CSCI-1680
WWW

Rodrigo Fonseca

Based partly on lecture notes by Scott Shenker and John Jannotti

Precursors

- **1945, Vannevar Bush, Memex:**
  - "*a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility*"
- **Precursors to hypertext**
  - "The human mind [...] operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate *web of trails* carried by the cells of the brain"
- **His essay, "As we may think", is worth reading!**

Tim Berners-Lee

- **Physicist at CERN, trying to solve real problem**
  - Distributed access to data
- **WWW: distributed database of pages linked through the Hypertext Transfer Protocol**
  - First HTTP implementation: 1990
  - HTTP/0.9 – 1991
    - Simple GET commant
  - HTTP/1.0 – 1992
    - Client/server information, simple caching
  - HTTP/1.1 – 1996
    - Extensive caching support
    - Host identification
    - Pipelined, persistent connections, …

- **HTTP/2 – 2015**
  - True multiplexing of messages
    - Multiple streams
    - Flow control
    - Prioritization
  - Binary encoding
  - Header compression
  - Main goal: reduce latency

http://httpwg.org/specs/rfc7540.html

Why so successful?

- **Ability to self publish**
  - Like youtube for video
- **But…**
  - Mechanism is *easy*
  - Independent, open
  - Free
- **Current debate**
  - Is it easy enough? Why is facebook so popular, even though it is not open?

Components

- **Content**
  - Objects (may be static or dynamically generated)
- **Clients**
  - Send requests / Receive responses
- **Servers**
  - Receive requests / Send responses
  - Store or generate content
- **Proxies**
  - Placed between clients and servers
  - Provide extra functions
    - Caching, anonymization, logging, transcoding, filtering access
  - Explicit or transparent

Ingredients

- **HTTP**
  - Hypertext Transfer Protocol
- **HTML**
  - Language for description of content
- **Names (mostly URLs)**
  - Won't talk about URIs, URNs

URLs

protocol://[name@]hostname[:port]/directory/resource?k1=v1&k2=v2#tag

- **URLs are a type of URIs**
- ***Name* is for possible client identification**
- ***Hostname* is FQDN or IP address**
- ***Port* defaults to protocol default (e.g., 80)**
- ***Directory* is a path to the resource**
- ***Resource* is the name of the object**
- **?*parameters* are passed to the server for execution**
- **#*tag* allows jumps to named tags within document**

HTTP

- **Important properties**
  - Client-server protocol
  - Protocol (but not data) in ASCII (before HTTP/2)
  - Stateless
  - Extensible (header fields)
- **Server typically listens on port 80**
- **Server sends response, may close connection (client may ask it to say open)**
- **Currently version 2**

