Computer Networks (UE18CS301)
Unit 2

Aronya Baksy

August 2020

1 Application Layer

1.1 Principles of Network Applications

- Network Applications are programs that run on different host systems and communicate with each other along a network.
- Network applications are written at an abstract level, without the hardware details of link-layer devices (switches) or network-layer devices (routers).
- These low level details are presented to the application layer only as an abstraction.

1.2 Network Application Architectures

1.2.1 Client-Server Architecture

- There is an always-on host machine called the **server**, which services requests from resources from many machines. A machine that makes a request is called a **client**.
- Clients do not directly communicate with each other. They always communicate via the server.
- Since the server has a fixed address called the IP Address, and it is always on, the server is always reachable by any client.
- In order to avoid a single server getting overwhelmed by large volume of client requests,**data centers** containing large number of servers aggregated together are used by large content providers.
- Such companies must pay bandwidth and interconnection costs, apart from maintenance and power costs for upkeep of these data centers.

1.2.2 Peer-to-Peer Architecture (P2P)

- In a P2P architecture, communication takes place between pairs of intermittently connected hosts (aka **peers**).
- These peer machines are not large servers but common desktop/laptop machines owned by daily users.
- P2P architectures have the advantage of self scalability as peers are capable of requesting and adding services to the network at the same time.
- P2P architecture has the following disadvantages:
  - Not ISP friendly as most ISPs are optimized for more volume of downstream traffic than upstream (aka asymmetric usage). But P2P applications shift a large volume of upstream traffic from servers to the local ISPs.
  - Not secure as they are open and highly distributed.
  - Not enough incentive for users to donate their own bandwidth and power to offer service to other peers.

1.3 Process Communication

- Processes are labelled as client and server based on the one who initiated the communication session (client), and the one who is waiting to be contacted to begin the session (server).
- The **socket** is the interface between an application running on the application layer, and the rest of the TCP/IP layers below it. A socket is a combination of the IP Address and the Port Number.
- The socket is a software interface that acts like a gateway for processes to communicate over a network.

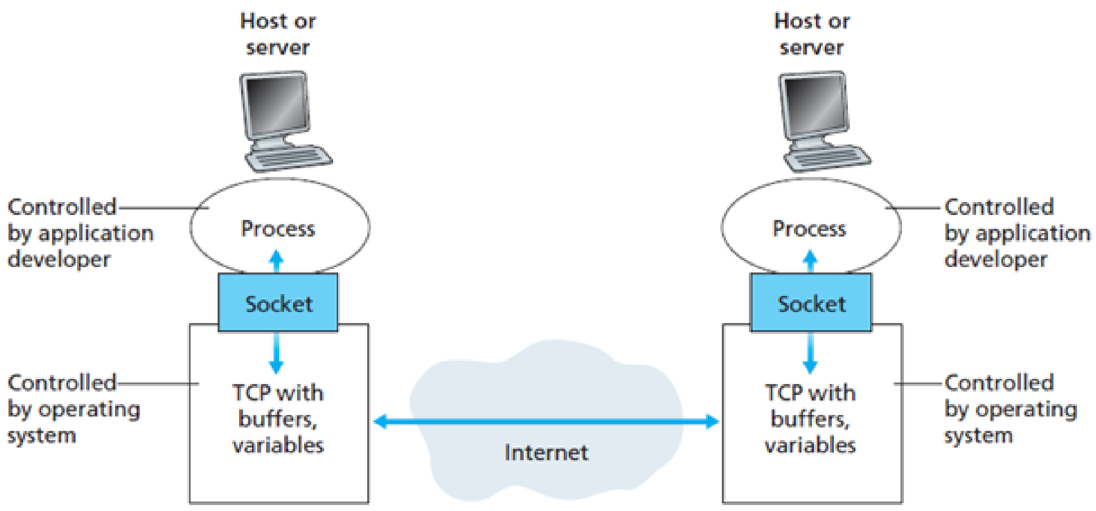
Host or server
Host or server
Controlled by application developer
Process
Socket
Process
Socket
Controlled by application developer
Controlled by operating system
TCP with buffers, variables
Internet
TCP with buffers, variables
Controlled by operating system

