COMPUTER NETWORKS - III

1. Cookies (HTTP)
   - XSS
2. DNS — TLS
   - structure
   - process
   - SSL
3. TCP vs UDP
   - 3 way handshake — IP — wireshark, socket

CORS

- headers
- Comparison

HTTP is stateless protocol

→ key cont Cookies

instructions [GET]

-token

Server - DB

1 ① email t pw
Validate

② token

③ token goes
in your DB

Browser - HTTP coobie

(4) Sends the response token

OSI - Session

header

① login

C — S

② token

response

Set-coobie

Set-coobie & Session_id : 123

Cookie

Coobie

local storage

{ session_id : 123 }

On every call, it sets it in the header

1. login → set cookie with session_id
2. instructions → cookie : { session_id }
3. logout → clean cookie

→ if a cookie exists

Random string

→

{

session id : 123

token : H123

}

Server - cookie

- Set-cookie

Browser - cookie

implement

---

headers

- set-coobie - manuel
- coobie - automatic (session)

/login

Validate

http.setCoobie("session_id", 123)

Cookies — vital info

— token

<a href=" scamm.net/text" + window.cookies

XSS — Cross site scripting

Cookie theft

no vital/secure info should be stored in cookies

JWT - Json web tokens

- manually set JWT

Refresh - Automatically

HTTP Cookie JWT - 30 minutes

FE - HTTP

S

Authorization: JWT

}

---

① login → Server

{token} → local sto
→ IndexDB

① set-coobie
return Jwt

② [instruction → auto coobie & ...]
→ manully
→ Authorization: Jwt

JWT - short lived
- 30 min
- revoking

JWT token - ID token
- access token
} short lived
- refresh token - get a new one

Channel - articles
- 5 articles per month

coobies → 5 minutes

→ block cookies

→ Delete this cookie

DNS

Call Alan Turing < Turing Co.