Steps in HTTP(1.0) Request

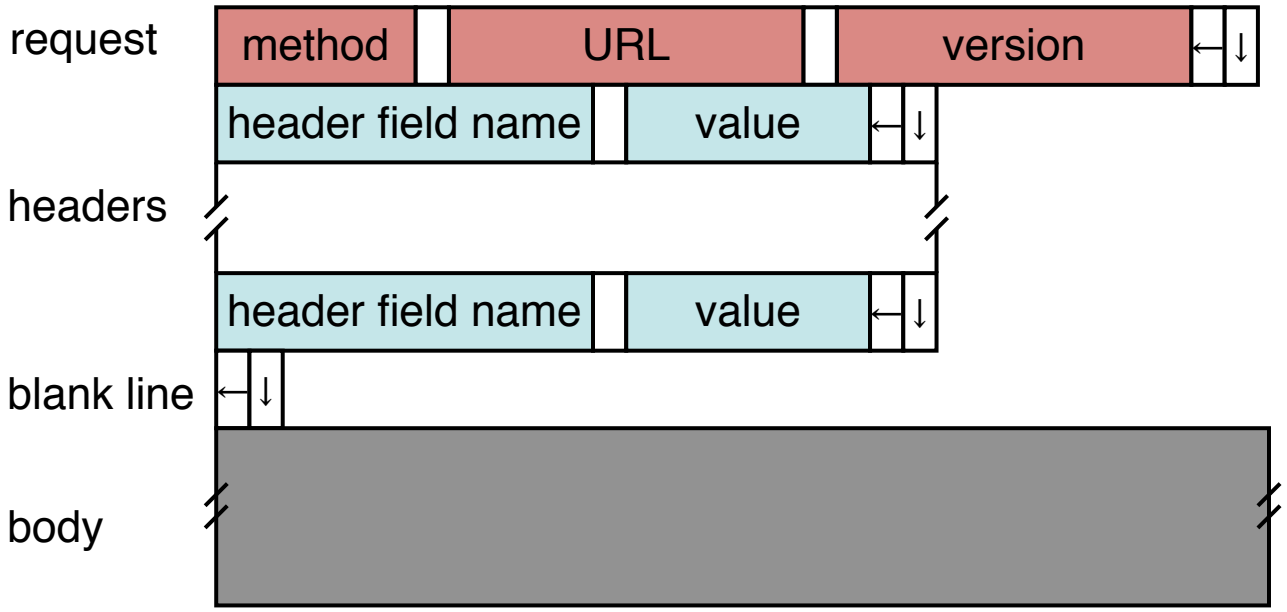
- **Open TCP connection to server**
- **Send request**
- **Receive response**
- **TCP connection terminates**
  - How many RTTs for a single request?
- **You may also need to do a DNS lookup first!**

```
> 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 Request



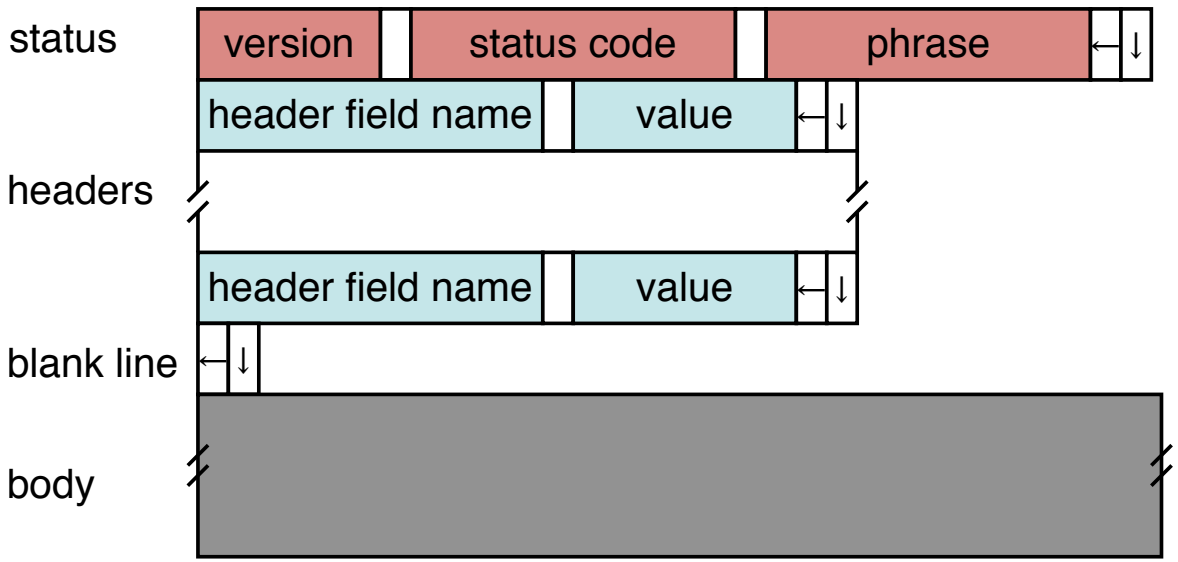
- **Method:**
  - GET: current value of resource, run program
  - HEAD: return metadata associated with a resource
  - POST: update a resource, provide input for a program
- **Headers: useful info for proxies or the server**
  - E.g., desired language

Sample 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
(empty line)
```

