**SERVICES OF DNS :**

There are two ways to identify a host--by a hostname and by an IP address. People prefer the more mnemonic 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 (1) a distributed database implemented in a hierarchy of DNS servers and (2) an application-layer protocol that allows hosts to query the distributed database. The DNS servers are often UNIX machines running the Berkeley Internet Name Domain (BIND) software. The DNS protocol runs over UDP and uses port 53.

DNS is commonly employed by other application-layer protocols--including HTTP, SMTP, and FTP--to translate user-supplied host names to IP addresses. IN order for the user's host to be able to send an HTTP request message to the Web server the user's host must first obtain the IP address.

* **Host aliasing**: A host with a complicated hostname can have one or more alias names.
* **Mail server aliasing**: For obvious reasons, it is highly desirable that e-mail addresses be mnemonic.
* **Load distribution**: DNS is also used to perform load distribution among replicated servers, such as replicated Web servers.

The DNS is dated in several additional RFC's. It is a complex system, and we only touch upon key aspects of its operation here

**2.5.2 Overview of How DNS Works:**

The hostname that needs to be translated is the function call that an application call in order to perform the translation. 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.

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 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.
* **Distant Centralized database**: A single DNS server can’t be "close to" all the querying clients.
* **Maintenance**: The single DNS server would have to keep records for all Internet hosts.

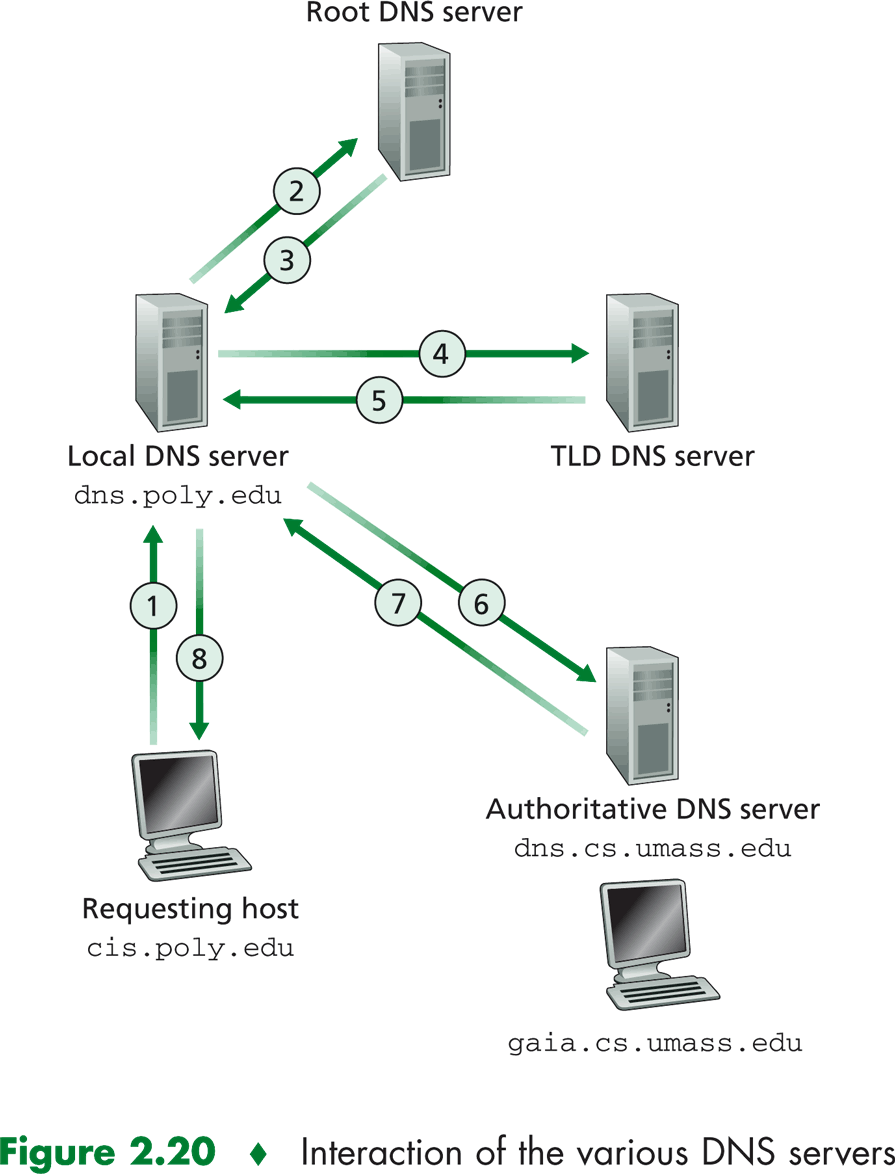
In summary, a centralized database in a single DNS server simply doesn't scale.