Call → Turing Co. → reception

→ Secretary

→ phone number

Library →

DNS - address book of the internet

→ 172.1.8.9 - fb.com

→ 45.7.8.9 - google.com

→ IP address

Domain Name → IP address

---

400 million domains

1 byte * 20 → 20* 400 x 10^6

→ 4 GB

Problems
→ traffic
→ SPOF

} x 1 Server

DNS - hierarchy

1. Root

② TLD - Top level domain server

extension →

③ Authoritative

Name Server

④ Recursive resolver | DNS resolver

Data flow

DNS - receptionist

① enter google.com in browser

(2) DNS Recursive -

(3) Root Server

(4) TLD - com

(5) NS -

```
r = call DS
if ( r = Not_found)
call DS( R)
```

Name Server

IP address

cloudflare —

8.8.8.8 → Isp dns

→ google DNS

DNS → latency

⇒ robustness

www.google.com

⇒ 45.1.7.8

→ DNS recursion

Server — DNS

offbank
- Route 53
- Netlify - {www.goog..}
↓
1.1.1.1

Record - CNAME / AAA → Server

AW

6:05 - 6:10 google.com → 1.1.1.1

10:35 - 10:40

tanmoy.site - DNS

DNS spoofing

8.8.8.8

GeoDNS - metadata

TCP vs UDP

TCP/UDP → host to host

HTTP → process to process

HTTP/2 → TCP

C

S

IP + IP

IP + Port

host

server

Application

Headers

Methods

HTTP

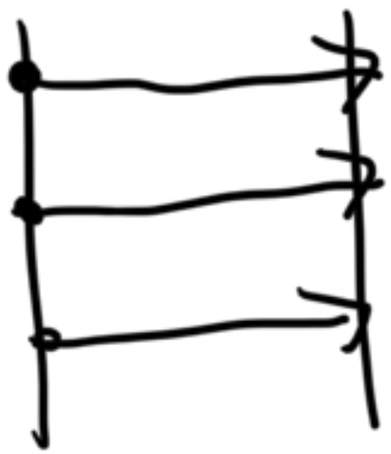
Transport

Origin

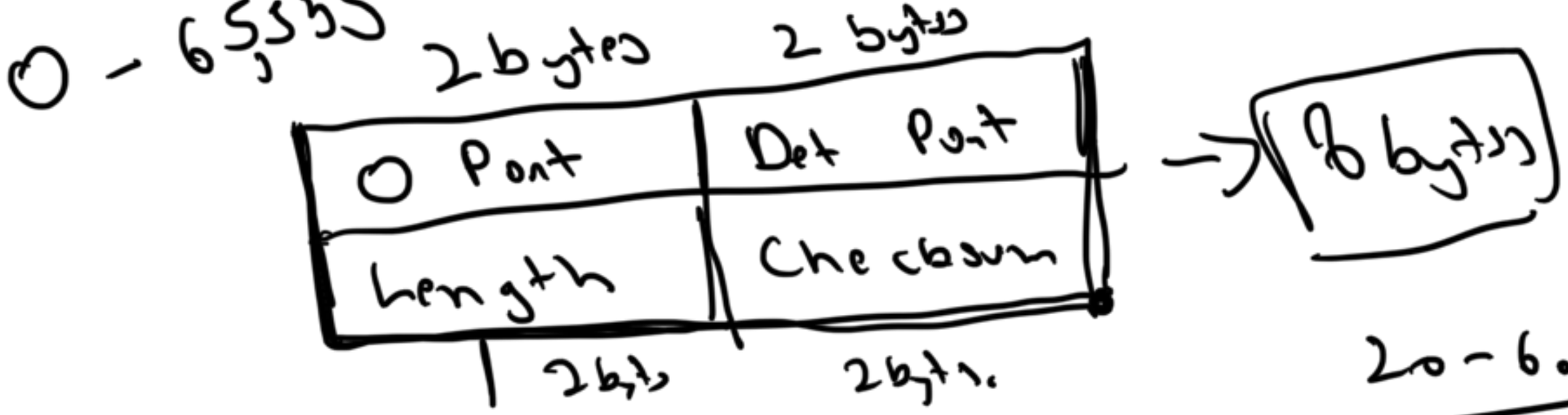
Destination

UDP – User Datagram Protocol

1. Faster
2. Connectionless
3. Reliable



0 - 65535

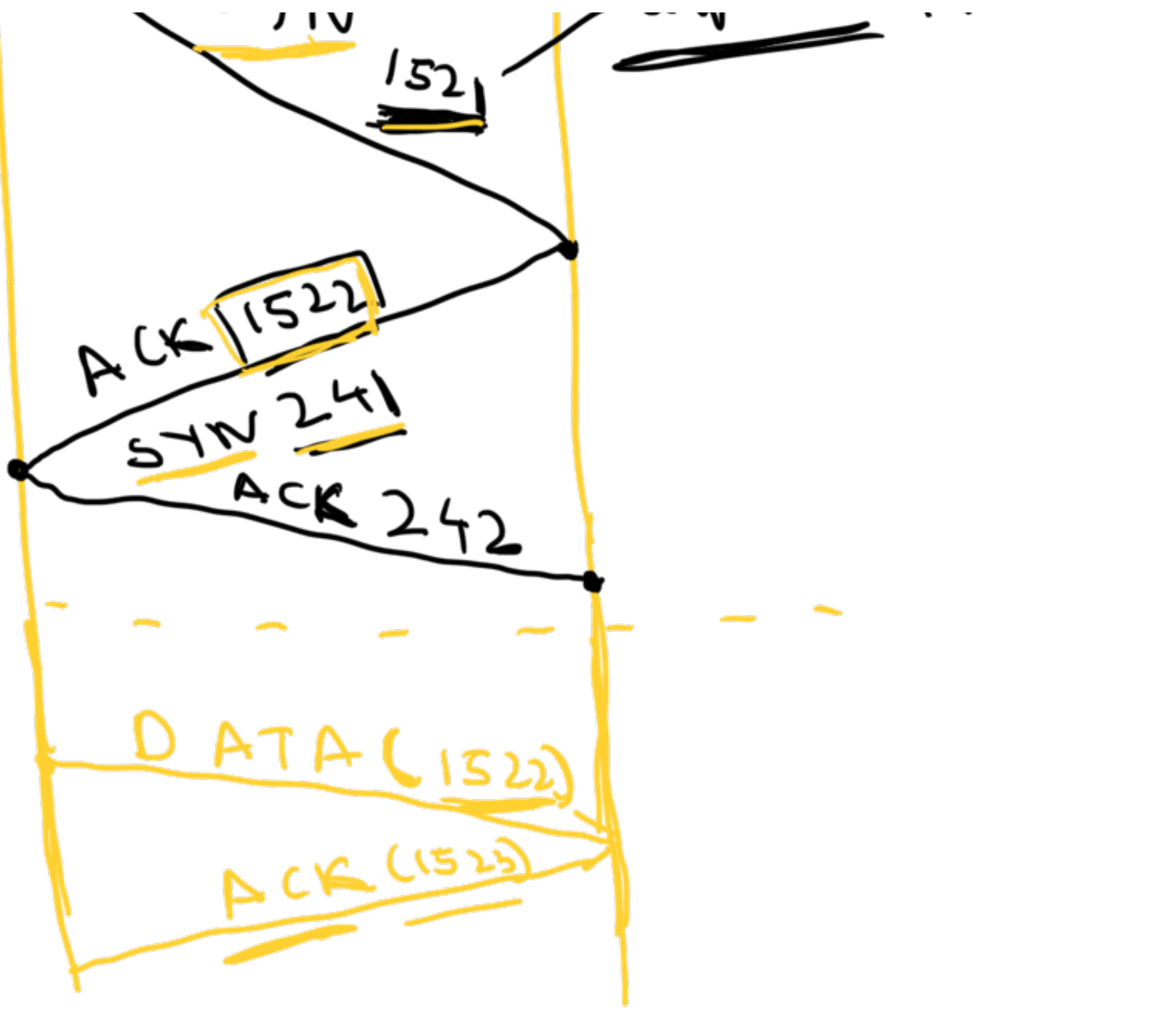


Content "Hijack"

TCP

1. connection - oriented
   - 3 way handshake
2. reliable
3. error-checking

3 way handshake


152
ACK 1522
SYN 241
ACK 242
DATA (152)
ACK (152)

C
S
① - SYN
(1521)
SYN
①
② - ACK
(1521)
+
SYN
(241)
SYN ACK
②
③ ACK
(242)
③
ACK
④ SYN
(1522)
④
⑤ ACK
(1523)
⑤

(1523)
6
6
7
Data
ACK 1524
time out
Data — ACK

ACK (152)

ACK + FIN - 4 way handshake..

① C - SYN (X) FIN

② S - ACK(X+1) + SYN (Y)

③ C - ACK(Y+1)

④ C - SYN(X+1)

⑤ S - ACK(X+2) + SYN (Y+1)

. . .