Figure 1: Socket illustration

- The IP address of a host machine identifies it uniquely in a network.
- The **port number** identifies a particular application or process within a host machine.
- The port number is used to distinguish between multiple different network applications running on a single host machine.
- Some common port numbers are: 80 for HTTP, 443 for HTTPS, 20 and 21 for FTP, and either 25, 465/587 or 2525 for SMTP.
- The IP Address is a 32 bit integer, while the port number is a 16 bit integer value (port numbers range from 0 to 65535).
- Port numbers 0 to 1023 are known as *well-known ports* allocated to server services by the IANA, ports 1024 to 49151 are *registered ports* (semi reserved, services) and ports 49151 onwards are used by *client* programs.

2 Transport Layer Services

The following are some of the services that are available to the application layer from the transport layer

- **Reliable Data Transfer**: Guarantee that no data will be lost or will arrive out of sequence at the receiver from the sender. Some applications do not require this (eg: audio/video streaming)
- **Throughput**: Guarantee of a fixed throughput across a link, important for some throughput sensitive applications (eg: internet telephony).
- **Timing**: Guarantee of delay between server and receiver being less than or equal to some fixed value, important for real-time applications like telephony, online gaming etc.
- **Security**: Services like encryption allow for confidentiality between sender and receiver, along with other security services like data integrity checking and end-point authentication.

2.1 Transmission Control Protocol (TCP)

- TCP is a **connection-oriented** protocol. This means that before any actual data is sent between 2 host machines, they must go through a **handshaking** process first.
- The objective of this handshaking is to set up the validity of the connection, and it consists of 3 packets sent: client sends SYN to server, server sends SYN/ACK to client, and client responds with ACK.
- TCP also implements **reliable data transfer**. TCP can be relied on to transmit data from one host to another in correct order with zero loss.
- TCP also includes additional services for **congestion control** and **flow control** to manage TCP traffic along busy networks.

2.2 User Datagram Protocol (UDP)

- UDP is a connection-less protocol, meaning that there is no initial set up of a connection between two communicating processes before the actual data transfer
- UDP does not provide reliable data transfer, and it also does not provide services like congestion control.
- UDP offers the advantage of greater speed than TCP, which is useful for real-time network applications (eg: video streaming, online conferencing etc.).

3 Application Layer Protocols

The following are defined as part of any application layer protocol.

- Types of messages to be exchanged (eg: request and response)
- Message syntax (eg: fields and how fields are delineated)
- Message semantics (meaning of all the fields in the message)
- Rules to determine when and how messages are sent and responded to.

Application layer protocols such as HTTP, SMTP, FTP and proprietary protocols like Skype etc. are one important component of all network applications.

4 Hypertext Transfer Protocol (HTTP)

- HTTP is the application-layer protocol that is at the heart of the world wide web.
- A web page consists of different **objects** (HTML files, images, documents, Java applets etc.). Every web page on the internet has a **base HTML** file and the associated resources with it.
- Every resource has its own unique Uniform Resource Locator (URL).
- The client requests for HTTP objects, receives them from server and processes them (displaying or otherwise).
- The server listens for requests and responds to them by sending the appropriate message.
- The HTTP protocol is defined in RFC 1945 (version 1.0) and RFC 2616 (version 1.1).