**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 mapping for all of the host in the Internet. Instead, the mappings are distributed across--root DNS servers. To first approximate, there are three classes of DNS servers--root DNS servers, top-level domain (TLD) DNS a servers, and authoritative DNS servers--organized in a hierarchy. To understand how these three classes of servers interact, suppose a DNS client wants to determine the IP address for the hostname the following events will take place. The client first contacts one of the roots 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. Finally, the client contacts on of the authoritative servers for which returns the IP address for the hostname. The three classes of DNS servers are:

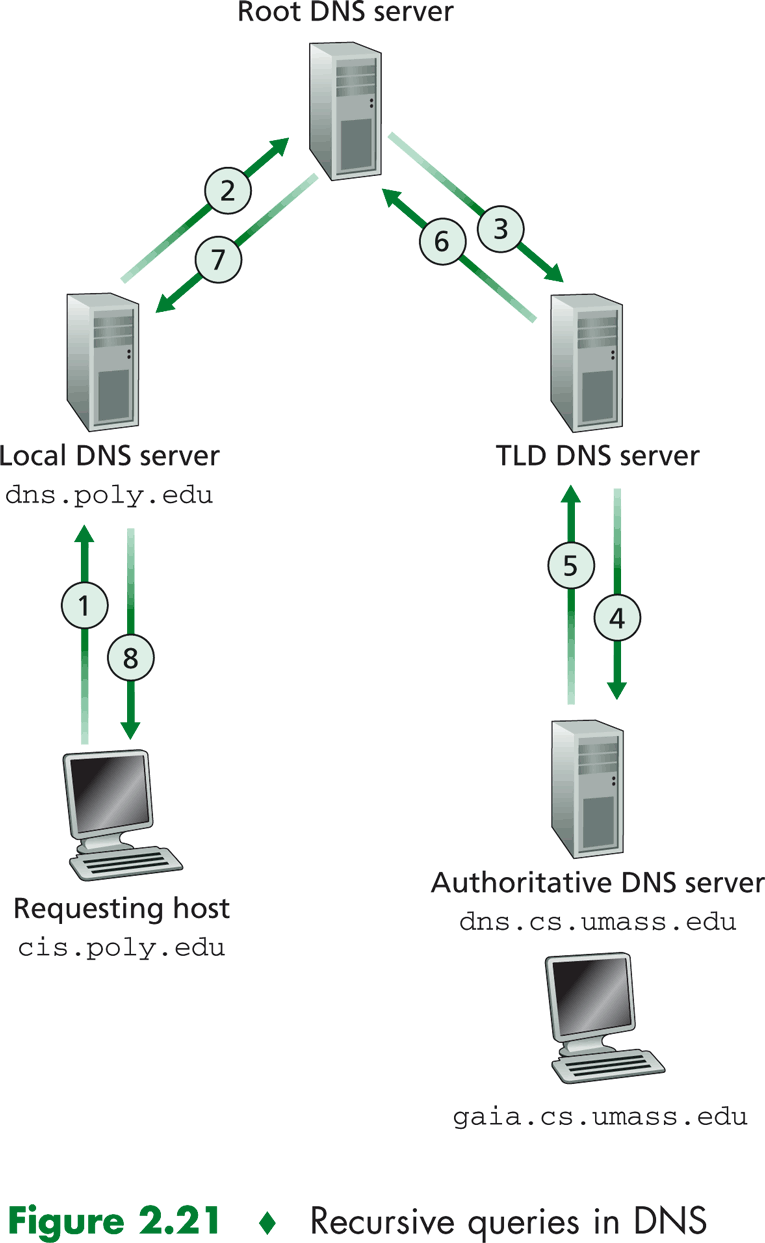
* Root DNS server. In the Internet there are 13 root DNS servers, most of which are located in North America.
* Top-Level Domain (TLD) servers. These servers are responsible for top-level domains such as com, org, net edu, and gov, and all of the country top-level domains such as uk, fr, ca, and jp.
* Authoritative DNS servers. Every organization with publically accessible hosts on the Internet must porvide publicly accessible DNS records that map the names of those hosts to IP addresses

The root, TLD, and authoritative DNS servers all belong to the hierarchy of DNS servers. There is another important type of DNS, 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 has a local DNS server. When a host connects to an ISP, the ISP provides the host with the IP addresses of one or more of its local DNS servers. You can easily determine the IP address of your local DNS server by accessing network status windows in Windows or UNIX. A host's local DNS server is typically "close to" the host. For an institutional ISP, the local DNS server may be on the same LAN as the host; for a residential ISP, it is typically separated from the host by no more than a few routers. 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.

