Chapter 2 : Application layer

Internet protocol stack: application ⇒ IMAP, SMTP, HTTP FTP
↓
Transport ⇒ TCP, UDP
↓
Network ⇒ IP, routing protocols
↓
link ⇒ Ethernet, WiFi, PPP
↓
physical ⇒ wire

- server - client paradigm :
  - server ⇒ always on, fixed IP
  - client ⇒ intermittently connected, dynamic IP
  - ex: HTTP, IMAP, FTP

- peer - peer architecture
  - no always on server
  - end systems communicate directly
  - peers request & provide services from each other
  - ex: p2p file sharing

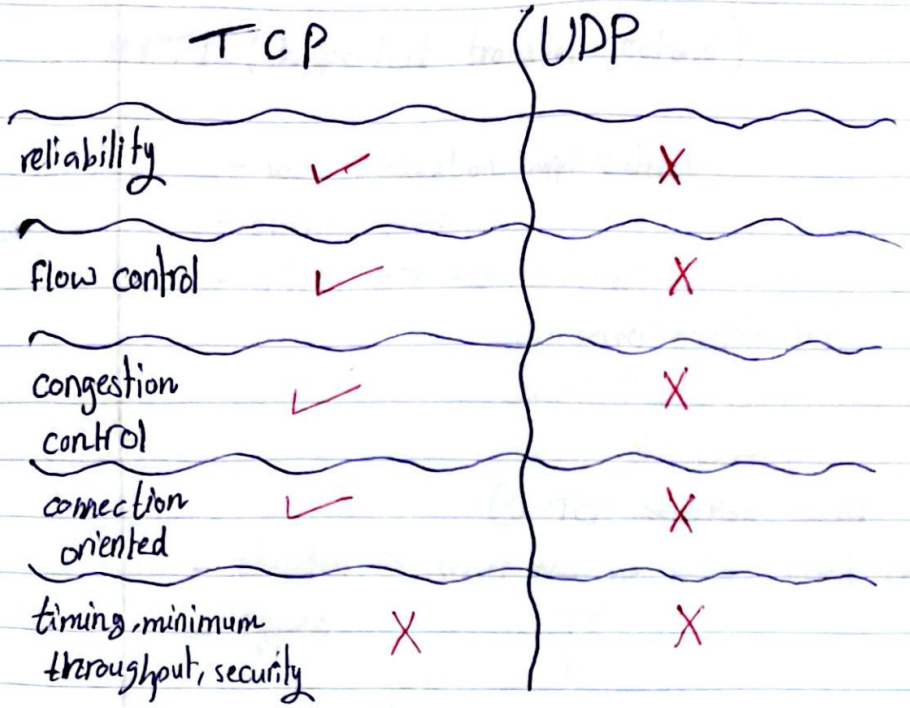
- Process : program running within a host.
- inter-process communication : how to processes in the same host communicate.
- processes on different hosts communicate by exchanging messages

- client processes : process that initiates communication.
- server process : process that waits to be contacted.

} p2p has both

- socket : process sends / recieves messages to / from its socket.

- identifier of a process is both IP address & port number
  - IP address ↳ 32 bit
- port of HTTP server is 80 & mail server is 25



- then why is there UDP ?? because its simple & doesn't need setup, so when we don't need reliable transport we use it

- Vanilla TCP & UDP sockets:
  - No encryption
  - clear text passwords sent into socket traverse internet in clear text

- TLS (transport layer security)
  - encrypted TCP conn. - data integrity - end point authentication
  - in application layer (apps use TLS libraries)

HTTP (hypertext transfer protocol)

- web's application layer protocol
- client/server
- TCP used ⇒ ① client initiates TCP conn. & creates socket to port 80.
  ② server accepts TCP
  ③ HTTP messages exchanged between browser & webserver.
  ④ TCP connection closed
- stateless ⇒ server has no history about past client requests
- Types:-

HTTP

Non persistent
- Parallel ← non parallel
- a TCP connections is used for requesting one object.

persistent (HTTP 1.1)
- pipelining ← non pipelined
- multiple objects can be requested with one TCP connection.

- RTT: time for a packet to travel from client to server & back.
- HTTP response time (per project)

non persistent
- 2RTT + file transmit time
- issues :- 1) 2 RTTs per project
  2) OS overhead for each TCP conn.
- solution : parallel TCP conn.

persistent
- non pipelined: client sends a request & waits for the response before sending another request
  → $(n+1)RTT + \sum_{i=1}^{N} FTT$
- pipelined: client send multiple requests to server without waiting for previous response
  → $3RTT + \sum_{i=1}^{N} FTT$

HTTP messages

① HTTP request message

ASCII

- request line → Get .html
- header lines start

end ⇒ \r\n in a line alone mean the end of header lines.

