HTTP/2-3 and DASH

COS 461 - Precept 8

Review: HTTP

HTTP: **H**yper**T**ext **T**ransfer **P**rotocol

Primary *application layer* protocol used for fetching and uploading web traffic

Used internally by your web browser, but can be implemented by any user-level application

You did this on Assignment 4!

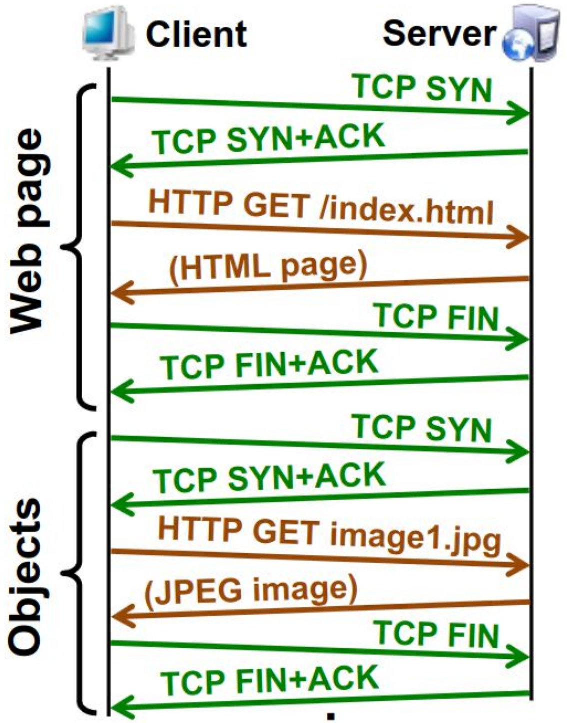
Review: HTTP/1.0

HTTP/1.0: Simple wrapper around TCP socket

Requires opening a new socket for each HTTP request

Requires 3 RTTs per request

Overhead is even worse when TLS is involved, because this requires establishing a new RSA key pair for each HTTP request.

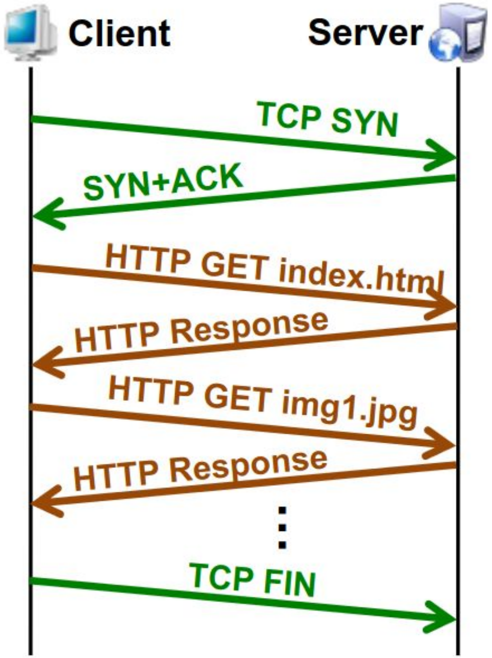


Review: HTTP/1.1

HTTP/1.1: Eliminates overhead of setting up a new socket for each connection

Web browser can cache sockets from recent connections and reuse them

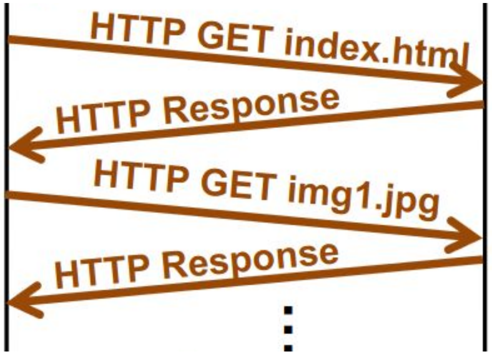
Client and server send keep-alive messages every few seconds to verify the connection is still live



HTTP/2

So what’s wrong with HTTP/1.1?

Web pages are fetched iteratively



Can we fetch resources all in one go?

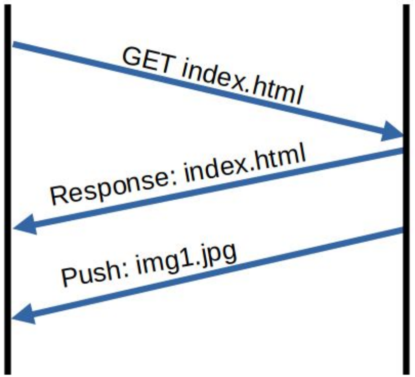
SPDY: Proposal by Google in 2009

“Server Push”: Server can return content the client did not directly request

HTTP/2

So what’s wrong with HTTP/1.1?