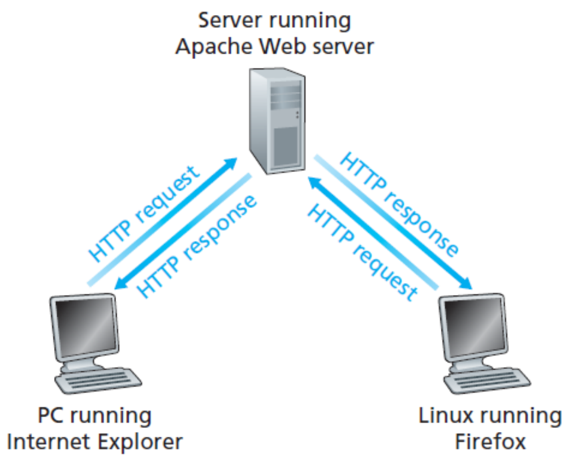

Figure 2: Request-Response Behaviour of HTTP

4.1 Overview of HTTP

- HTTP uses TCP as its underlying transport layer protocol.
- First, the client initiates a TCP connection with the server. Once the connection is established, client and server communicate via their respective socket interfaces.
- The client sends requests and receives responses through its socket interface, while the server receives requests and sends responses through its socket interface.
- Reliable data transfer and the transport layer services are maintained by the TCP protocol on the transport layer, and are not of concern to HTTP.
- HTTP is a **stateless** protocol, in that a server does not maintain any information about any past transactions with a client.
- Whenever a client requests for a resource from a server that has been previously requested for the same resource by the same client, a **new connection** will be opened that does not contain any information about past connections.
- The advantage of a stateless protocol are as follows:
  - If either client or server fails, then the state that each machine sees may differ, hence causing conflict when the connection is to be resumed.
  - No need for dynamic memory allocation when connection is starting and ongoing, and no need for any cleanup when a connection terminates/fails.

4.2 Persistent and Non-Persistent HTTP

- **Non persistent connections** describe the situation where each request-response pair between 2 hosts takes place over a separate TCP connection
- Only one object can be sent along a non-persistent connection before it is closed.
- The steps in establishing a non-persistent connection are:
  - Client initiates a connection on Port 80 to the server. The server starts up a socket associated with this connection
  - Client sends HTTP request to server containing the path of the requested resource within the server
  - The server receives the request, locates the resource requested, encapsulates it within the response message and sends it.

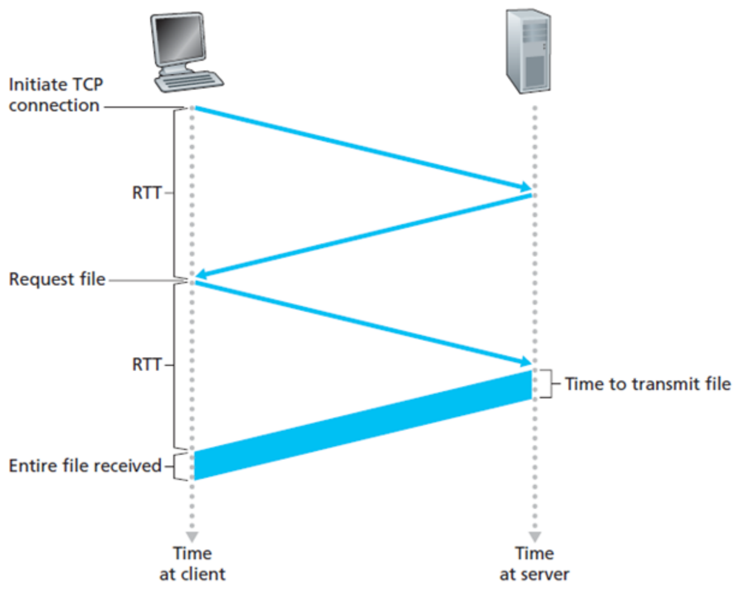

Figure 3: Non-persistent HTTP connection

  - Client receives the response and closes the TCP connection. Client processes the response and handles the object that was sent back.

- The entire response time for a single object over a Non-Persistent connection is

$$HTTP\ Response\ Time = 2 \times RTT + Obj\ Transfer\ Time$$

- Non persistent connections suffer from increased overhead on the server end to open and close multiple TCP connections, and increased time for object transmission (2 RTTs).
- In a **persistent connection**, the connection is kept open even after the server sends a response
- All subsequent communication between server and client can take place over the same connection.
- These requests for objects can be made back-to-back, without waiting for replies to pending requests (pipelining).
- When the server receives the back-to-back requests, it sends the objects back-to-back.
- Response time is cut in half in persistent connection compared to non-persistent.

4.3 HTTP Message Format

4.3.1 HTTP Request Format

- The first line of HTTP request is called the request line. It is followed by the header lines that contain information about the request, and the entity body.
- The request line contains information about the type of request (GET, POST, PUT, HEAD or DELETE), the location of the requested resource on the server, and the version of HTTP used (1.0 or 1.1). It is used by Web caches.
- The `Connection` header field defines whether the connection is persistent (`Keep-Alive`) or non-persistent (`close`).
- The entity body is empty for GET requests, but contains information when used with POST method.

