CSCI-1680
HTTP II + TLS

Nick DeMarinis

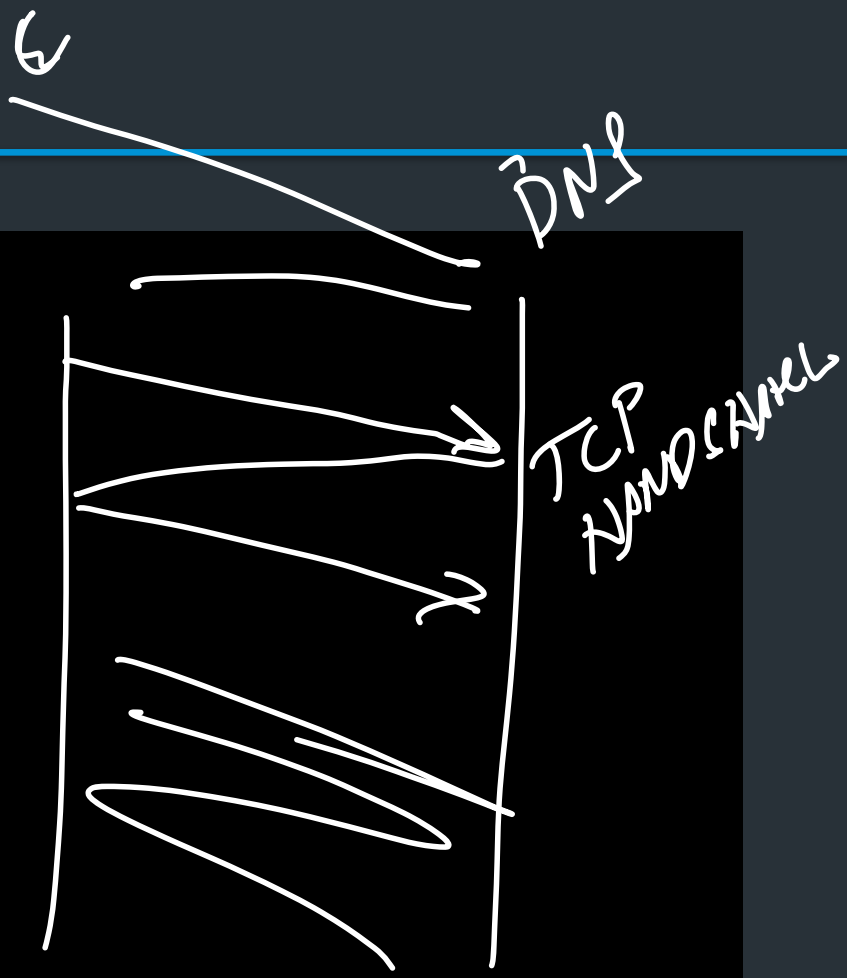
Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti

Administrivia

- Thursday, 12/8: last lecture
- If you haven't scheduled a TCP grading meeting, please do so (or contact me ASAP)
- Grading feedback: very soon
- Final project
  - Proposal feedback on Gradescope
  - If you want to talk, please come to office hours or ask for a meeting
- My office hours: today 3-5pm (Zoom), Thursday 3-5pm (CIT316)

HTTP

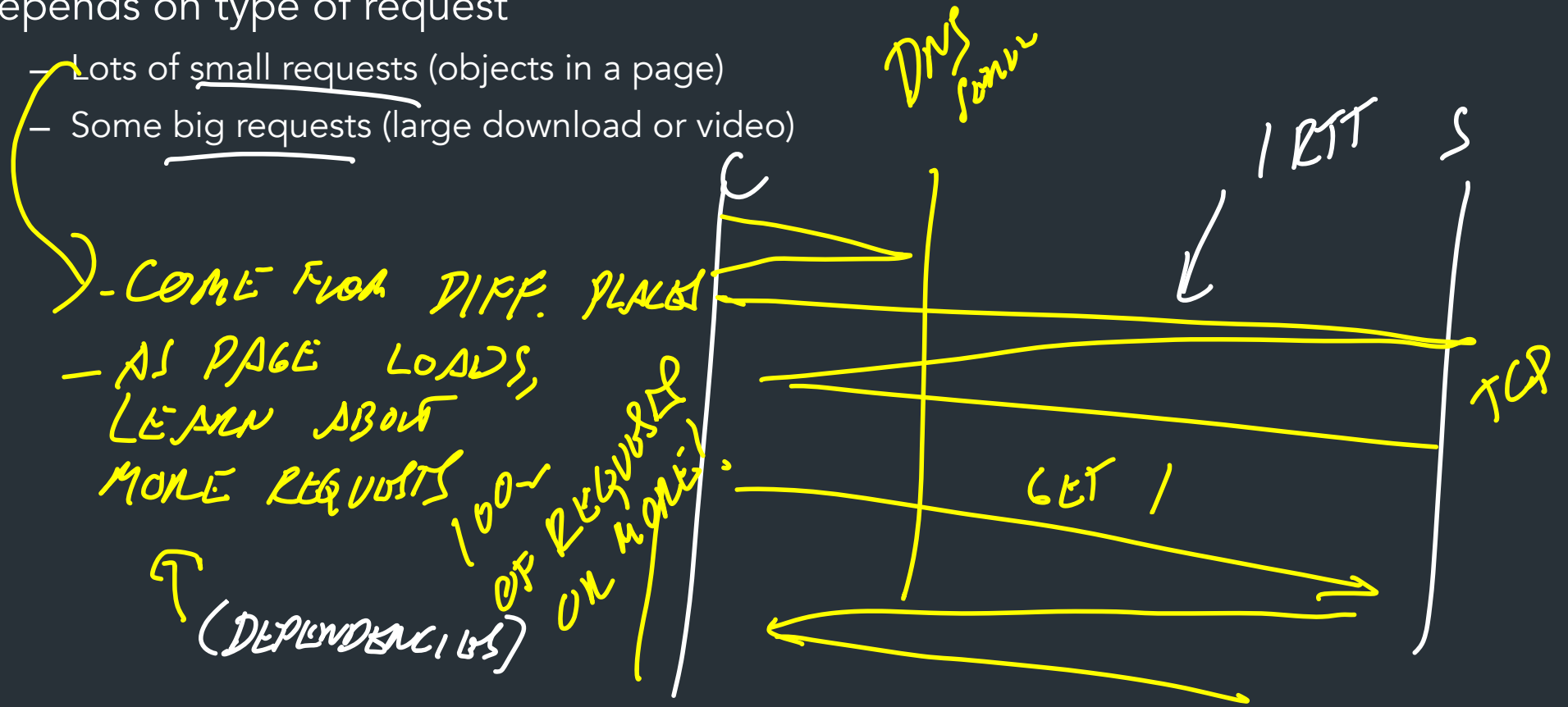
```
> telnet www.cs.brown.edu 80
Trying 128.148.32.110...
Connected to www.cs.brown.edu.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 24 Mar 2011 12:58:46 GMT
Server: Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Thu, 24 Mar 2011 12:25:27 GMT
ETag: "840a88b-236c-49f3992853bc0"
Accept-Ranges: bytes
Content-Length: 9068
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en”>
...
```