Web pages are fetched iteratively



Can we fetch resources all in one go?

SPDY: Proposal by Google in 2009

“Server Push”: Server can return content the client did not directly request

In the example to the left, the server can return index.html and img1.jpg at the same time.

HTTP/2

SPDY’s general strategy was adapted into HTTP/2 in 2015, along with other performance features:

Header compression (reduce data size)

Multiplexing (eliminates head-of-line blocking)

Prioritization of Requests (browser can specify which requests are most time-dependent)

HTTP/2 is now the dominant HTTP flavor used by browsers and servers.

HTTP/3

HTTP/3: New version of HTTP standard that is not yet widely deployed

Problem: HTTP/2 solves the head-of-line blocking problem at the application layer, but not the transport layer.

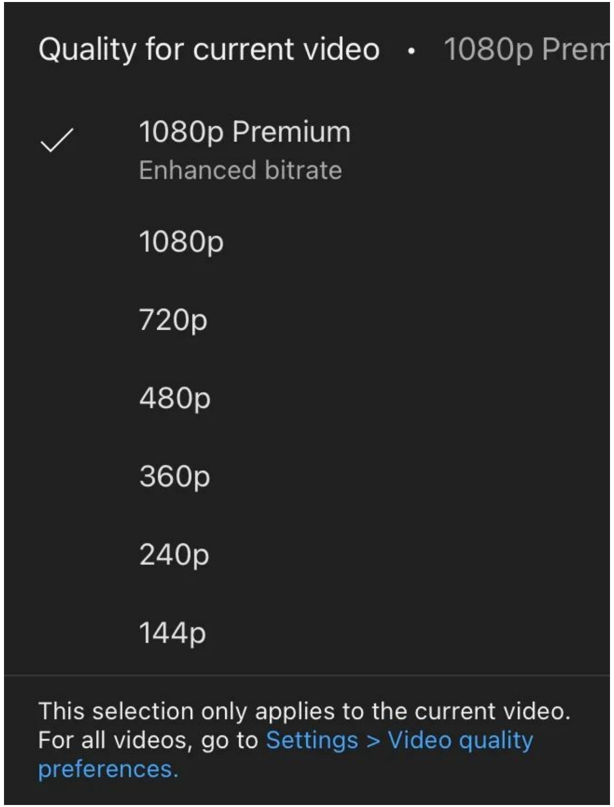
If TCP encounters a packet loss, this affect *all* open HTTP/2 requests.

HTTP/3 uses QUIC, which uses UDP instead of TCP and re-implements some TCP features in user-space.

Retransmission in HTTP/3 does not block other outstanding requests.