- The POST method is used to securely fill forms without exposing user information. The HEAD method is used when the client only wants the HTTP header in response and not any actual content. The PUT and DELETE methods are used to store and remove files to/from a certain directory in the web server.

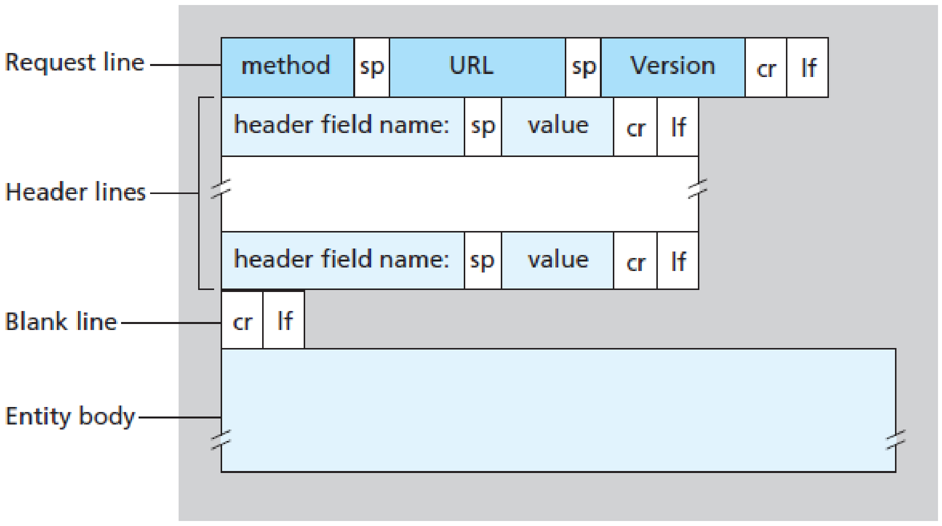

Figure 4: HTTP Request Message Format

4.3.2 HTTP Response Format

- The HTTP response consists of a status line, followed by header lines, then a blank line and finally the resource content (the entity body).
- The status line contains the HTTP version (1.0 or 1.1), the status code (a 3 digit number) and the English phrase associated with that code.
- Some important status codes are `200 OK`, `301 Moved Permanently`, `304 Not Modified`, `400 Bad Request`, `404 Not Found` and `501 HTTP Version Not Supported`.