Firefox extension LiveHTTPHeaders is a cool way to see this

HTTP Response



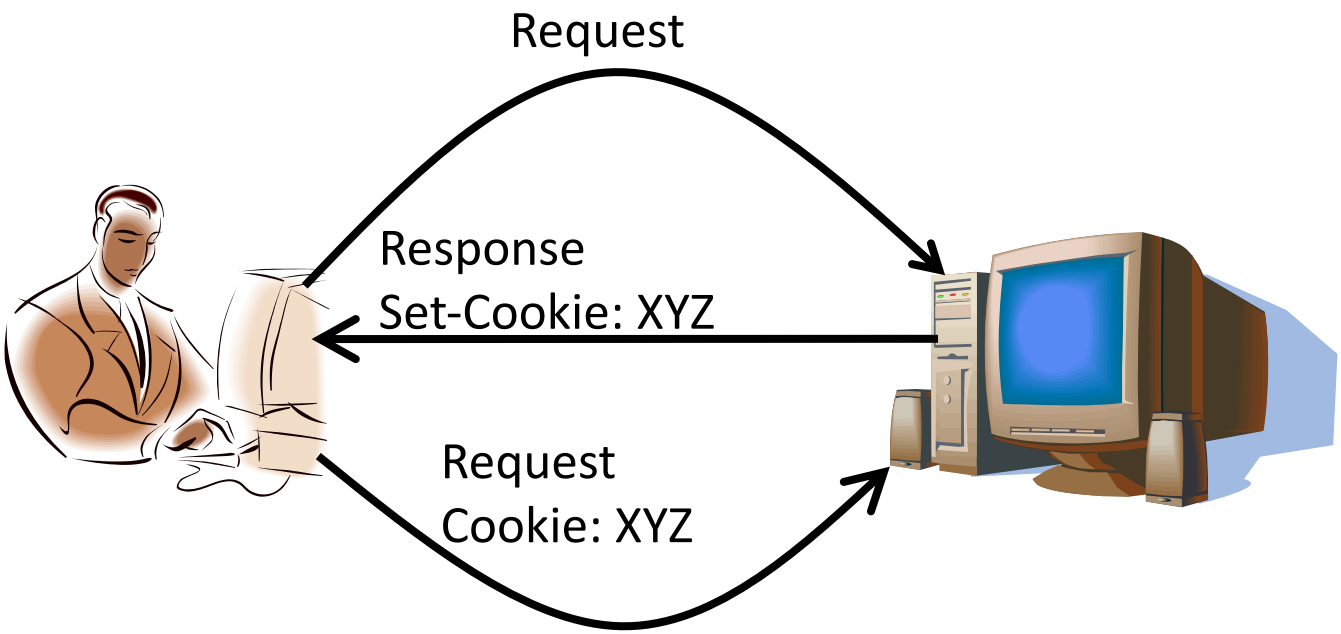
- **Status Codes:**
  - 1xx: Information e.g, 100 Continue
  - 2xx: Success e.g., 200 OK
  - 3xx: Redirection e.g., 302 Found (elsewhere)
  - 4xx: Client Error e.g., 404 Not Found
  - 5xx: Server Error e.g, 503 Service Unavailable

HTTP is Stateless

- **Each request/response treated independently**
- **Servers not required to maintain state**
- **This is good!**
  - Improves server scalability
- **This is also bad…**
  - Some applications need persistent state
  - Need to uniquely identify user to customize content
  - E.g., shopping cart, web-mail, usage tracking, (most sites today!)