DASH

DASH - Dynamic Adaptive streaming over HTTP



Quality for current video • 1080p Prem

✓ 1080p Premium
Enhanced bitrate

1080p

720p

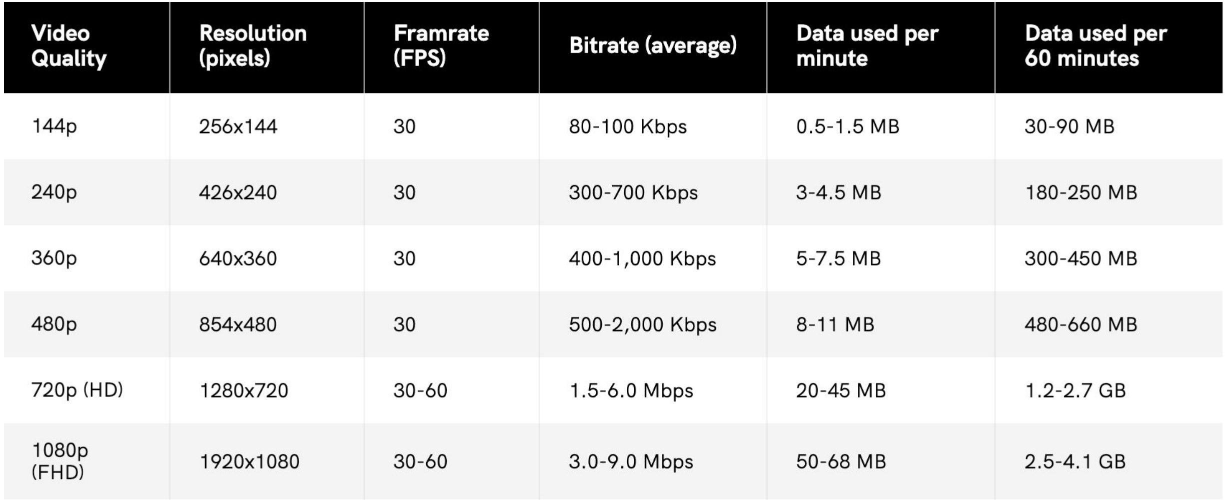
480p

360p

240p

144p

This selection only applies to the current video. For all videos, go to Settings > Video quality preferences.



Video Quality	Resolution (pixels)	Framrate (FPS)	Bitrate (average)	Data used per minute	Data used per 60 minutes
144p	256x144	30	80-100 Kbps	0.5-1.5 MB	30-90 MB
240p	426x240	30	300-700 Kbps	3-4.5 MB	180-250 MB
360p	640x360	30	400-1,000 Kbps	5-7.5 MB	300-450 MB
480p	854x480	30	500-2,000 Kbps	8-11 MB	480-660 MB
720p (HD)	1280x720	30-60	1.5-6.0 Mbps	20-45 MB	1.2-2.7 GB
1080p (FHD)	1920x1080	30-60	3.0-9.0 Mbps	50-68 MB	2.5-4.1 GB

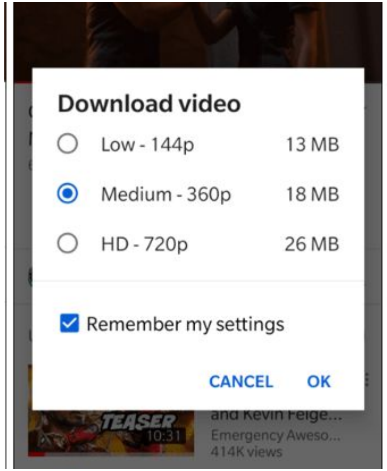
Downloading versus streaming

Difference

- Consume on the fly: streaming
- Consume later: downloading

Need for Streaming

- Don’t have to wait and load a 1GB video to start watching it!
- Just the first minute is enough to start watching.



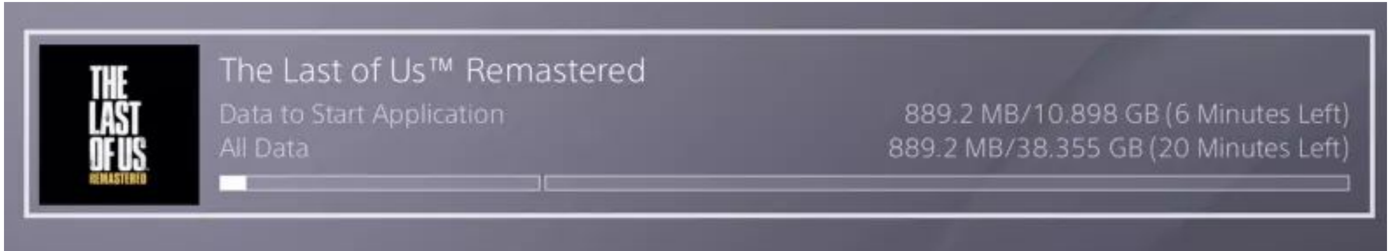
Download video

○ Low - 144p 13 MB

● Medium - 360p 18 MB

○ HD - 720p 26 MB

☑ Remember my settings



The Last of Us™ Remastered

Data to Start Application 889.2 MB/10.898 GB (6 Minutes Left)

All Data 889.2 MB/38.355 GB (20 Minutes Left)