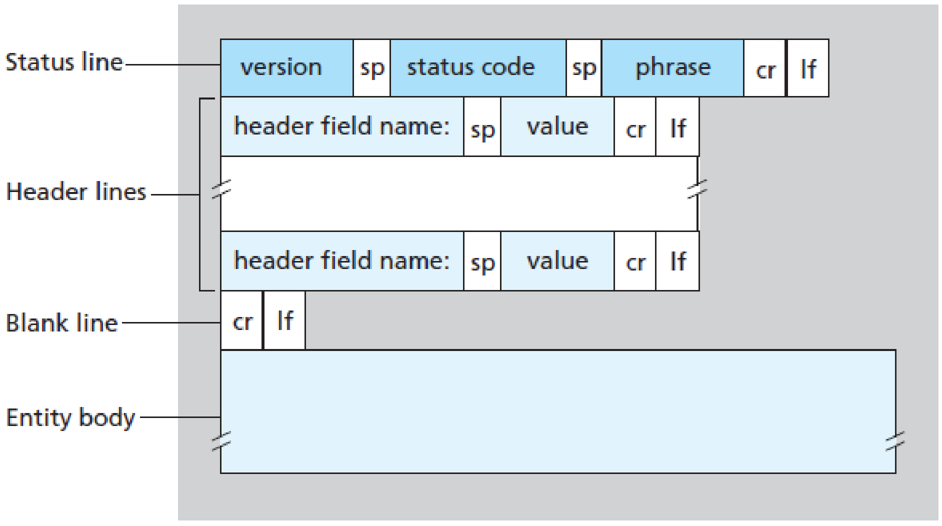

Figure 5: HTTP Response Message Format

4.4 HTTPS

- HTTPS stands for HTTP Secure or HTTP over Secure Socket Layer (SSL). It integrates HTTP with an encryption standard that is bidirectional (client to server and server to client are both encrypted communications).
- HTTPS uses port number 443 as its standard port.
- TLS (Transport Layer Security) is the most recent encryption standard that is used by HTTPS (because SSL is now deprecated). The most recent version is TLS 1.3 (RFC 8446).
- All TLS connections have the following properties:
  1. Connection is **secure** as the key is based on a *shared secret* that is negotiated between client and server before any data is transmitted, at the start of the session. This negotiation is secure (the key cannot be found even by an attacker in the middle of the connection) and reliable (no attacker can modify this communication without being detected).
  2. The identity of the two parties can be **authenticated** using public key cryptography. This authentication can be made optional, but is generally required for at least one party (server)
  3. Connection is **reliable** as message integrity is checked using a *message authentication code* to prevent loss or alteration of data.
- TLS and SSL rely on **certificates** issued by trusted third parties that certifies the ownership of a public key by the subject of the certificate.
- The following steps are involved in a HTTPS communication:
  1. Browser requests secure pages from server
  2. Server sends its public key with its SSL certificate (digitally signed by a third party – CA)
  3. On receipt of certificate, browser verifies issuer's digital signature. (This is commonly denoted by green padlock key symbol in the browser).
  4. Browser creates a symmetric key (shared key), keeps one and gives a copy to server. This symmetric shared key is encrypted using server's public key before sending.
  5. On receipt of encrypted secret key, server decrypts it using its private key and gets browser's secret symmetric key.
- Asymmetric encryption is used to verify identity of the owner and its public key, hence establishing trust between client and server.
- Symmetric encryption is used for actual data transfer.

4.5 Cookies

- Even though HTTP is a stateless protocol, cookies are one method for servers to maintain state information about clients.
- Cookie technology has 4 components: HTTP Request header, HTTP Response header, the cookie file kept on client end system and managed by the user's browser, and the back end database at the server end.
- Cookie information is exchanged over request and response headers.
- The identification number for a user can be linked to a database on the server side that contains all the information of that user. This can be used to quickly offer personal experience to the user
- This is used by e-commerce websites like Amazon and eBay to deliver their shopping cart service and personalized recommendations.

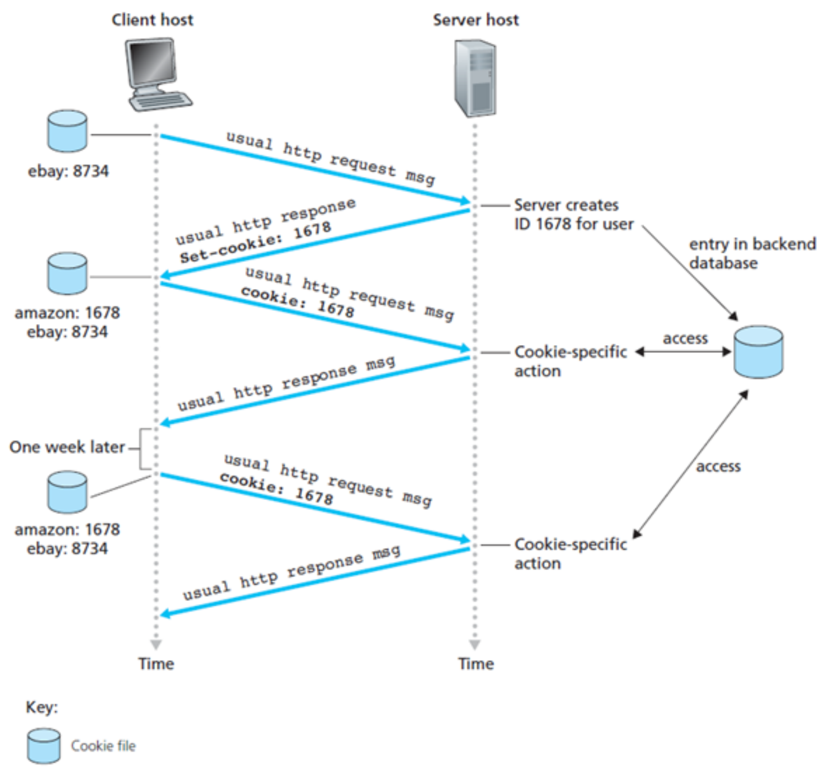

Figure 6: Maintaining user state with cookies

4.6 Web Caching

Web cache is a server in between client and server that services requests for popular resources much faster than the actual server.