HTTP Cookies

- **Client-side state maintenance**
  - Client stores small state on behalf of server
  - Sends request in future requests to the server
  - Cookie value is meaningful to the server (e.g., session id)
- **Can provide authentication**



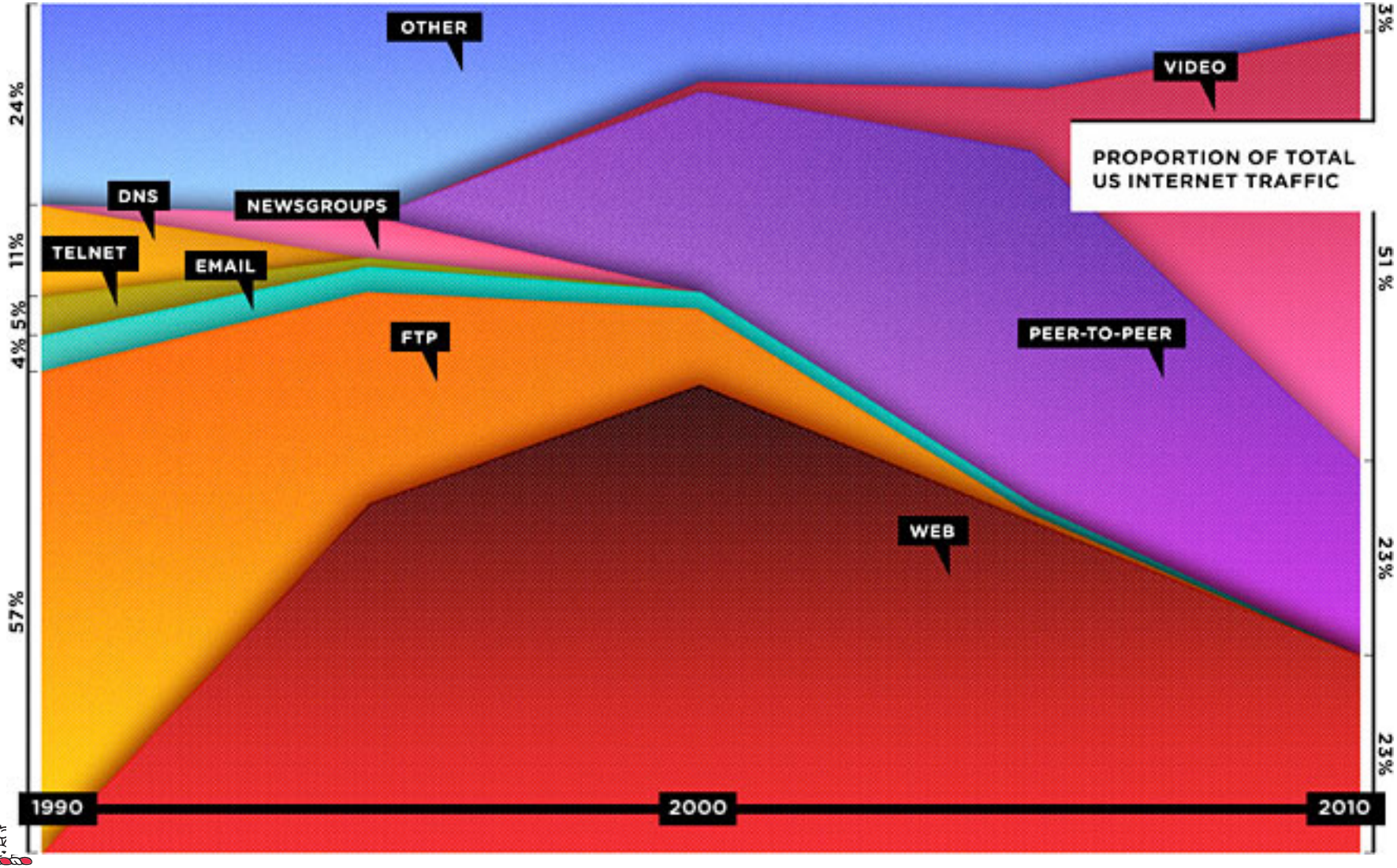
Anatomy of a Web Page

- **HTML content**
- **A number of additional resources**
  - Images
  - Scripts
  - Frames
- **Browser makes one HTTP request for each object**
  - Course web page: 14 objects
  - My facebook page this morning: 100 objects

What about AJAX?