HTTP: What matters for performance?

Depends on type of request

- Lots of small requests (objects in a page)
- Some big requests (large download or video)



Small Requests

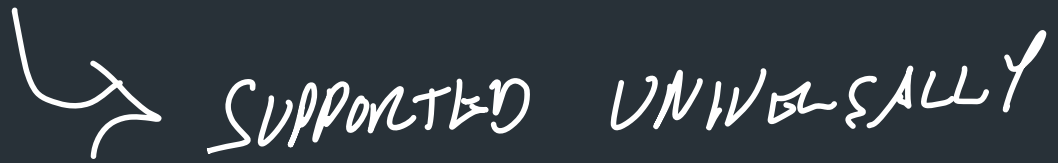
- Latency matters
- RTT dominates
- Major steps:
  - DNS lookup (if not cached)
  - Opening a TCP connection
  - Setting up TLS (optional, but now common)
  - Actually sending the request and receiving response

HTTP/1.0: NEED TO DO THIS FOR EACH REQUEST

How can we reduce the number of connection setups?

- DNS: caching
- HTTP uses TCP: Keep the connection open and request all objects serially
  - Works for all objects coming from the same server
  - Which also means you don't have to "open" the window each time

=> HTTP/1.1: Persistent connections

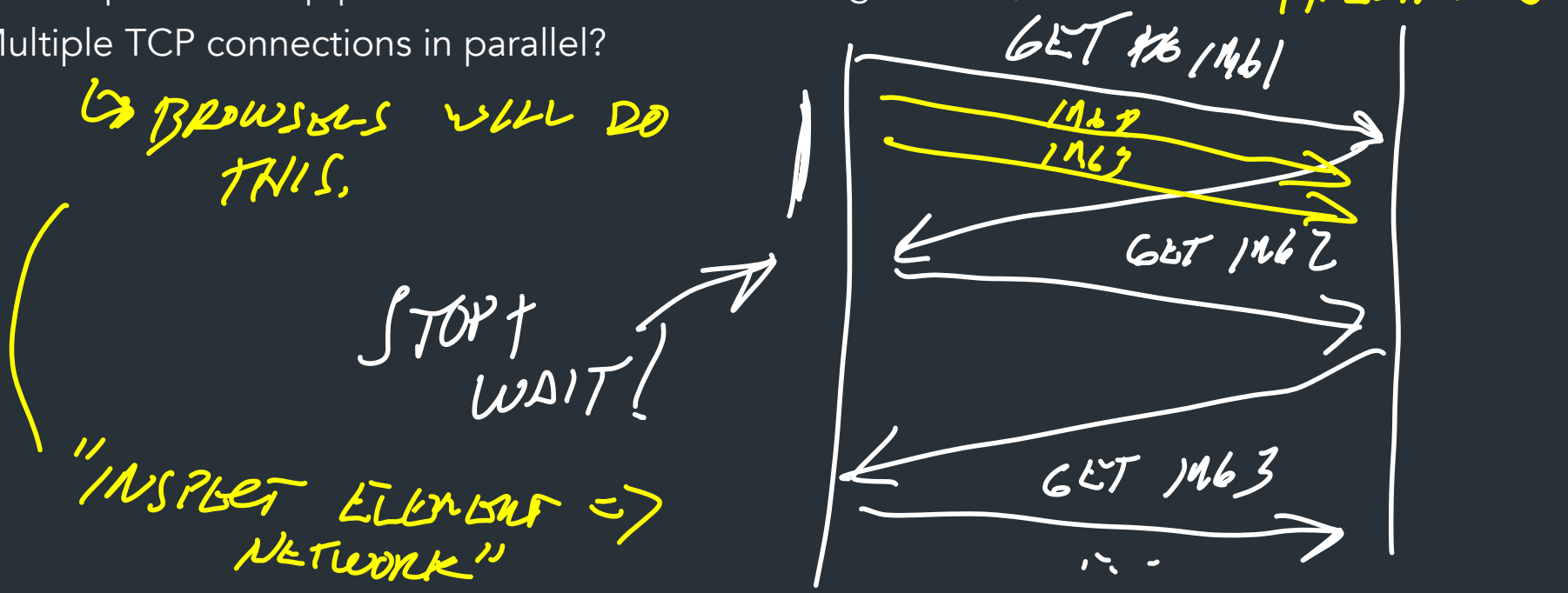


Browser Request

```
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macinto ...
Accept: text/xml,application/xm ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Small Requests (cont)

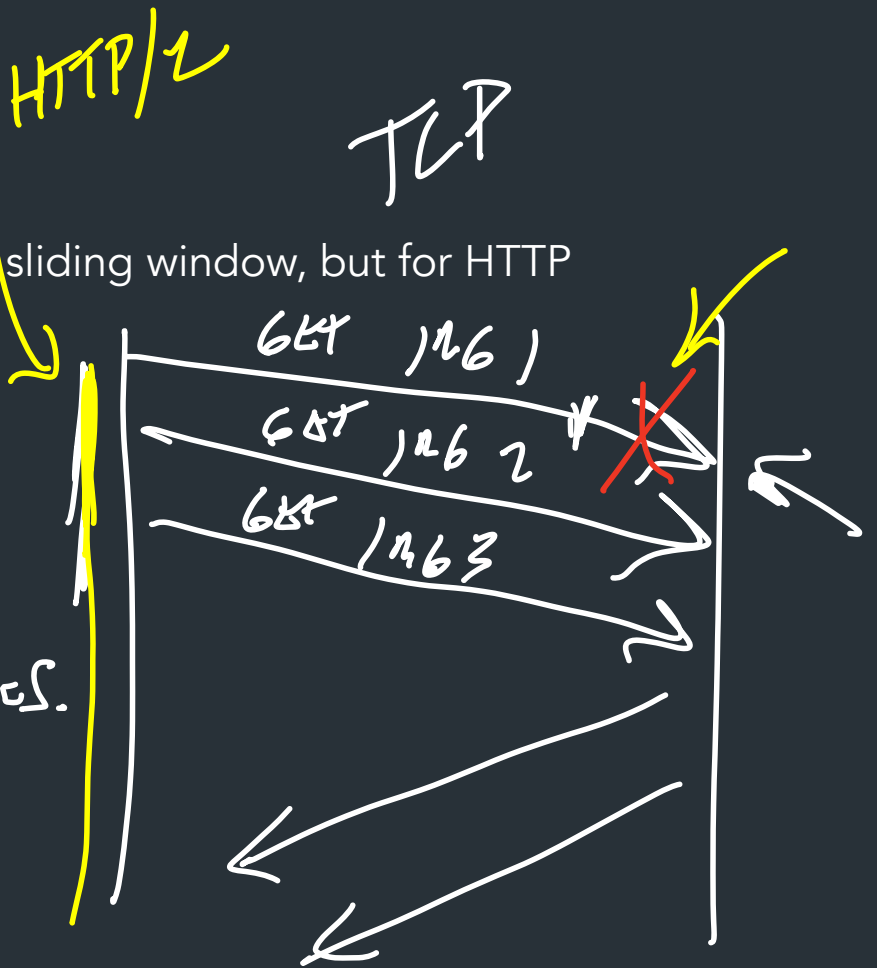
- Problem: requests are serialized
- Two solutions