- The client opens TCP connection to the web cache and sends HTTP request to it.
- Cache checks if a copy of the resource is stored, If yes then it is sent on the response
- If not, the cache sends an HTTP request to the origin server for the resource, receives a response, and stores a copy of the object from the origin server in its storage. Then that object is sent back to the client.
- Cache acts as server for the client side, and as a client to the origin server. They are installed by the ISP.
- Cache reduces the traffic intensity on access link between institutional network and public network, as well as reducing average response times for HTTP requests.

5 Domain Name System (DNS)

- The DNS offers the service of translating human-readable hostnames to IP Addresses that can be used to locate hosts.
- DNS is an application-layer protocol that relies on a **distributed network** of **DNS Servers** to provide this service.

5.1 DNS Services

- **Host Aliasing**: Mapping shorter hostnames like `enterprise.com` and `www.enterprise.com` to larger actual hostnames like `relay1.west-coast.enterprise.com` (aka the *canonical hostname*).

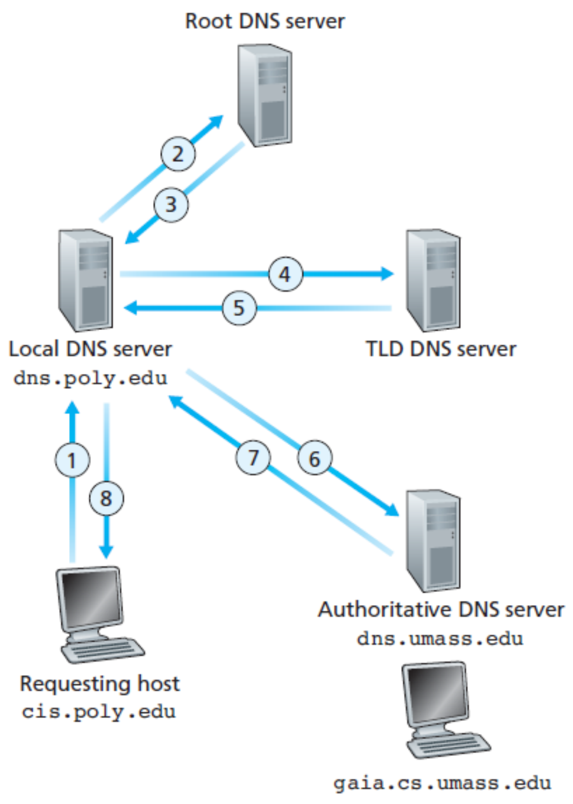

Figure 7: Working of DNS

- **Mail Server Aliasing**: Translation between canonical mail server host names and user level mnemonic mail server host names. The MX record in the DNS header allows the mail server and web server to share the same hostname.
- **Load Distribution**: If a busy website has multiple servers, and the DNS is aware of the IP Addresses of all these servers, then requests to that website are rotated between all the servers to ensure uniform distribution between all the servers.
  This is done by the DNS changing the order of the table of IP Addresses that are sent to the client.

5.2 Working of DNS

- DNS consists of the following hierarchy of distributed servers: (top to bottom)
  - **Root DNS Servers**: There are 13 IP Addresses associated with root level server networks. Each IP Address leads to a network of replicated root level DNS Servers.
  - **Top Level Domain (TLD) Servers**: The DNS servers for top level domains like com, edu as well as country level TLDs like in, pk, fr, us, jp etc.
  - **Authoritative DNS Servers**: Every pyblicly accessible host provides a public DNS record on the authoritative servers. Large institutions/companies maintain their own Authoritative servers, and smaller ones rent this service from a provider.
- The **Local DNS server** does not belong to this hierarchy but is very important. It acts as a web cache or a proxy server to the client machines.
- The local DNS server is installed by an ISP, and when a host connects to this ISP's network, it is provided with a list of IP Addresses of the local DNS servers (done using DHCP).
- The local DNS server forwards a query from host to the main DNS hierarchy.