- ***Asynchronous Javascript and XML***
- **Based on XMLHttpRequest object in browsers, which allow code in the page to:**
  - Issue a new, non-blocking request to the server, without leaving the current page
  - Receive the content
  - Process the content
- **Used to add interactivity to web pages**
  - XML not always used, HTML fragments, JSON, and plain text also popular

The Web is Dead? (Wired, Aug 2010)

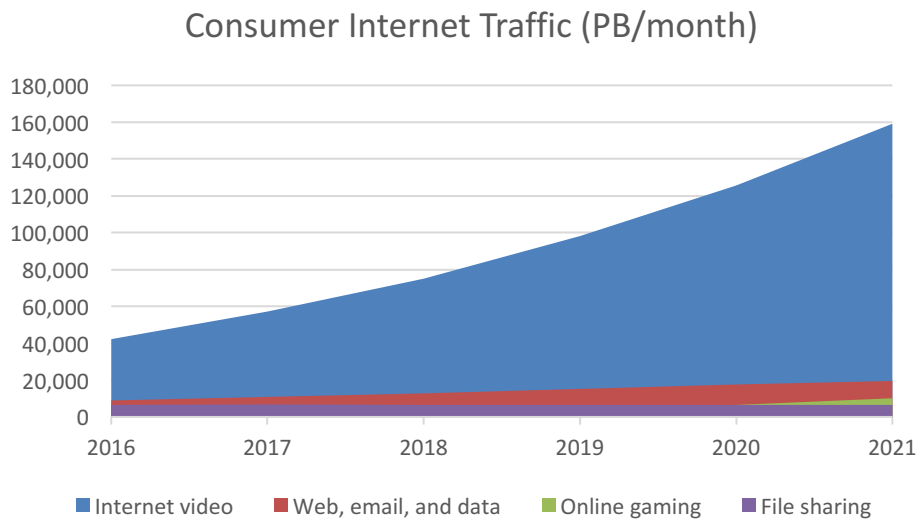


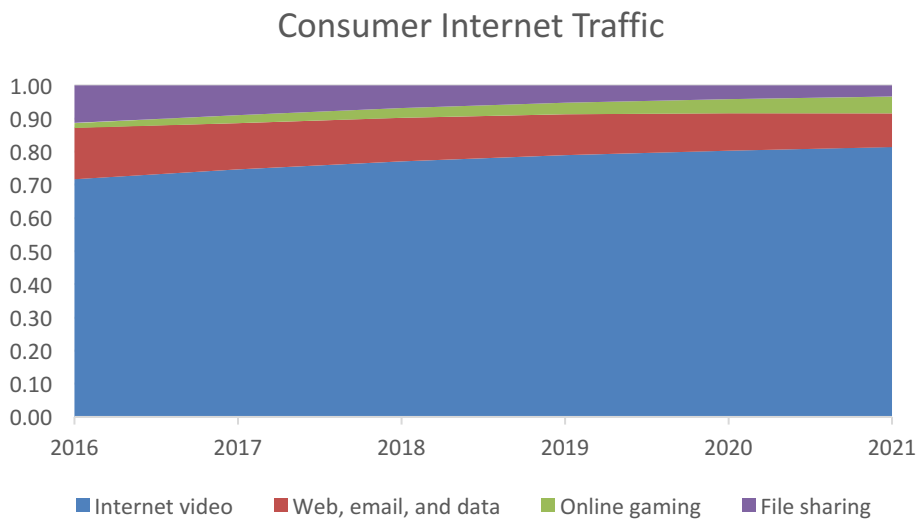
http://www.wired.com/magazine/2010/08/ff_webrip/all/1

The Web is Dead? (Wired, Aug 2010)

- You wake up and ***check your email*** on your bedside iPad — that's one app. During breakfast you browse Facebook, Twitter, and The New York Times — three more ***apps***. On the way to the office, you listen to a ***podcast*** on your smartphone. Another app. At work, you scroll through ***RSS feeds in a reader*** and have ***Skype*** and ***IM*** conversations. More apps. At the end of the day, you come home, make dinner while listening to ***Pandora***, play some games on ***Xbox Live***, and watch a movie on ***Netflix***'s streaming service. You've spent the day on the Internet — but not on the Web. And you are not alone.

