2. APPLICATION LAYER

M T W T F S S
Page No.:
Date:
YOUVA

2.1 PRINCIPLES OF NETWORK APPLICATION

Network Application Architecture.

1. Client - Server Architecture

→ There is an always-on host called the server which services requests from various hosts known as clients.

2. Peer to Peer Architecture

→ There are no dedicated servers or data centres. Instead direct communication between hosts called peers

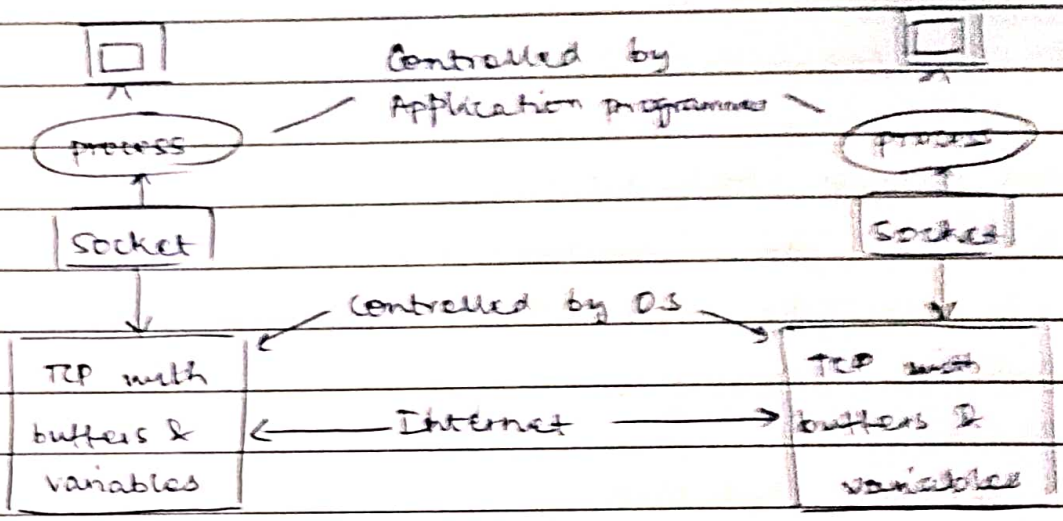
* P2P are cheaper than client server and self scalable
* CSA is more secure and stable. P2P may suffer due to assymmetric upstream and downstream traffic

PROCESS COMMUNICATING

→ Processes on 2 different ~~de~~ systems communicate with each other by exchanging messages across the network A sending process sends messages into the network and the recieving process recieves the message and possibly responds by sending a message back. Processes communicating with each other reside in the application layer.

→ In a client-server model, the process ~~process~~ that initiates the communication is the client while the one that waits for to be contacted is the server.

SOCKETS



→ A process can send and recieve messages ~~from~~ the network through the socket.

* Each process must have a unique identifier called its I.P ADDRESS

Each host has its own I.P and multiple processes may be running each on a separate port

Services Available to Applications

1. Reliable Data Transfer → Sending process should be able to pass data into the socket with complete confidence that the data will be recieved without any errors

2. Throughput → Transport layer protocol should guarentee throughput at a specific rate

3. Timing → It should also guarentee that each bit should arrive no more than x seconds late

4. Security → Should encrypt data, Provide confidentiality

APPLICATION LAYER PROTOCOLS

It defines the following

1. The types of messages exchanged
2. The syntax of various message types
3. The semantics of fields ie. thier meaning
4. Rules for determining when and how a process sends and responds to messages.

2.2 WEB AND HTTP

→ HTTP is the web's application-layer protocol. It is implemented in two programs → server and client

→ HTTP defines how web clients request web pages from web servers and how they are transferred back to clients

→ When a user requests a page, the browsers sends an http get request and when the server receives it, it responds with Http response messages ~~to~~ that contain the objects.