- Queries to DNS servers can be iterative or recursive.
- The query from client to the Local DNS server is a recursive query as the query asks the local DNS server to obtain the mapping on its behalf.
- The other queries are all iterative as the local DNS directly receives the reply for its requests.
- DNS queries can be entirely recursive or entirely iterative, but in the real world the pattern below is most common.

5.3 DNS Caching

- Whenever a DNS server receives a response to a query, it can store this response in its own memory.
- The next time a request is received for this particular mapping, it can respond with the IP Address despite not being an authoritative DNS server for that host.
- The cache is cleared every 2 days as mappings between hostnames and IP Addresses are not permanent.

5.4 DNS Records and Messages

- A DNS record consists of a 4-tuple `(Name, Value, Type, TTL)`
  - If `Type=A`, then `Name` is the hostname and `Value` is the IP Address for that hostname.
  - If `Type=NS`, then `Name` is a domain and `Value` is the hostname of an authoritative server that knows the IP Addresses of all the hosts in that domain.
  - If `Type=CNAME`, then `Name` is an alias hostname and `Value` is the canonical hostname.
  - If `Type=MX`, then `Name` is an alias hostname and `Value` is the canonical hostname of a mail server.
- The `CNAME` and `MX` types allow mail and web servers to have the same alias hostname. If mail server's canonical hostname is required, then the MX record is queried, and CNAME is queried if web server's canonical hostname is required.

5.4.1 DNS Message Format

- The first 12 bytes (96 bits) makes up the header section
- The header section consists of a 16-bit query ID, 16-bit flags field, and the number of question, answer, authority and additional records in the message.
- Some flags are:
  - *Query(0)/reply(1) flag*, indicates whether message is query or reply
  - *Authoritative flag* is set in response if the server sending the response is authoritative
  - *Recursion desired*, and *recursion supported* flags
- The question section contains query information, with the name field and the type of query.
- Answer section in a reply message consists of the resource records pulled from the server. This can have more than 1 record as one hostname can have multiple IP Addresses (replicated servers).
- Authority section contains info about other authoritative servers
- The additional section contains other helpful records. For example, the answer field in a reply to an MX query contains a resource record providing the canonical hostname of a mail server. The additional section contains a Type A record providing the IP address for the canonical hostname of the mail server.

6 Peer to Peer Applications

6.1 BitTorrent Protocol

- The collection of all peers participating in file distribution is called the *torrent.* Peers in a torrent download **chunks** of a file from each other (typical chunk size is 256 kB).
- Peers may leave once the file is downloaded entirely (selfish) or continue to supply chunks to other peers in the torrent (altruistic).
- In the network of peers, one node is designated as a **tracker**. Tracker tracks all the nodes that are in the torrent. Before joining the torrent a peer must inform the tracker of this.
- When a new peer joins, the tracker randomly selects 50 nodes in the torrent and sends their IP Addresses to the new node. The new node establishes concurrent TCP connections with all these nodes. The nodes with which TCP connection is successfully established by the new node are called the **neighbouring peers**.
- Periodically, a peer asks its neighbouring peers for a list of chunks they have. Given these lists, the peer can ask its neighbours for the chunks it does not have using the TCP connection.
- The **rarest first** protocol is used to determine the order of chunk gathering. In this protocol, the least frequently stored chunk is requested first by the peer. Over time, this equalizes the number of copies of each chunk stored accross the torrent.
- When responding to requests for chunks, a peer will give priority to the 4 peers that send it chunks at the **maximum rate**. The peer sends these 4 replies in return. These 4 peers are said to be **unchoked**.
- The top 4 peers are updated every 10 seconds by checking the transfer rates.
- In addition to the 4 unchoked peers, one additional neighbour is randomly chosen and the peer sends it chunks. This is called the **optimistically unchoked** peer. A new randomly chosen peer is optimistically unchoked every 30 seconds.
- If the two peers are satisfied with the trading, they will put each other in their top four lists and continue trading with each other until one of the peers finds a better partner.
- The effect is that peers capable of uploading at compatible rates tend to find each other.
- The random neighbor selection also allows new peers to get chunks, so that they can have something to trade.