4 way handshake - FIN

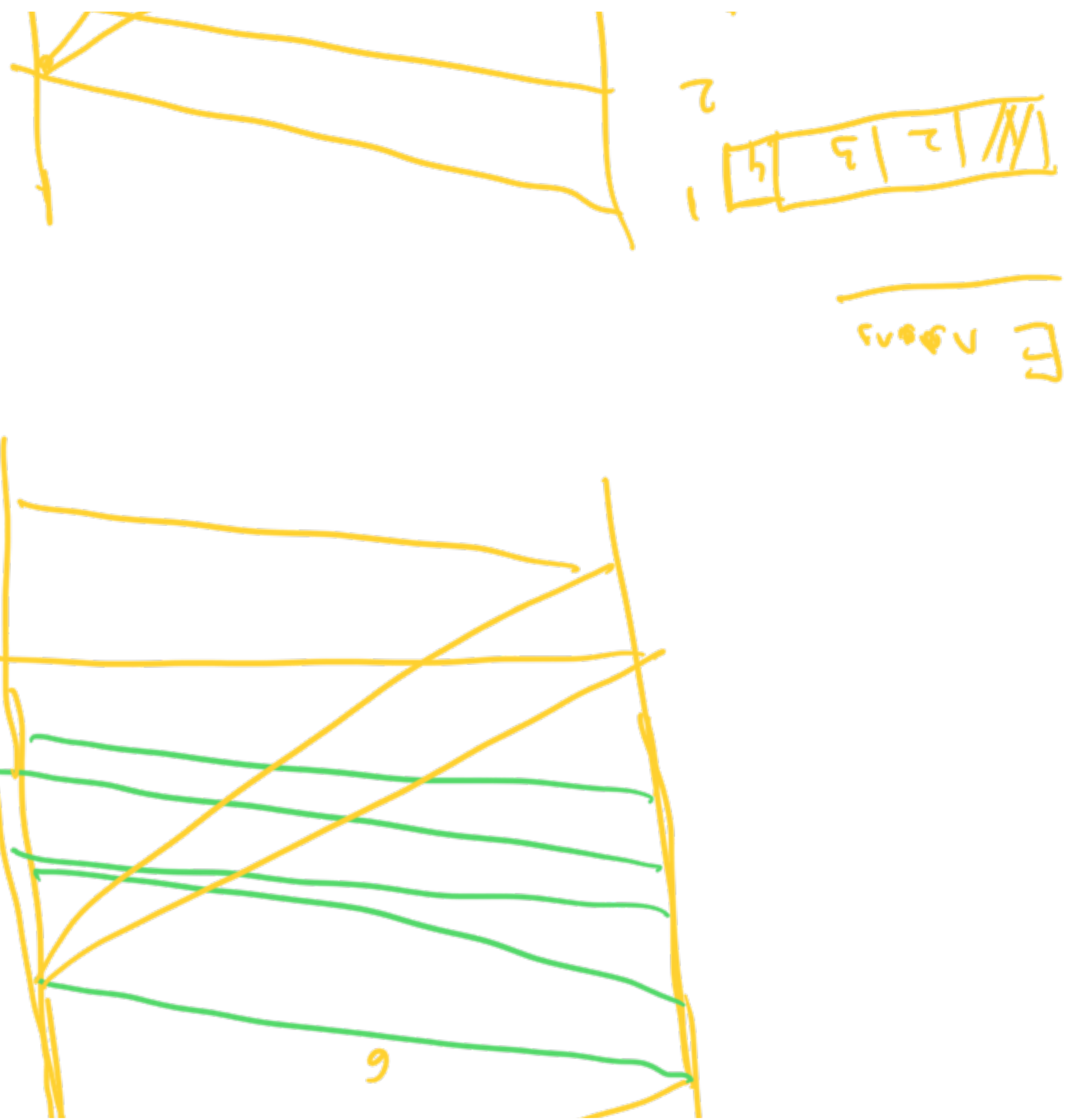
SYN + Ack + FIN

Python, C++, C

TCP vs UDP

Sequential

Sliding Window protocol


9

TCP

HTTP/2 → TCP

HTTP/3 → QUIC
(UDP)

TCP header

① Origin Port | SYN Number | ACK Number

② Dest Port

Flags - 9 bits

Checksum | Window - Window Size

If we cannot afford packet loss

→ TCP

Video streaming, VoIP

→ UDP

FileZilla

FTP Server

HTTP

TCP

|

→ reliable

→ connection oriented

stateful

→ error handling

UDP

packet loss

connectionless

stateless

no error handling

→ slow

→ extensive validation

→ segments

→ ordered

→ bonking

→ fast

checksum

datagram

→ streaming DNS
VoIP

Auth0 → Request body

cookies — expiry — long lived

token — short lived

HTTPS — e2e

C — S

HTTP — TLS

Request + certificate

encode + decode → verify

Run level encoding

document.cookies

SWT - local storage
- index DB

client - HTTP to HTTPS

http:

HSTS list

https

HSTS - HTTP strict transport security

{

token : { value : 123

escape

}

http only

}

XSS

gmail.com