Caching**

Overview: How DNS Works?

Suppose that some application (such as a Web browser or a mail reader) running in a user's host needs to translate a hostname to an IP address. The application will invoke the client side of DNS, specifying the hostname that needs to be translated. DNS in the user's host then takes over, sending a query message into the network. All DNS query and reply messages are sent within UDP datagrams to port 53. After a delay, ranging from milliseconds to seconds, DNS in the user's host receives a DNS reply message that provides the desired mapping. This mapping is then passed to the invoking application. Thus, from the perspective of the invoking application in the user's host, DNS is a black box providing a simple, straightforward translation service. But in fact, the black box that implements the service is complex, consisting of a large number of DNS servers distributed around the globe, as well as an application-layer protocol that specifies how the DNS servers and querying hosts communicate.

Centralized vs. Distributed DNS Server

A simple design for DNS would have one DNS server that contains all the mappings. In this centralized design, clients simply direct all queries to the single DNS server, and the DNS server responds directly to the querying clients. Although the simplicity

of this design is attractive, it is inappropriate for today's Internet, with its vast (and growing) number of hosts. The problems with a centralized design include:

- **A single point of failure**. If the DNS server crashes, so does the entire Internet!
- **Traffic volume**. A single DNS server would have to handle all DNS queries (for all the HTTP requests and e-mail messages generated from hundreds of millions of hosts).
- **Distant centralized database**. A single DNS server cannot be "close to" all the querying clients. If we put the single DNS server in New York City, then all queries from Australia must travel to the other side of the globe, perhaps over slow and congested links. This can lead to significant delays.
- **Maintenance**. The single DNS server would have to keep records for all Internet hosts. Not only would this centralized database be huge, but it would have to be updated frequently to account for every new host.

A Distributed, Hierarchical Database

In order to deal with the issue of scale, the DNS uses a large number of servers, organized in a hierarchical fashion and distributed around the world. No single DNS server has all of the mappings for all of the hosts in the Internet. Instead, the mappings are distributed across the DNS servers. To a first approximation, **there are three classes of DNS servers—root DNS servers, top-level domain (TLD) DNS servers, and authoritative DNS servers**—organized in a hierarchy as shown in Figure 2.19.

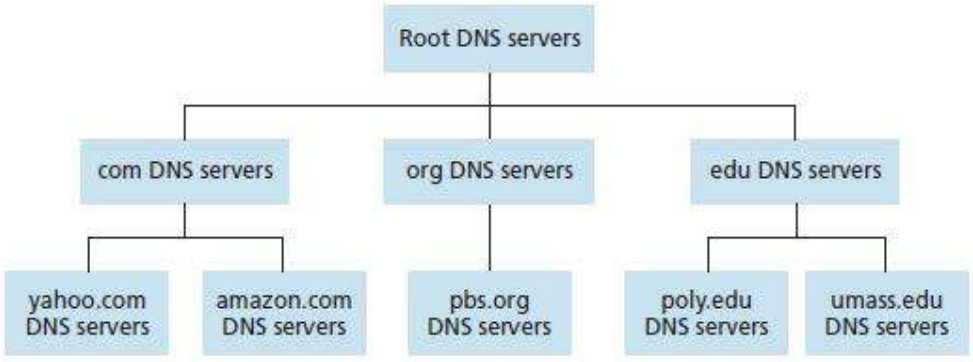

Figure 2.19 ♦ Portion of the hierarchy of DNS servers

To understand how these three classes of servers interact, suppose a DNS client wants to determine the IP address for the hostname www.amazon.com. To a first approximation, the following events will take place. The client first contacts one of the root servers, which returns IP addresses for TLD servers for the top-level domain com.

The client then contacts one of these TLD servers, which returns the IP address of an authoritative server for amazon.com. Finally, the client contacts one of the authoritative servers for amazon.com, which returns the IP address for the hostname www.amazon.com. We'll soon examine this DNS lookup process in more detail. But let's first take a closer look at these three classes of DNS servers:

Root DNS servers

Although we have referred root DNS server as a single server, but a single server is actually a network of replicated servers, for both security and reliability purposes. In the Internet there are multiple root DNS servers located in different parts of the world.

