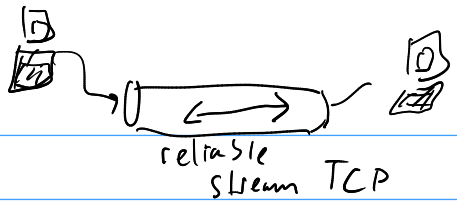
CS 3214 "HTTP"



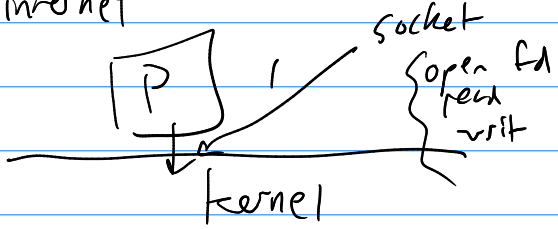
SPOT surveys

Networking: 7-layer

7. app
4. transport TCP/UDP ← sockets
3. network (IP)
2. link (ethernet)
1. phys

Nice properties of TCP

congestion control ← "saved the internet"
flow control

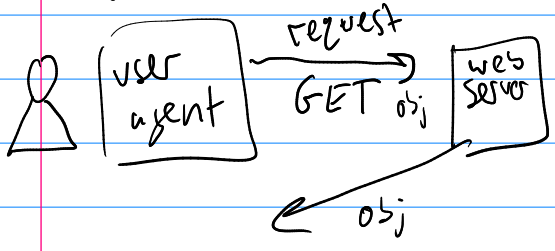


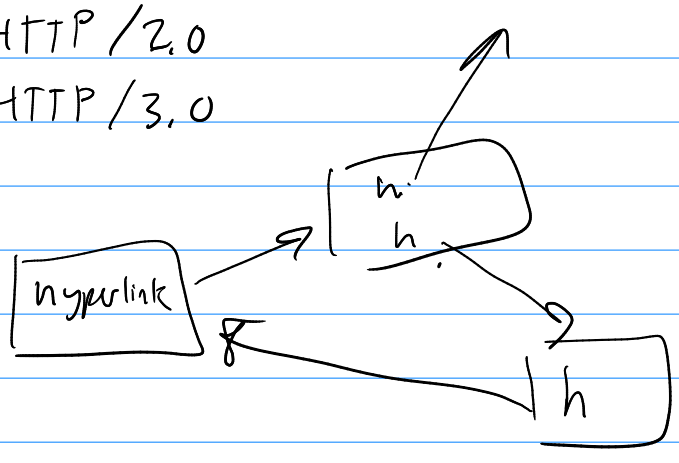
Application-layer

HTTP - world wide web www
modern web apps (REST)

1991 Tim Berners-Lee HTTP 0.9 RFC
1996, 1997 HTTP/1.0
/1.1
2015 HTTP/2.0
HTTP/3.0

Request/Response





obj: URL
universal record locator

http(s): // john:pw@www.example.com : 123 /forum/questions/ ? tag=foo&a=5
scheme — host — port — path — query

Smtp://
sftp://
ftp://

DNS
authority

#top
fragment

responses: Status codes
200 OK
301 permanently moved
404 Not found
⋮

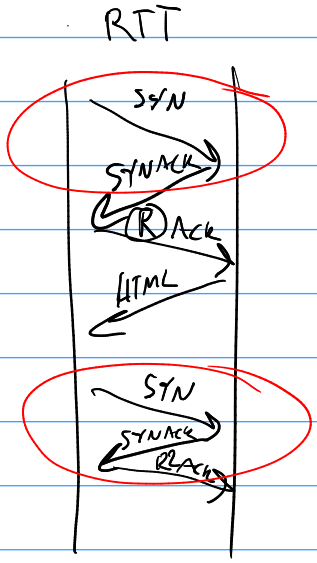
requests
GET - requesting transfer
POST - sending data to be processed
PUT
⋮

REST

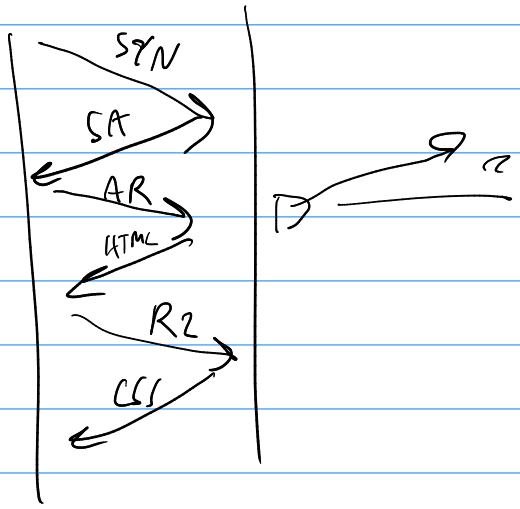
HTTP protocol is stateless
Cookies: clients present tokens to recognize reqs as belonging to session info

fetch a webpage
- multiple requests HTML, img, css, JS, fonts...

HTTP 1.0
- new conn per request
- handshake has a cost
- inefficient for lots of small reqs

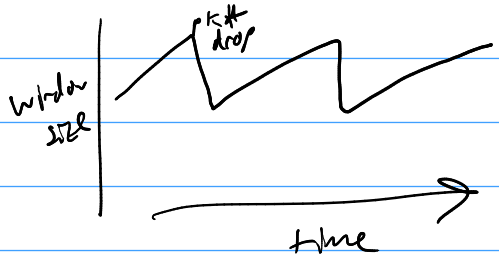


HTTP 1.1
- persistent connections
- forced an order
- multiple connections
  servers are built for concurrent reqs
  but
  want fairness



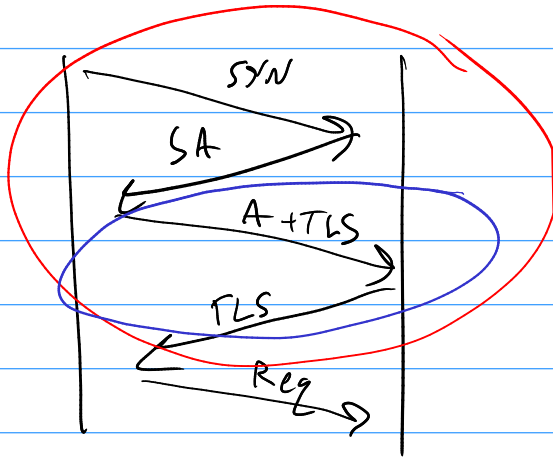
HTTP 2.0
- server sends back in any order
- can return chunks (frames)
- can push
  header compression

TCP slow start



Issues: pkt drops are expensive
- TCP retransmit/window size affects all reqs

- TCP handshake is still expensive
  TLS: "transport layer security"
  on top of TCP
  encryption/certs PKI authentication



HTTP/3 over QUIC
quick UDP internet connections

- UDP instead of TCP - avoid blocking entire stream on pkt loss
  - fold in connection handshake w/ TLS
  - → reimpl reliability
  - → reimpl congestion control
  - → connection + crypto handshake combined
  - → FEC (fwd error correction)
  - connection migration

mobile video uses QUIC
challenges NAT