Day 1

Networking
Client-server architecture
based on request-response model

Why networking ?

OSI - Open System Interconnection Model
TCP/IP - TCP/IP architecture

Application
Transport
Network
Link

Layer
Layers → stack

Application layer & transport layer are important in interview

→ Application → webbrowser, internet
↓
→ Transport → Datadelivery

Transport Layers

A → B
Speed
UDP
User Datagram Protocol

A → B
Safe, Secure
Reliable
TCP → Transmission Control Protocol

In interview when time TCP/UDP is enough

1) TCP

Hello
Ry → Hashing algo → ×
Hashing algo → ✓

→ what happens over call ?

3-way TCP Handshake / Connection Establishment

Sender SYN Receiver
SYN-ACK
ACK
A B

After Connection Establishment, we do Reliable Data Transfer
Reliable Data Transfer

we want to ensure
safe
reliable
error-free
Nothing is lost
Let's divide & break into segments

Segmentation → TCP segments
→ Sequence no

Termination

Client → FIN
Server → ACK

if internet happens we do retransmission
Checksum for flowing data validation

Server is sending so many ACK which is costly
so a technique called Flow Control which sends ACK for window of segments

RC TCP

1) Connect Establishment
- 3-way handshake
SYN
SYN-ACK
ACK

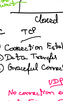
2) Data Transfer
- Segmentation
- Retransmission
- Seq. nos & Checksums

3) Graceful Connection Termination

Both client & server can terminate

Client → FIN → Server
Server → Ack → Client

4 way Handshake



RC TCP

1) Connection Establishment
2) Data Transfer
3) Graceful Connection Termination

UDP

No connection establishment
- just Fire & Forget
- stateless communication
- every udp packet is independent
- contains enough information for server

TCP	UDP
1) Transmission Control Protocol	1) User Datagram Protocol
2) Reliable - guarantee delivery of data	2) Unreliable - May get corrupted
3) Connection oriented protocol	3) Connectionless protocol
4) Overhead of establishing connection	4) no overhead for establishing, maintaining / termination
5) Error checking	5) Basic error checking
6) Sequencing & Ordering	6) No sequencing
7) Slower	7) Faster
8) Retransmission of lost packets	8) No retransmission
→ Safe, Trustworthy	→ Quick
→ Email, Multimedia, Banking	→ Gaming, Video, Music Streaming
→ Broadcasting, client-oriented	→ Broadcasting, support

DNS

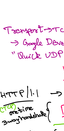
URL → IP
DNS uses UDP for most speed
↓
TCP for large data.

Application Layer

HTTP
HTTPS
SMTP
FTP
DNS
SNMP
RTP
POP
IMAP

Lets focus on HTTP & HTTPS

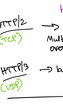
HTTP



Transport - TCP, UDP
→ Google Developed "QUIC" on top of UDP
"Quick UDP Internet Connection"

HTTP

HTTP/1.1 → Persistent connection
Connection reuse
3-way handshake
Multiple req over single connection



Many Gets
Receive
Sequentially

→ Problem is every request had to be handled sequentially.
There is No Concurrency
Head of line blocking

HTTP/2 (TCP) → Multiplexing
Multiple req, & res concurrently over a single connection

HTTP/3 (UDP) → built over QUIC

Header → Key : value

Body { }

Method → GET, POST, PUT, DELETE,

Status Code → 200, 404

RC

In interview
→ Client - Server
→ when to use TCP, UDP
→ Compare TCP, UDP
→ how entire flow happens
→ http has different versions
→ every req - res header, body, method, status

HTTP Method

1) GET

2) POST

3) PUT

4) DELETE

DAY 1 EOD
16 Jan 2024