Top-level domain (TLD) servers

These servers are responsible for top-level domains such **as com, org, net, edu, and gov**, and all of the country top-level domains such as **uk, fr, ca, and pk.**

Authoritative DNS servers

Every organization with publicly accessible hosts (such as Web servers and mail servers) on the Internet must provide publicly accessible DNS records that map the names of those hosts to IP addresses. An organization's authoritative DNS server holds these DNS records. An organization can choose to implement its own authoritative DNS server to hold these records; alternatively, the organization can pay to have these records stored in an authoritative DNS server of some service provider. Most universities and large companies implement and maintain their own primary and secondary (backup) authoritative DNS server.

Local (default) DNS server

The root, TLD, and authoritative DNS servers all belong to the hierarchy of DNS servers, as shown in Figure 2.19. There is another important type of DNS server called the local DNS server. A local DNS server does not strictly belong to the hierarchy of servers but is nevertheless central to the DNS architecture. Each ISP—such as a university, an academic department, an employee's company, or a residential ISP—has a local DNS server (also called a default name server). The local DNS server may be on the same LAN as the host. When a host makes a DNS query, the query is sent to the local DNS server, which acts a proxy, forwarding the query into the DNS server hierarchy, as we'll discuss in more detail below.

DNS Example

Let's take a look at a simple example. Suppose the host cis.poly.edu desires the IP address of gaia.cs.umass.edu. Also suppose that Polytechnic's local DNS server is

called dns.poly.edu and that an authoritative DNS server for gaia.cs.umass.edu is called dns.umass.edu.

As shown in Figure 2.21, the host cis.poly.edu first sends a DNS query message to its local DNS server, dns.poly.edu. The query message contains the hostname to be translated, namely, gaia.cs.umass.edu. The local DNS server forwards the query message to a root DNS server. The root DNS server takes note of the edu suffix and returns to the local DNS server a list of IP addresses for TLD servers responsible for edu. The local DNS server then resends the query message to one of these TLD servers. The TLD server takes note of the umass.edu suffix and responds with the IP address of the authoritative DNS server for the University of Massachusetts, namely, dns.umass.edu. Finally, the local DNS server resends the query message directly to dns.umass.edu, which responds with the IP address of gaia.cs.umass.edu. Note that in this example, in order to obtain the mapping for one hostname, eight DNS messages were sent: four query messages and four reply messages! We'll soon see how DNS caching reduces this query traffic.

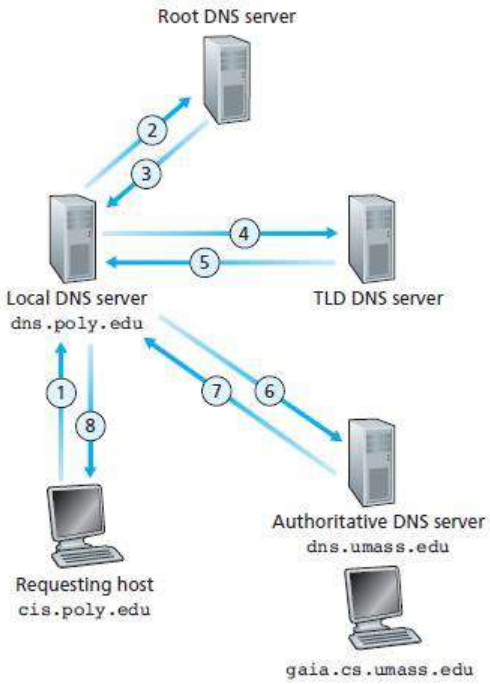

Figure 2.21 ♦ Interaction of the various DNS servers

Our previous example assumed that the TLD server knows the authoritative DNS server for the hostname. In general, this not always true. Instead, the TLD server may know only of an intermediate DNS server, which in turn knows the authoritative DNS server for the hostname. For example, suppose again that the University of Massachusetts has a DNS server for the university, called dns.umass.edu. Also

suppose that each of the departments at the University of Massachusetts has its own DNS server, and that each departmental DNS server is authoritative for all hosts in the department. In this case, when the intermediate DNS server, dns.umass.edu, receives a query for a host with a hostname ending with cs.umass.edu, it returns to dns.poly.edu the IP address of dns.cs.umass.edu, which is authoritative for all hostnames ending with cs.umass.edu. The local DNS server dns.poly.edu then sends the query to the authoritative DNS server, which returns the desired mapping to the local DNS server, which in turn returns the mapping to the requesting host. In this case, a total of 10 DNS messages are sent!