  - Extend protocol to pipeline connections: like sliding window, but for HTTP
  - Multiple TCP connections in parallel?



Small Requests (cont)

- Problem: requests are serialized
- Two solutions
  - Extend protocol to pipeline connections: like sliding window, but for HTTP
  - Multiple TCP connections in parallel?

How do these differ?



⇒ IF PACKET LOST, DON'T GET LATER RESOURCES.

⇒ HEAD OF LINE BLOCKING

HTTP/2

Adds more options to trade off:

- Binary protocol
- Improved pipelining: can multiplex streams on same connection
  - Plus stream weights, dependencies

But what happens if there is packet loss?

ENCUMBERED BECAUSE PROTOCOL ISN'T AWARE OF LOSS W/ TCP (OR VICE VERSA)

https://www.twilio.com/blog/2017/10/http2-issues.html

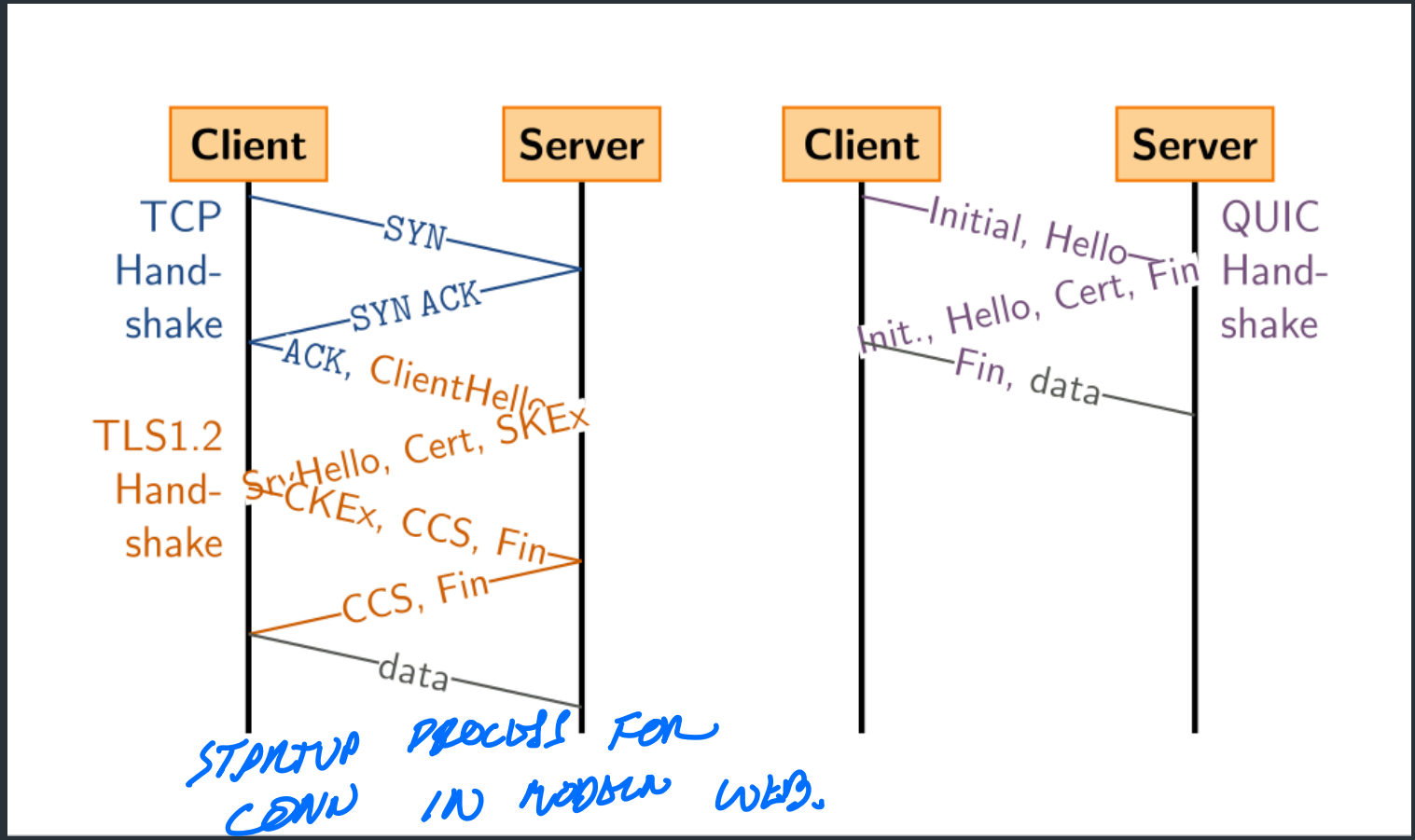
HTTP/3

- Mapping of HTTP semantics onto QUIC
  - E.g., QUIC already implements multiple streams, and HTTP doesn't need to do it
- QUIC: Another transport-layer protocol, intended to replace TCP
  - RFC9000
  - Same goals as TCP, but…
  - Integrates security by default (TLS, next class)
  - Supports multiple streams at once