6.2 Distribution Time Calculation

The distribution time is defined as the length of time taken to transfer a copy of the file to all the machines on the network.

6.2.1 Client-Server Architecture

- Number of clients in the network is N and file size is F bits.
- The upload rate of server's access link is u_s, the upload rate of the client access links is u_i for the i^{th} client, and the download rate of the client access links is d_i for the i^{th} client.
- Initially only the server has the file. The time taken to upload N copies of the file is $\frac{NF}{u_s}$
- The time taken for the clients to receive the file is limited by the one with the smallest download rate d_i. Let $d_{min} = min(d_1, d_2, .., d_N)$. Then the time taken to download the file is $\frac{F}{d_{min}}$.
- The distribution time can be calculated as

$$d_{CS} = max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\} \tag{1}$$

6.2.2 Peer to Peer Architecture

- The server must send one copy of the file into the access network. This is done in time $\frac{F}{u_s}$.
- The peer with the lowest download rate will get a copy of the file from the network in time $\frac{F}{d_{min}}$.
- The total upload capacity of the system $u_{total} = u_s + u_1 + u_2 + .. + u_n$. The system must deliver (upload) F bits to each of the N peers, thus delivering a total of NF bits. This cannot be done at a rate faster than u_{total}. Hence the distribution time is also at least $\frac{NF}{u_{total}}$
- Hence the distribution time is given by

$$d_{P2P} = max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{total}}\right\} \tag{2}$$

7 Other Application Layer Protocols

7.1 File Transfer Protocol (FTP)

- Uses TCP to exchange large files over internet. FTP uses two parallel TCP connections for a single file exchange, called the **data connection** (port 20) and the **control connection**(port 21).
- File manipulation (rename, move, put, get) as well as identification (user ID, password) information is sent over the control connection while the actual file contents are exchanged over the data connection.

7.2 Simple Mail Transfer Protocol (SMTP)

- It is an application layer protocol or electronic mail transfer. It uses TCP port 25.
- All SMTP communication are secured using SSL or TLS.
- SMTP servers run on a centralized mail server (like gmail.com or yahoo.com). The clients open TCP connections to these mail servers.
- Message from client enters client's mailbox located in client server. From the client mailbox, it enters the central message queue of the client mail server. From here, the message is sent by TCP to the destination mail server's message queue.
- From the destination message queue, the message is routed to the destination user's individual mailbox. The destination user can now access their mailbox at their convenience.

7.3 Dynamic Host Configuration Protocol (DHCP)

- It is an application layer service that is used to assign IP Addresses to hosts on a network dynamically.
- For server it runs on port 67, on client it runs on port 68.
- DHCP uses a client-server model, and it runs on a sequence of discovery, offer, request and ACK.
- Aside from the IP Address, DHCP assigns the subnet mask, the local DNS server, and the default gateway.

7.4 Simple Network Management Protocol (SNMP)

- Exchange of management information between network devices is handled by SNMP protocol
- SNMP runs on ports 161 and 162. It uses both TCP and UDP.
- SNMP consists of the following components:
  - *SNMP Manager*: Oversees all managed devices.
  - *Managed Devices*: Host machines, routers, link-layer switches that run agent software.
  - *SNMP Agents*: Software installed in all managed devices.
  - *Management Info Base*: Store details of managed devices. One per device.

7.5 Remote Access: Telnet and SSH

- Telnet and SSH are remote access protocols that allow, for example, network admins to remotely monitor the state of devices that are connected to a network.
- Telnet uses port 23 of TCP, while SSH uses port 22. Telnet uses TCP exclusively while SSH uses both TCP and UDP.
- Telnet transfers all data as plain text, while SSH transfers data in encrypted form using *public key* encryption.
- Telnet is susceptible to middle man attacks where hackers listen to the telnet connection and get the user's authentication information