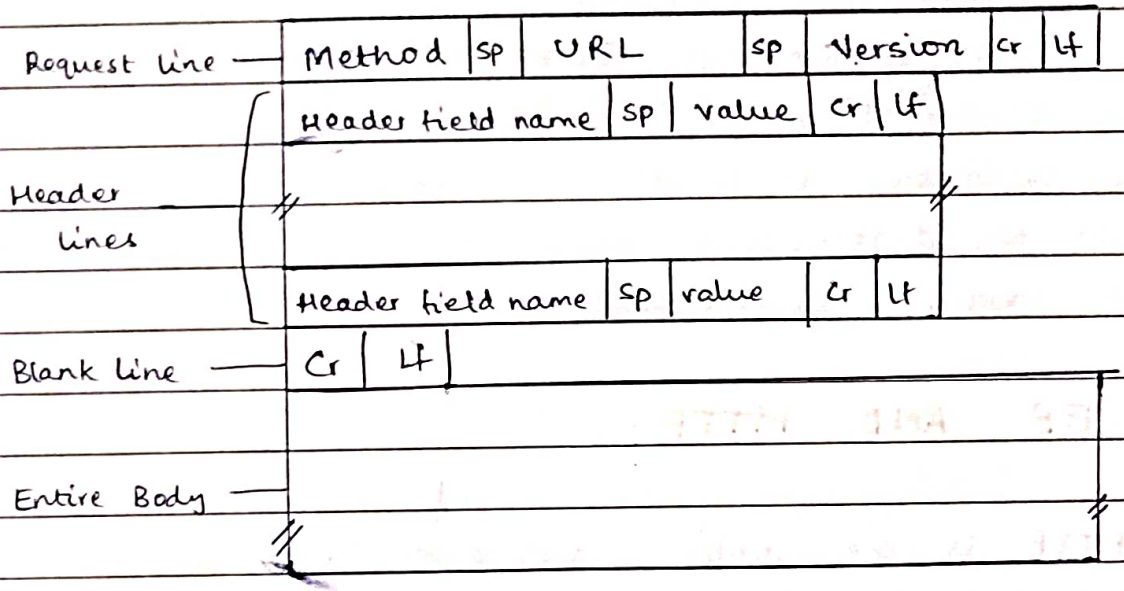
→ HTTP uses TCP as its underlying transport protocol

→ HTTP maintains no information about its past clients and is hence called stateless protocol

HTTP Connections

Persistent	Non-Persistent
Multiple objects can be sent	Single object is sent
Server is left open	Requires 2 RTT's per object
One RTT for all objects	Overhead for each TCP conn.

HTTP Request Message



HTTP Request messages

1. POST Method → User input sent from client to server in body of request message

2. GET Method → Include user data in URL field

3. HEAD Method → Only headers

4. PUT Method → Upload new file to server.

HTTP Response Codes

200 (OK) → Request Successful
301 (Moved) → Request object moved permanently
400 (Bad Request) → Message not understood
404 (Not found)
505 → HTTP version not supported

2.2 USER SERVER INTERACTION

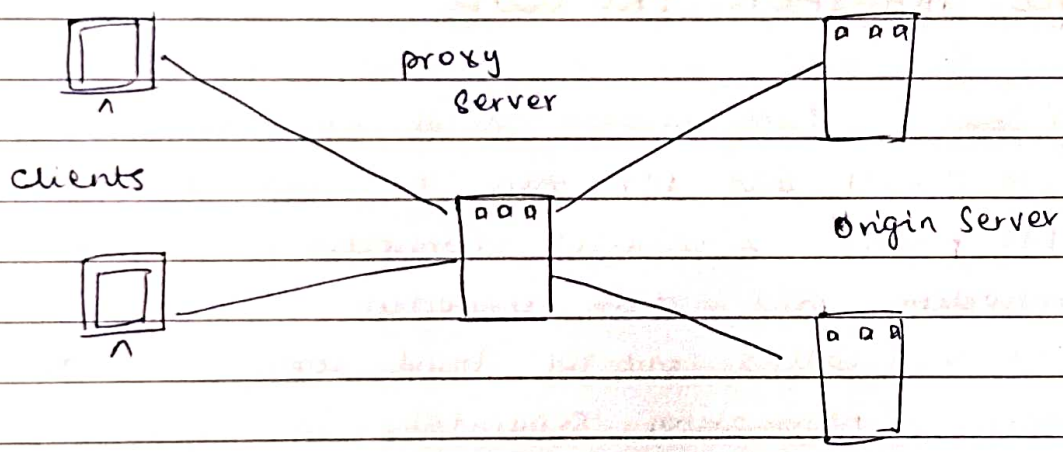
Cookies

→ Since HTTP is stateless, it cannot keep track and identify users. For this purpose, cookies are used

→ Cookies can be used for user identification. During subsequent sessions, the browser passes a cookie header thereby identifying the user to the server

→ Although useful, can cause privacy issues.

WEB CACHING



→ A web-cache / proxy server is a network entity that satisfies HTTP requests on behalf of the origin web servers

→ A web cache acts as a server as well as a client

→ Once a client / browser sends a request to the cache, it returns the object back to the client or else requests the object from the origin server

→ Reduces response time and traffic

11 CONDITIONAL GET

→ Web caching reduces response time but introduces a new problem of stale pages residing in the proxy servers

→ HTTP has a mechanism that allows cache to verify that the objects are up-to-date using the mechanism of Conditional Get.

→ The HTTP request header contains an If-Modified-Since header line

→ Goal is to send a new object only if browser has outdated page object

2.3 FILE TRANSFER PROTOCOL

→ FTP moves files between local and remote file systems

→ Both HTTP and FTP run on top of TCP.