- body

types:

ⓐ post method :- user input

ⓑ Get method :- (for data sending to a server after url & ?)

ⓒ head method :- requests only headers that would be returned if specified URL were requested with an HTTP Get method.

ⓓ put method :- uploads new file (object) to server

② HTTP response message :-

- status line (protocol, status code, status phrase)
- header lines ⇒ end with line \r\n
- data requested

status codes :-

I. 200 K — object request succeeded its in this message

II. 301 moved permenantly — requested object moved, new location in this message (field)

III. 400 Bad request — request message not understood by servers.

IV. 404 Not found — requested object not on this server

V. 505 HTTP version not supported

- stateful protocol :· client makes to changes to x or none at all

- cookies are used to maintain user/server state between transactions.

cookie components ⇒ 1) cookie header line of HTTP response message.
2) cookie = = in next HTTP request message
3) cookie file kept on user's host managed by user's browsers
4) back-end database at website.

- use of cookies :-
• authorization • shopping carts • recommendations • user session state.

* Web Cachas (proxy servers)
⇒ satisfy client request without involving origin server.
⇒ they act like both client & server, client when they request sth from origin server & server when they response to an existing response.

⇒ an object allowable of caching is in response header:
cache-control: max-age = <seconds>
no-cache

⇒ cache is installed by ISP

* conditional get is used so an object is not sent if cache has up to date version
cache:- if modified since: (date)
server:- no object if up to date
HTTP/1.0 304 Not Modified

HTTP 1.1 ⇒ multiple pipelined gets over a TCP connection.
⇒ FCFS responds by server to get requests.
⇒ loss recovery stalls object transmition.

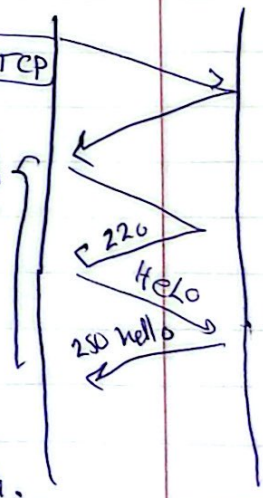
HTTP 2 ⇒ methods, status codes, header fields same as HTTP 1.1
⇒ transmission order of objects is By priority
→ divides objects to frames to lessen HoL Blocking
⇒ same as HTTP 1.1 in loss recovery & no security over vanilla TCP connection.

HTTP 3 → added security

* Email components → user agents
→ mail server → mailbox :- contains incoming messages for user
→ message queue : outgoing messages
→ SMTP → between mail servers to exchange messages

① client sends their message to their mail server.
② client mail server of SMTP opens TCP conn. with server mail.
③ SMTP client sends message over tcp
④ message is put in SMTP server in mailbox.

SMTP RFC (5321) ⇒ uses TCP, port 25
① TCP ③ transfer
② handshaking ③ closure of SMTP
- commands : ASCII
- response : status code & phrase.
↳ 220 SMTP ready
221 service closing
250 Request completed
354 start message input & end with .
MAIL FROM DATA
RCPT TO QUIT



- SMTP: protocol for exchanging email messages defined in RFC 5321
- HTTP is defined in RFC 7281
- RFC 2822 defines syntax for email like html defines syntax web browsers.
- IMAP (Mail access protocol) :- retrieval from server (RFC 3501)

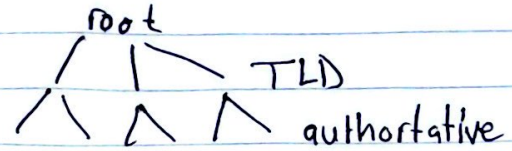
DNS

- IP 32 bit
- name

DNS is used to map between IP address & name.

- Distributed base in name servers
- Application layer protocol

- why not centralize DNS??
  - single point of failure
  - traffic
  - maintenance
  - distant



- Local DNS : ⇒ Does not belong to hierarchy
  ⇒ each ISP has one

DNS: distributed database storing resource records.
RR format (name, value, type, ttl)

- RR types :-

① A
(host name, IP, A, TTL)

② NS
(domain, hostname of authoritative, NS, TTL)
name server for this domain

③ CNAME
(alias, canonical, CNAME, TT)

④ MX
(~~alia~~ domain, canonical, MX, TT)

* DNS → reply
↘ query

identification 16 bit
flags 16 bit
- query 0 reply 1
- recursion desired
- recursion avilable
- reply is authoritative