Consumer Internet Traffic, 2016–2021							
	2016	2017	2018	2019	2020	2021	CAGR 2016–2021
By Subsegment (PB per Month)							
Internet video	42,029	57,116	75,109	98,182	125,853	159,161	31%
Web, email, and data	9,059	10,681	12,864	15,120	17,502	19,538	17%
Online gaming	915	1,818	2,857	4,396	6,753	10,147	62%
File sharing	6,628	6,810	6,717	6,554	6,388	6,595	0%





Except that video is *also* transferred over HTTP!

HTTP Performance

- **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**
- **Two major causes:**
  - Opening a TCP connection
  - Actually sending the request and receiving response
  - And a third one: DNS lookup!

How can we reduce the number of connection setups?

- **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
- **Persistent connections (HTTP/1.1)**

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)

- **Second problem is that requests are serialized**
  - Similar to stop-and-wait protocols!
- **Two solutions**
  - Pipelined requests (similar to sliding windows)
  - Parallel Connections
    - HTTP standard says no more than 2 concurrent connections per host name
    - Most browsers use more (up to 8 per host, ~35 total)
      - See http://www.browserscope.org/
  - How are these two approaches different?

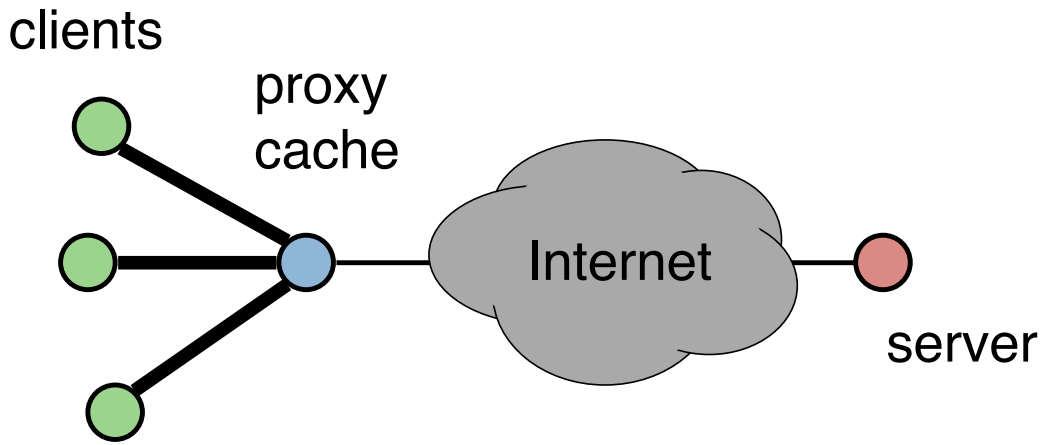
HTTP/2

- **Adds more options to trade off:**
- **Multiplexed streams on same connection**
  - Plus stream weights, dependencies
- **No head of line blocking!**
  - But what happens if there is packet loss?

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

Larger Objects

- **Problem is throughput in bottleneck link**
- **Solution: HTTP Proxy Caching**
  - Also improves latency, and reduces server load



How to Control Caching?

- **Server sets options**
  - **Expires** header
  - No-Cache header
- **Client can do a conditional request:**
  - Header option: if-modified-since
  - Server can reply with 304 NOT MODIFIED
- **More when we talk about Content Distribution**

Next Class

- **Global data distribution**
  - CDN and P2P
- **How to create your own application layer protocol!**
  - Data / RPC