  - Various tricks to reduce message size and latency
- By moving multiplexing into the transport layer, can do so in a way that benefits HTTP (no head of line blocking!)

NEW TRANSPORT LAYER PROTOCOL (REPLACES TCP IN CERTAIN CONDITIONS)

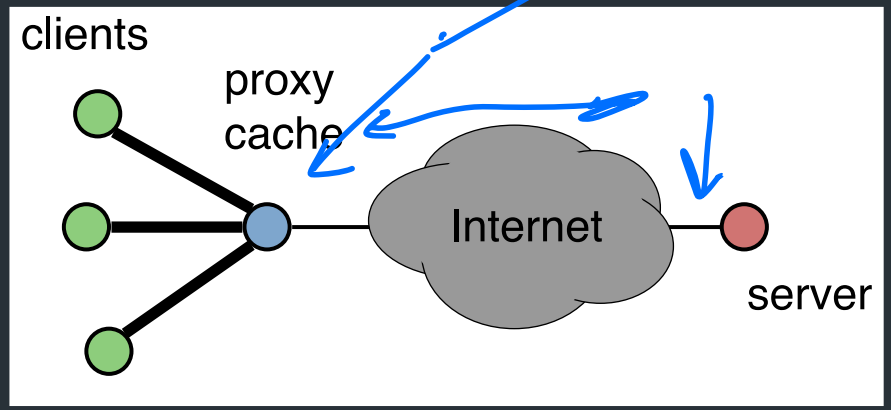
QUIC PROVIDES TLS

Comparison: QUIC's handshake



Larger Objects

- Problem is throughput in bottleneck link
- Solution: HTTP Proxy Caching
  - Direct requests to a proxy that can store some objects
  - Also improves latency, and reduces server load



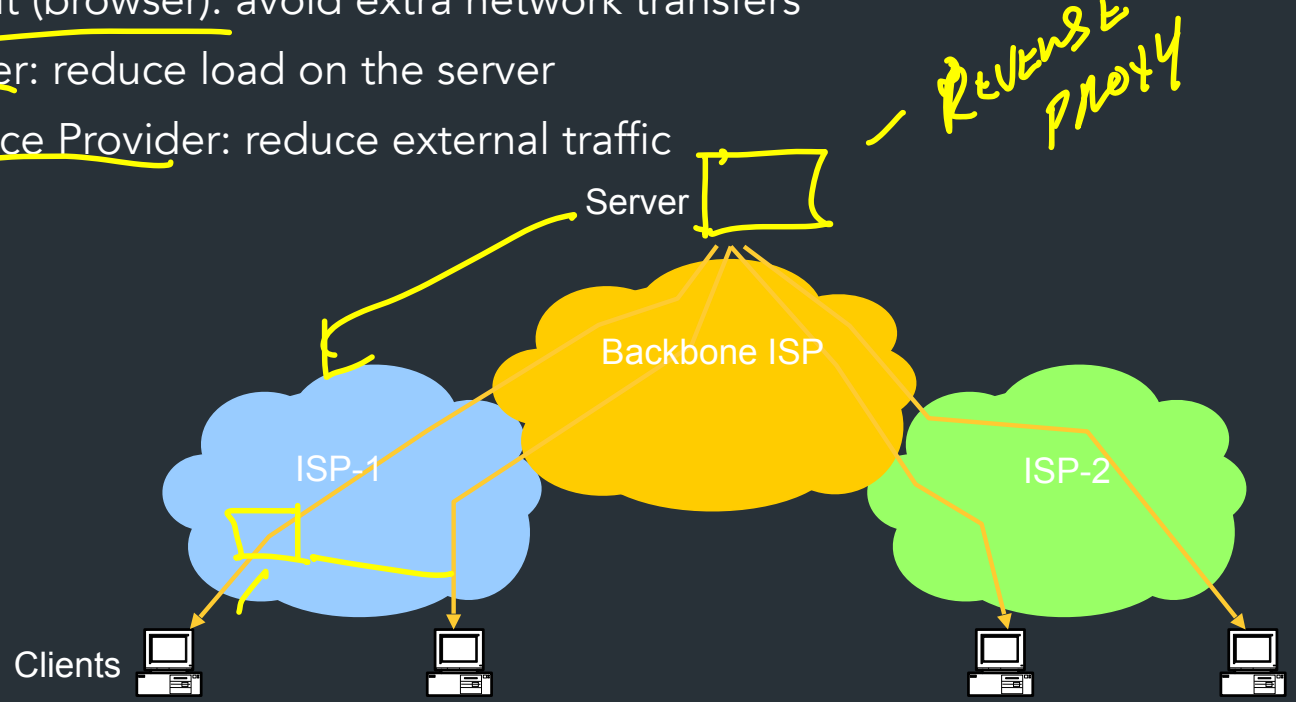
How to Control Caching?

- Server sets options
  - **Expires** header — TTL: How long to cache.
  - No-Cache header
- Client can do a conditional request:
  - Header option: if-modified-since
  - Server can reply with 304 NOT MODIFIED

Caching

Where to cache content?

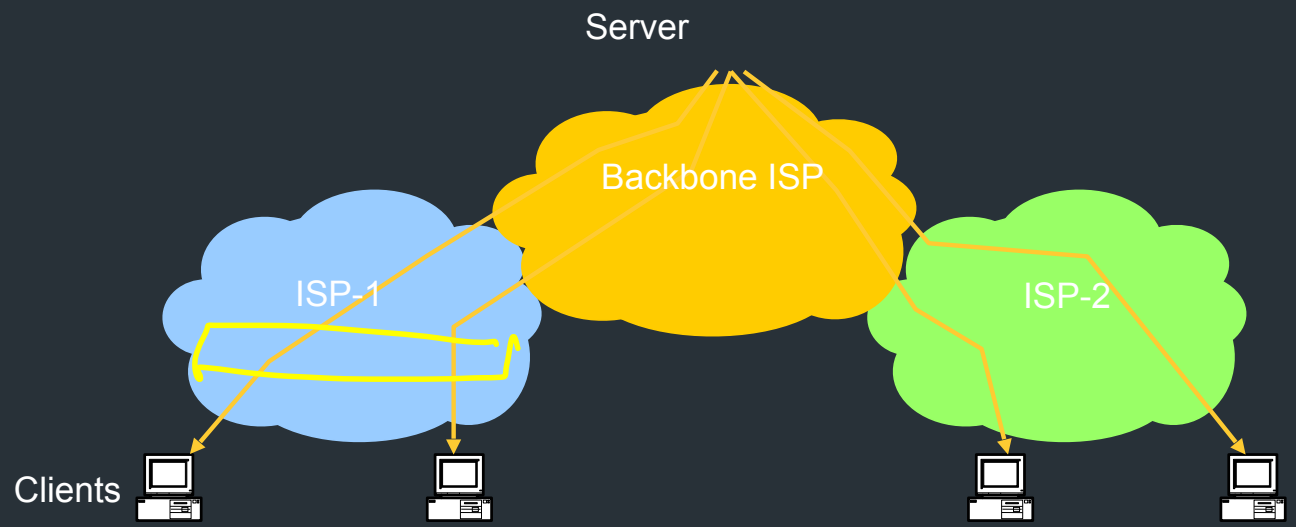
- Client (browser): avoid extra network transfers
- Server: reduce load on the server
- Service Provider: reduce external traffic



Why does caching work?

Locality of reference:

- Users tend to request the same object in succession