→ FTP uses 2 parallel connections ie. the control connection and data connection

→ FTP is called out of band connection as it sends control information separately.

→ Throughout the session FTP maintains the state about the user

FTP Commands & Replies

2.3 SMTP

→ SMTP transfers messages from senders mail servers to the receipients

→ SMTP first establishes a connection between client and server. After connection is established, the server and client perform application layer handshaking

→ the client sends a message after this and this process is repeated if other messages are to be sent or else the TCP connection is closed.

→ SMTP is a primarily PUSH protocol used for sending files to the server.

→ SMTP uses a persistent connection and can send multiple objects in a multipart message.

→ A different protocol such as IMAP/POP must be used for retrieval of message from server

2.3 D.N.S

→ DNS provides a hostname to IP address translation

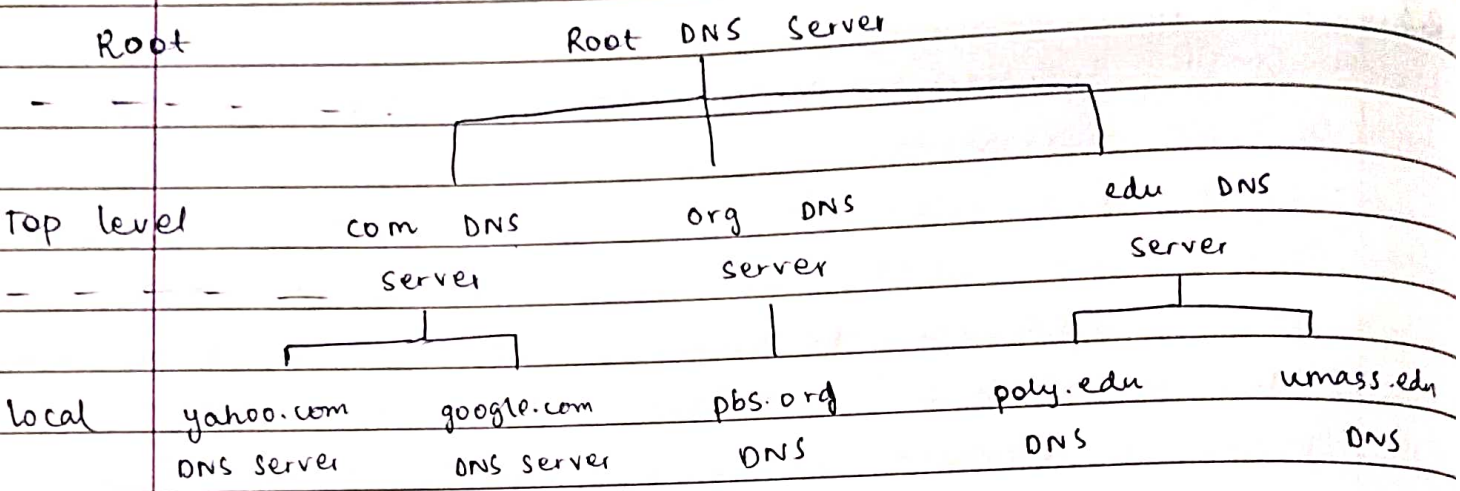
→ Provides Host Aliasing: Cannonical name (More complex) along with aliases for hostnames

→ Mail server aliasing:

→ load distribution among replicated servers.

* DNS does not have a centralized design as it can lead to single point of failure. Also it will not be able to handle high traffic and doesn't scale

→ DNS can be implemented in a hierarchical / distributed manner.



→ When a host makes a DNS query, it is first sent to the local DNS server
→ Local DNS replies from its cache

* If name-to-address translation not found, either iterative or recursive query takes place
Recursive queries may increase load and hence a mix of both is used in practice

DNS CACHING

→ Once a server recieves a DNS reply in its query chain, it can cache the mapping in its local memory.
→ As these mappings are not permanent, they will be discarded after some time

DNS MESSAGES

→ DNS queries and replies have the following same format

Identification Flags Flags	12 bytes
No of questions Number of answers RRs	
No of Authority RRs No of additional RRs	
Questions	Name, type fields for query
Answers	RR's in resp to query
Authority	Records for auth servers
Additional Info	Additional helpful info

Flags contain info about → query or reply
→ Recursion desired
→ Recursion available
→ Authoritive Reply

DNS SECURITY

1. DDOS Attack → Attacker sends large number of packets resulting in legitimate DNS queries not being answered. More dangerous way is to bombard the TLD's. TLDs are not easily bypassed as root servers

2. Man in the Middle → Spoofing and intercepting queries from hosts and returning bogus replies