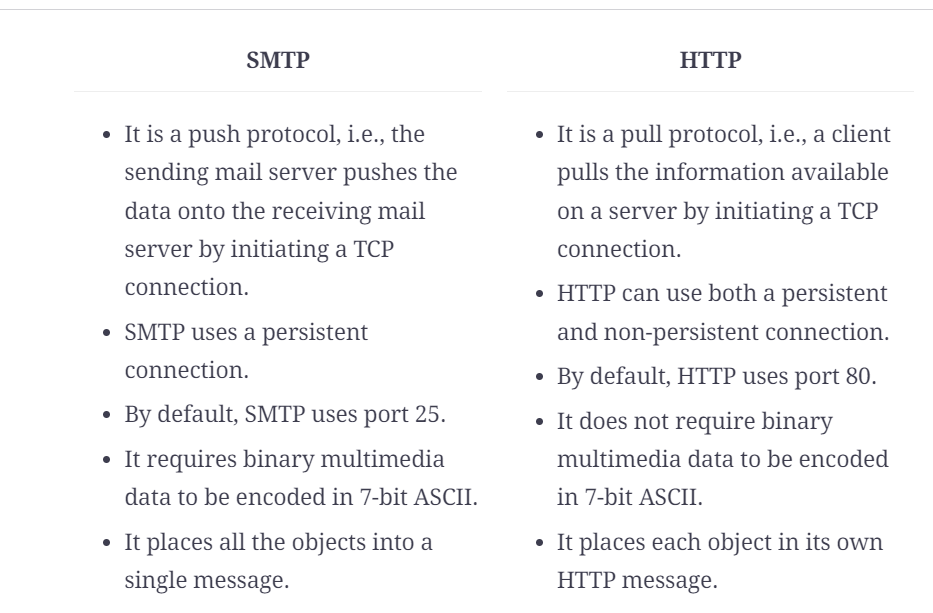


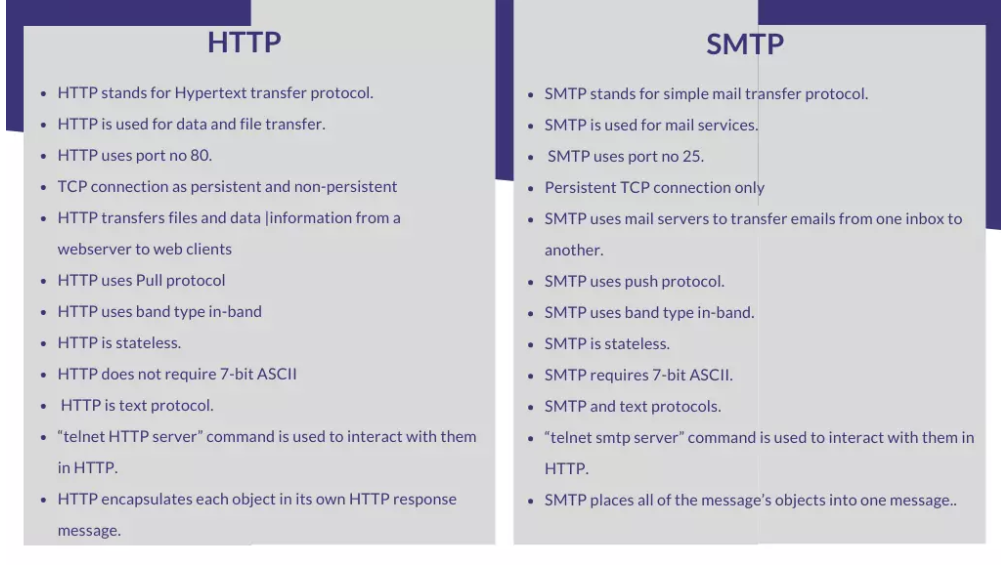
**DNS Caching:** DNS caching, a critically important feature of the DNS system. In truth, DNS extensively exploits DNS caching in order to improve the delay performance and to reduce the number of DNS messages ricocheting around the Internet. The idea behind DNS caching is very simple. In a query chain, when a DNS server receives a DNS reply it can cache the information in the reply in its local memory.