- Some objects are popular: requested by many users



How well does caching work?

- Very well, up to a point
  - Large overlap in requested objects
  - Objects with one access place upper bound on hit ratio
  - Dynamic objects not cacheable*
  - TLS can complicate this
- Example: Wikipedia
  - About 400 servers, 100 are HTTP Caches (Squid)
  - 85% Hit ratio for text, 98% for media

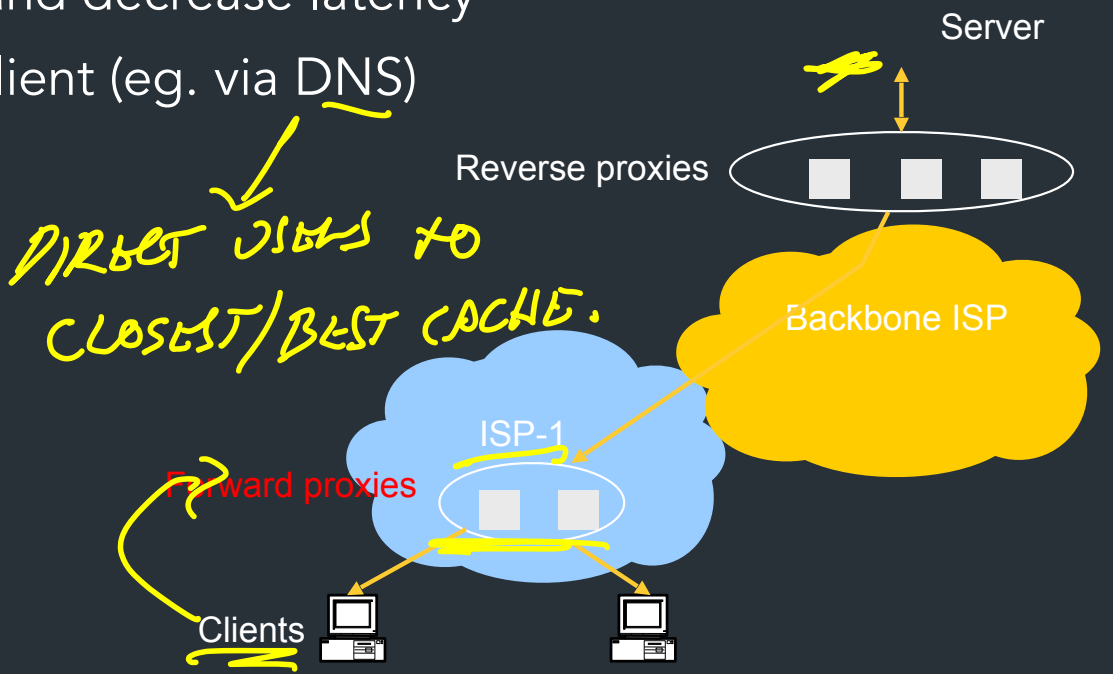
OPEN SOURCE HTTP PROXY/CACHE.

* But can cache portions and run special code on edges to reconstruct

Forward Proxies

Place a proxy "close" to the client

- Deployed in enterprise of ISP network => direct client requests here
- Reduce network traffic and decrease latency
- Can be transparent to client (eg. via DNS)



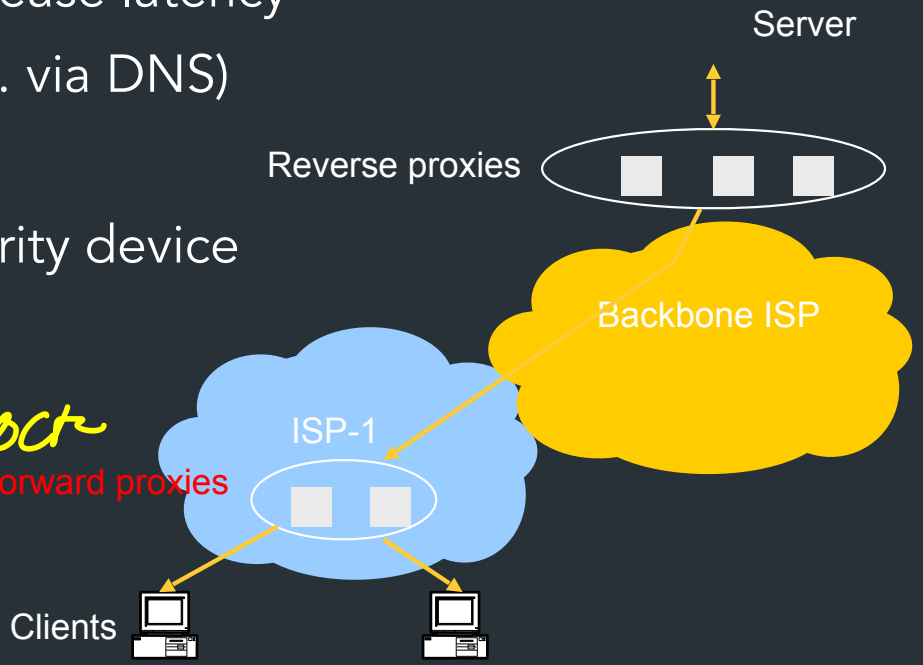
Forward Proxies

Place a proxy "close" to the client

- Deployed in enterprise of ISP network => direct client requests here
- Reduce network traffic and decrease latency
- Can be transparent to client (eg. via DNS)

Separately: can be used as a security device for clients

↳ CAN BE USED TO FILTER/BLOCK TRAFFIC



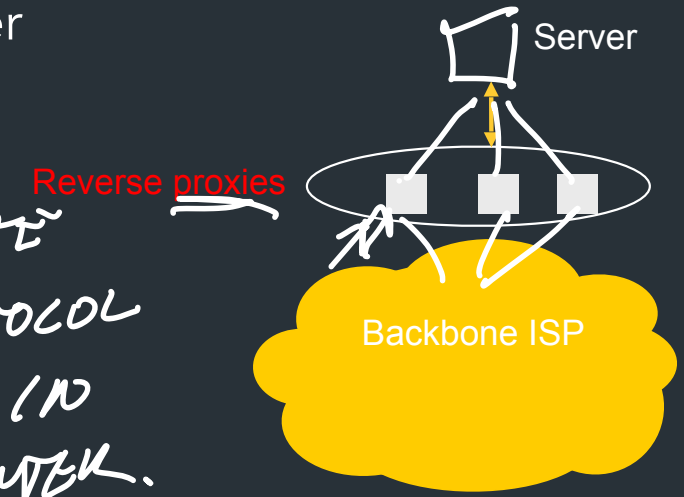