Iterative vs. recursive query

The example shown in Figure 2.21 makes use of both recursive queries and iterative queries. **The query sent from cis.poly.edu to dns.poly.edu is a recursive query, since the query asks dns.poly.edu to obtain the mapping on its behalf. But the subsequent three queries are iterative since all of the replies are directly returned to dns.poly.edu.** In practice, the queries typically follow the pattern in Figure 2.21: The query from the requesting host to the local DNS server is recursive, and the remaining queries are iterative.

In theory, any DNS query can be iterative or recursive. For example, Figure 2.22 shows a DNS query chain for which all of the queries are recursive.

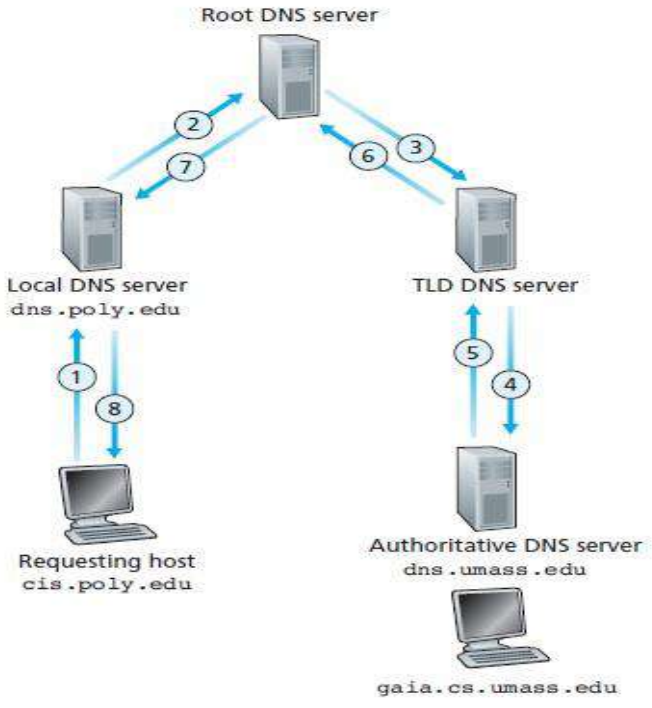

Figure 2.22 ♦ Recursive queries in DNS

DNS Caching

DNS caching is an important feature of the DNS system. In truth, DNS extensively exploits DNS caching in order to improve the delay performance and to reduce the number of DNS messages spreading around the Internet. The idea behind DNS caching is very simple. In a query chain, when a DNS server receives a DNS reply (containing, for example, a mapping from a hostname to an IP address), it can cache the mapping in its local memory.

For example, in Figure 2.21, each time the local DNS server dns.poly.edu receives a reply from some DNS server, it can cache any of the information contained in the reply. If a hostname/IP address pair is cached in a DNS server and another query arrives to the DNS server for the same hostname, the DNS server can provide the desired IP address, even if it is not authoritative for the hostname.

Since hosts and mappings between hostnames and IP addresses are by no means permanent, DNS servers discard cached information after a period of time (often set to two days).

Example

As an example, suppose that a host apricot.poly.edu queries dns.poly.edu for the IP address for the hostname cnn.com. Furthermore, suppose that a few hours later, another Polytechnic University host, say, kiwi.poly.edu, also queries dns.poly.edu with the same hostname. Because of caching, the local DNS server will be able to immediately return the IP address of cnn.com to this second requesting host without having to query any other DNS servers. A local DNS server can also cache the IP addresses of TLD servers, thereby allowing the local DNS server to bypass the root DNS servers in a query chain (this often happens).