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. Because hosts and mappings between hostnames and IP addresses are by no means permanent, DNS servers discard cached information after a period of time.



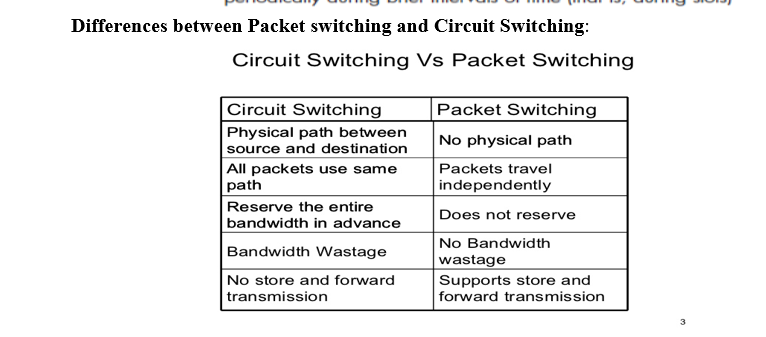
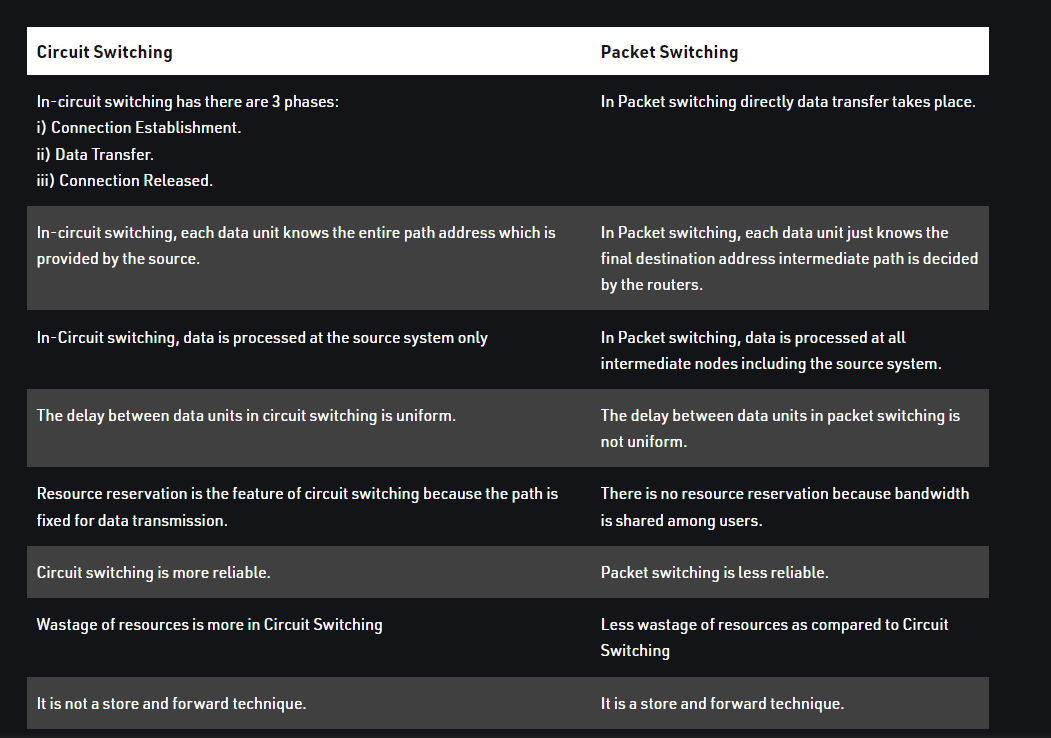
**SMTP VS HTTP**