Reverse Proxies

Proxy that lives close to server

- Also called "accelerators"
- Can provide load balancing within datacenter
- Can also do other transformations (TLS, transcoding, etc…)

↳ CAN TRANSLATE TO DIFF. PROTOCOL FOR USE IN DATACENTER.

⇒ CAN HAVE SOME MORE INTELLIGENCE TO SPEED UP APPLICATIONS.



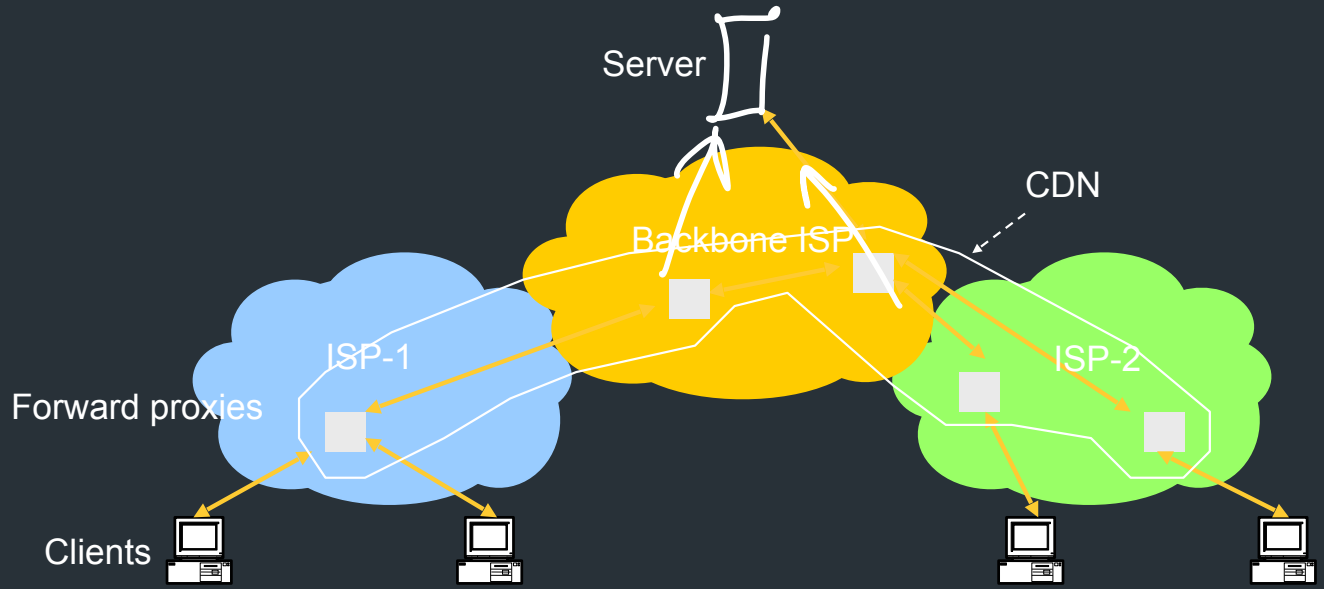
Content Distribution Networks (CDN)

- Integrate forward and reverse caching
  - One network generally administered by one entity
  - E.g. Akamai
- Provide document caching
  - Pull: result from client requests
  - Push: expectation of high access rates to some objects
- Can also do some processing
  - Deploy code to handle some dynamic requests
  - Can do other things, such as transcoding

→ e.g. TLS, ...

] CAN ADD LOGIC TO HOW REQUESTS ARE MADE FOR PERFORMANCE.

Example CDN



How Akamai works

Akamai has cache servers deployed close to clients

- Co-located with many ISPs

- Challenge: make same domain name resolve to a proxy close to the client
- Lots of DNS tricks. BestBuy is a customer
  - Delegate name resolution to Akamai (via a CNAME)

DNS Resolution

```
dig www.bestbuy.com
;; ANSWER SECTION:
www.bestbuy.com. 3600     IN       CNAME    www.bestbuy.com.edgesuite.net.
www.bestbuy.com.edgesuite.net. 21600 IN    CNAME    a1105.b.akamai.net.
a1105.b.akamai.net.       20       IN       A        198.7.236.235
a1105.b.akamai.net.       20       IN       A        198.7.236.240
;; AUTHORITY SECTION:
b.akamai.net.             1101     IN       NS       n1b.akamai.net.
b.akamai.net.             1101     IN       NS       n0b.akamai.net.
;; ADDITIONAL SECTION:
n0b.akamai.net.           1267     IN       A        24.143.194.45
n1b.akamai.net.           2196     IN       A        198.7.236.236
```

- **n1b.akamai.net finds an edge server close to the client's local resolver**
  - Uses knowledge of network: BGP feeds, traceroutes. Their secret sauce…

Example

From Brown

```
dig www.bestbuy.com
;; ANSWER SECTION:
www.bestbuy.com.  3600  IN    CNAME www.bestbuy.com.edgesuite.net.
www.bestbuy.com.edgesuite.net. 21600 IN   CNAME a1105.b.akamai.net.
a1105.b.akamai.net.     20    IN    A     198.7.236.235
a1105.b.akamai.net.     20    IN    A     198.7.236.240
```

- Ping time: 2.53ms

From Berkeley, CA