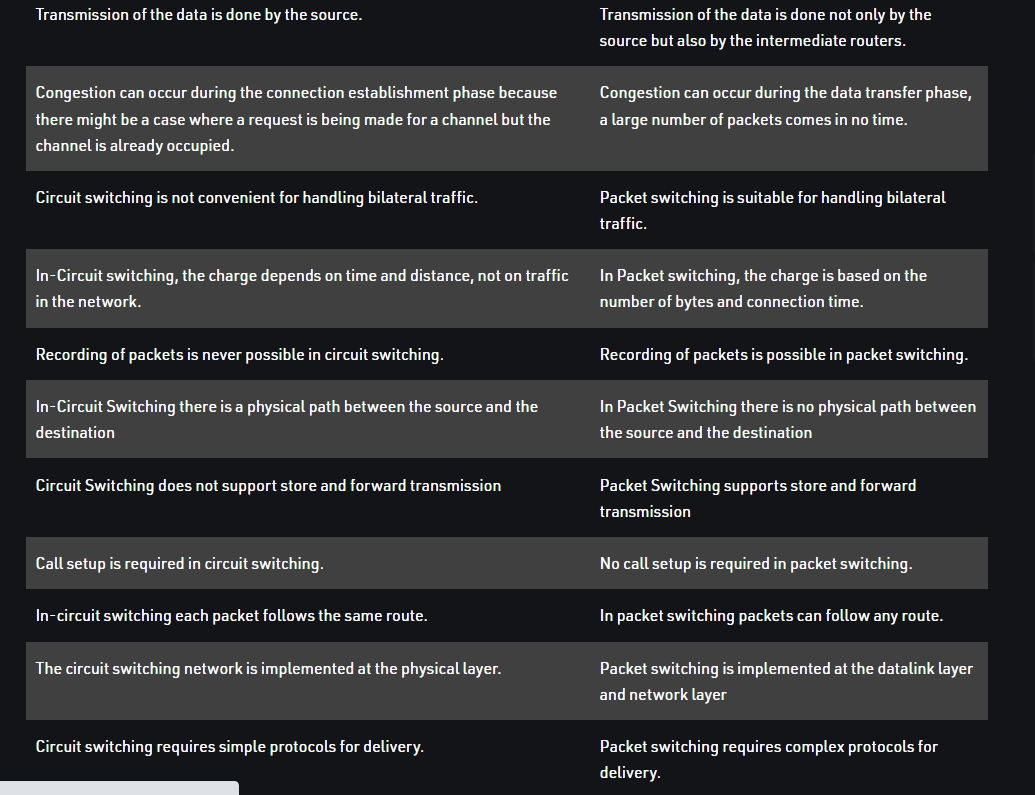




**Differentiate circuit switching and packet switching**

**Unit 1 question**



**UDP SEGMENT :**

**UDP:** UDP, Aside from the multiplexing/demultiplexing function and some light error checking, it adds nothing to IP. In fact, if the application developer chooses UDP instead of TCP, then the application is almost directly talking with IP. UDP takes messages from the application process, attaches source and destination port number fields for the multiplexing/demultiplexing service, adds two other small fields, and passes the resulting segment to the network layer. The network layer encapsulates the segment into an IP datagram and then makes a best-effort attempt to deliver the segment to the receiving host. If the segment arrives at the receiving host, UDP uses the port numbers and the IP destination address to deliver the segment’s data to the correct application process. Note that with UDP there is no handshaking between sending and receiving transport-layer entities before sending a segment. For this reason, UDP is said to be connectionless

DNS is an example of an application-layer protocol that uses UDP. When the DNS application in a host wants to make a query, it constructs a DNS query message and passes the message to a UDP socket without performing any handshaking; UDP adds header fields to the message and passes the resulting segment to the network layer. The network layer encapsulates the UDP segment into a datagram and sends the datagram to a name server. The DNS application at the querying host then waits for a reply to its query. If it doesn’t receive a reply it either tries sending the query to another nameserver, or it informs the invoking application that it can’t get a reply.

**3.3.1 UDP Segment Structure:** In the UDP segment structure, the application data occupies the data field of the UDP datagram. For example, for DNS, the data field contains either a query message or a response message. For a streaming audio application, audio samples fill the data field. The UDP header has only four fields, each consisting of two bytes. The port numbers allow the destination host to pass the application data to the correct process running on the destination (that is, the demultiplexing function). The checksum is used by the receiving host to check if errors have been introduced into the segment. In truth, the checksum is also calculated over a few of the fields in the IP header in addition to the UDP segment. The length field specifies the length of the UDP segment, including the header, in bytes



UDP Header FORMATE –

UDP header is an 8-bytes fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consist of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.