```
a1105.b.akamai.net.     20    IN    A     198.189.255.200
a1105.b.akamai.net.     20    IN    A     198.189.255.207
```

- Ping time: 3.20ms

dig www.bestbuy.com
;; QUESTION SECTION:
;www.bestbuy.com. IN A

;; ANSWER SECTION:
www.bestbuy.com. 2530 IN CNAME www.bestbuy.com.edgekey.net.
www.bestbuy.com.edgekey.net. 85 IN CNAME e1382.x.akamaiedge.net.
e1382.x.akamaiedge.net. 16 IN A 104.88.86.223

;; Query time: 6 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Nov 16 09:43:11 2017
;; MSG SIZE rcvd: 123

traceroute to 104.88.86.223 (104.88.86.223), 64 hops max, 52 byte packets
 1 router (192.168.1.1) 2.461 ms 1.647 ms 1.178 ms
 2 138.16.160.253 (138.16.160.253) 1.854 ms 1.509 ms 1.462 ms
 3 10.1.18.5 (10.1.18.5) 1.886 ms 1.705 ms 1.707 ms
 4 10.1.80.5 (10.1.80.5) 4.276 ms 6.444 ms 2.307 ms
 5 lsb-inet-r-230.net.brown.edu (128.148.230.6) 1.804 ms 1.870 ms 1.727 ms
 6 131.109.200.1 (131.109.200.1) 2.841 ms 2.587 ms 2.530 ms
 7 host-198-7-224-105.oshean.org (198.7.224.105) 4.421 ms 4.523 ms 4.496 ms
 8 5-1-4.bear1.boston1.level3.net (4.53.54.21) 4.099 ms 3.974 ms 4.290 ms
 9 * ae-4.r00.bstnma07.us.bb.gin.ntt.net (129.250.66.93) 4.689 ms 4.109 ms
10 ae-6.r24.nycmny01.us.bb.gin.ntt.net (129.250.4.114) 8.863 ms 10.205 ms 10.477 ms
11 ae-1.r08.nycmny01.us.bb.gin.ntt.net (129.250.5.62) 9.298 ms
    ae-1.r07.nycmny01.us.bb.gin.ntt.net (129.250.3.181) 10.008 ms 8.677 ms
12 ae-0.a00.nycmny01.us.bb.gin.ntt.net (129.250.3.94) 8.543 ms 7.935 ms
    ae-1.a00.nycmny01.us.bb.gin.ntt.net (129.250.6.55) 9.836 ms
13 a104-88-86-223.deploy.static.akamaitechnologies.com (104.88.86.223) 9.470 ms 8.483 ms 8.738 ms

dig www.bestbuy.com @109.69.8.51

e1382.x.akamaiedge.net. 12 IN A 23.60.221.144

traceroute to 23.60.221.144 (23.60.221.144), 64 hops max, 52 byte packets
1 router (192.168.1.1) 44.072 ms 1.572 ms 1.154 ms
2 138.16.160.253 (138.16.160.253) 2.460 ms 1.736 ms 2.722 ms
3 10.1.18.5 (10.1.18.5) 1.841 ms 1.649 ms 3.348 ms
4 10.1.80.5 (10.1.80.5) 2.304 ms 15.208 ms 2.895 ms
5 lsb-inet-r-230.net.brown.edu (128.148.230.6) 1.784 ms 4.744 ms 1.566 ms
6 131.109.200.1 (131.109.200.1) 3.581 ms 5.866 ms 3.238 ms
7 host-198-7-224-105.oshean.org (198.7.224.105) 4.288 ms 6.218 ms 8.332 ms
8 5-1-4.bear1.boston1.level3.net (4.53.54.21) 4.209 ms 6.103 ms 5.031 ms
9 ae-4.r00.bstnma07.us.bb.gin.ntt.net (129.250.66.93) 3.982 ms 5.824 ms 4.514 ms
10 ae-6.r24.nycmny01.us.bb.gin.ntt.net (129.250.4.114) 9.735 ms 12.442 ms 8.689 ms
11 ae-9.r24.londen12.uk.bb.gin.ntt.net (129.250.2.19) 81.098 ms 81.343 ms 81.120 ms
12 ae-6.r01.mdrdsp03.es.bb.gin.ntt.net (129.250.4.138) 102.009 ms 110.595 ms 103.010 ms
13 81.19.109.166 (81.19.109.166) 99.426 ms 93.236 ms 101.168 ms
14 a23-60-221-144.deploy.static.akamaitechnologies.com (23.60.221.144) 94.884 ms 92.779 ms 93.281 ms

Other CDNs

- Akamai, Limelight, Cloudflare
- Amazon, Facebook, Google, Microsoft
- Netflix
- Where to place content?
- Which content to place? Pre-fetch or cache?

Security topics + TLS

This is not a security class
(as much as I would like it to be…)

- This isn't intended to be a lecture on all crypto

- I want you to appreciate the important principles, understand what's important for TLS (and other protocols like it)

Want to know more?

- CS1660 (Spring): Intro to Computer Systems Security
- CS1510 (Fall): Intro to Cryptography and Computer Security

Internet's Design: Insecure

- Designed for simplicity in a naïve era
- Lots of insecure systems that can be compromised
- Attacks look like normal traffic
- Internet's federated operation obstructs cooperation for diagnosis/ mitigation

Basic Requirements for Secure Communication

- Availability
- Authentication
- Integrity:
- Confidentiality:

Basic Requirements for Secure Communication

- Availability ⇒ WILL NETWORK DELIVER DATA?
  - DoS DDoS
- Authentication: WHO IS THIS ACTOR? (CLIENT, SERVER?)
  - ATTACK: SPOOFING, IMPERSONATION
- Integrity: DO MESSAGES ARRIVE IN ORIGINAL FORM?
  ATTACK: TAMPERING.
- Confidentiality: CAN ADVERSARY READ DATA?
  ATTACK: EAVESDROPPING.
- Provenance:
  ↳ WHO IS RESPONSIBLE FOR DATA?

Basic Requirements for Secure Communication

- Availability: Will the network deliver data?
  - Infrastructure compromise, DDoS
- Authentication: Who is this actor?
  - Spoofing, phishing
- Integrity: Do messages arrive in original form?
- Confidentiality: Can adversary read the data?
  - Sniffing, man-in-the-middle
- Provenance: Who is responsible for this data?
  - Forging responses, denying responsibility
  - Not who sent the data, but who created it

Other Desirable Security Properties

- Authorization: is actor allowed to do this action?
  - Access controls

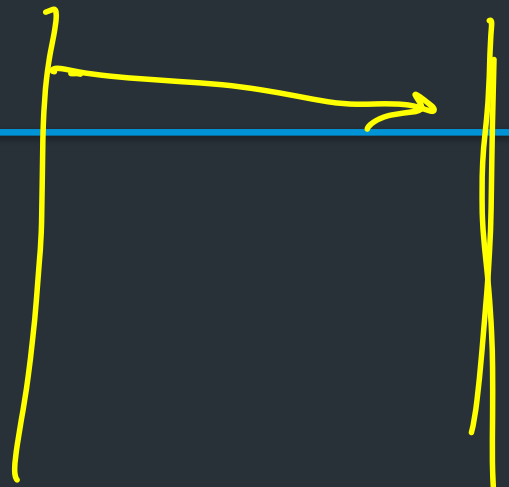
Other Desirable Security Properties

- Authorization: is actor allowed to do this action?
  - Access controls
- Accountability/Attribution: who did this activity?
- Audit/Forensics: what occurred in the past?
  - A broader notion of accountability/attribution
- Appropriate use: is action consistent with policy?
  - E.g., no spam; no games during business hours; etc.

Other Desirable Security Properties

- Authorization: is actor allowed to do this action?
  - Access controls
- Accountability/Attribution: who did this activity?
- Audit/Forensics: what occurred in the past?
  - A broader notion of accountability/attribution
- Appropriate use: is action consistent with policy?
  - E.g., no spam; no games during business hours; etc.
- Freedom from traffic analysis: can someone tell when I am sending and to whom?

What is TLS?



- Security for the transport layer

- Bidirectional pipe between two parties (client and server), but can enable:
  - Confidentiality
  - Integrity
  - Authentication

Is this all the security properties we might want? No!

1 – CLIENT MUST AUTHENTICATE SERVER

2 – EXCHANGE "SESSION KEY"
– CONFIDENTIALITY
– INTEGRITY

HOW DO WE AUTHENTICATE SERVER?

BOB.COM ⟶

- HOW ESTABLISH CONNECTION:
  - SERVER SENDS CERTIFICATE.
    - NAME
    - PUBLIC KEY FOR SERVER
    - SIGNATURE FROM CERTIFICATE AUTHORITY (CA)

ENCRYPTED

NEED TO PROVE PUBLIC KEY COMES FROM THIS ENTITY, VERIFY SIGNATURE

ASYMMETRIC CRYPTO

- EVERYONE GETS TWO KEYS FROM TRUSTED
  - PUBLIC: SHARE W/ ALL PARTY.
  - PRIVATE KEY: SECRET

ENCRYPTION: SEND TO B

- ENCRYPT W/ B'S $K_{PUB,B}$
  B NEEDS K_{PRIV} TO DECRYPT

SIGNATURE

$SIGN(M, K_{PRIV}) = D$

- B SIGNS W/ K_{PRIV}
- ANYONE CAN USE PUBLIC KEY

$VERIFY(M, K_{PUB}) = \{0, 1\}$

$\Rightarrow$ IF 1, ONLY SOMEONE W/ K_{PRIV} COULD HAVE MADE D

$\Rightarrow$ CAN PROVE THAT A MESSAGE CAME FROM ~~HOLD~~ ENTITY ~~W~~ THAT HELD PRIVATE KEY.

$\Rightarrow$ AUTHENTICATION

TWO FUNDAMENTAL PROBLEMS

1. NEED TO TRUST THAT K_{PUB} BELONGS TO THIS ENTITY $\Rightarrow$ CERTIFICATE AUTHORITY
2. ENCRYPTION / INTEGRITY: $K_{PUB} \Rightarrow$ SESSION KEY. PKI