Streaming videos - choosing a quality ahead of time

Assume 720p requires a 1Mbps connection

User's network < 1Mbps → buffering 😢

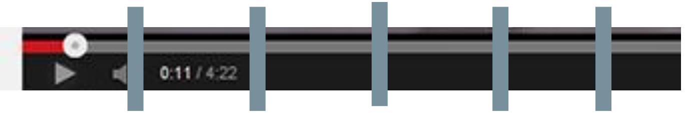
User's network == 1Mbps → perfect! 😁

User's network > 1Mbps → could use higher quality! 😒

DASH

Adapt video quality **dynamically** during **stream**

- Chunks of video



- Use a “playlist” of chunks

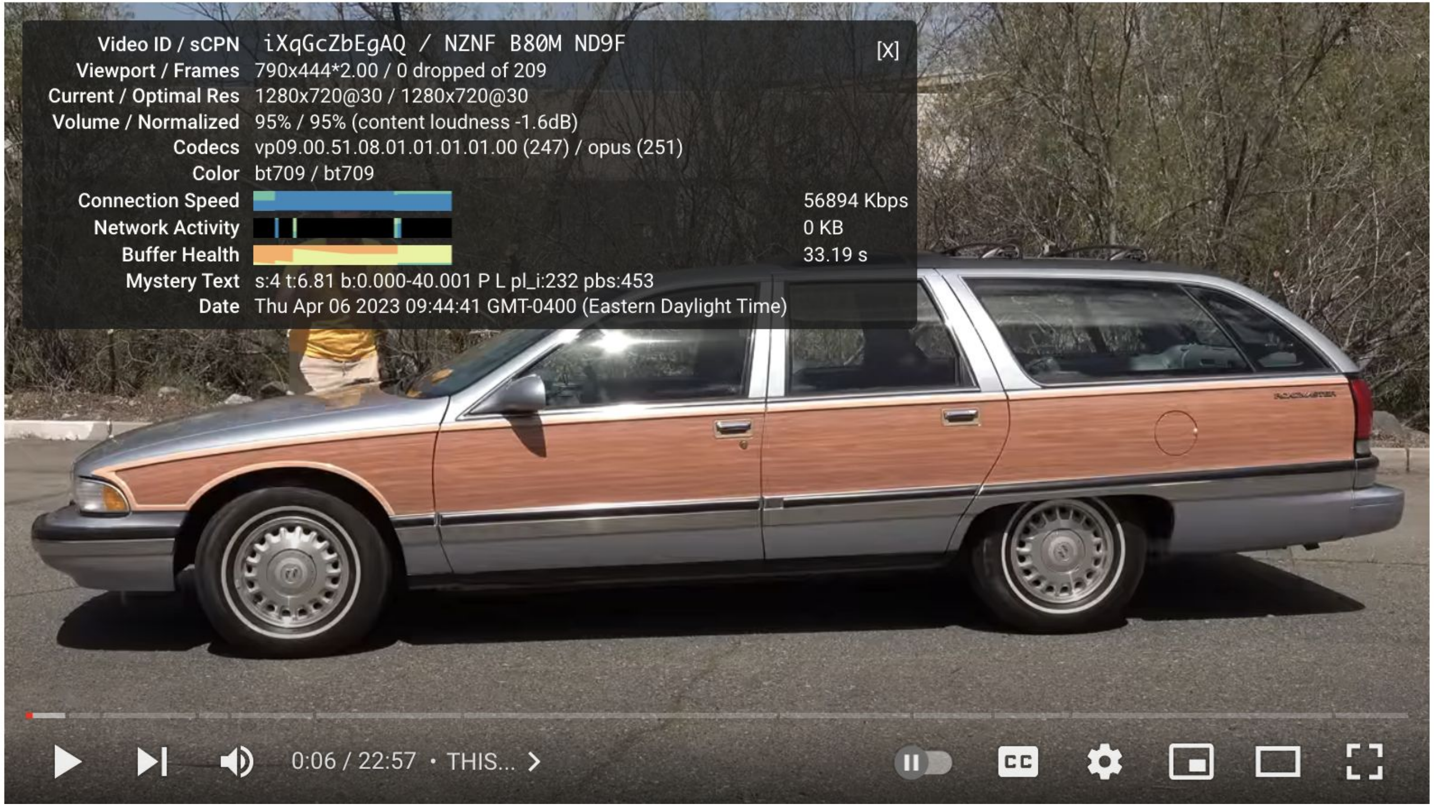
```
{
  "0": {
    "360p": "http://youtube.com/video1_360p_part0.mp4, "
     720p: "http://youtube.com/video1_720p_part0.mp4,
   },
  "1": {
    "360p": "http: //youtube.com/video1_360p_part1.mp4,
    "720p: "http: //youtube.com/video1_720p_part1.mp4,
  },
}
```

- Change quality for next chunk
- Signals to increase or decrease video quality?
    - Buffer occupancy, network throughput estimates etc.

Discussions

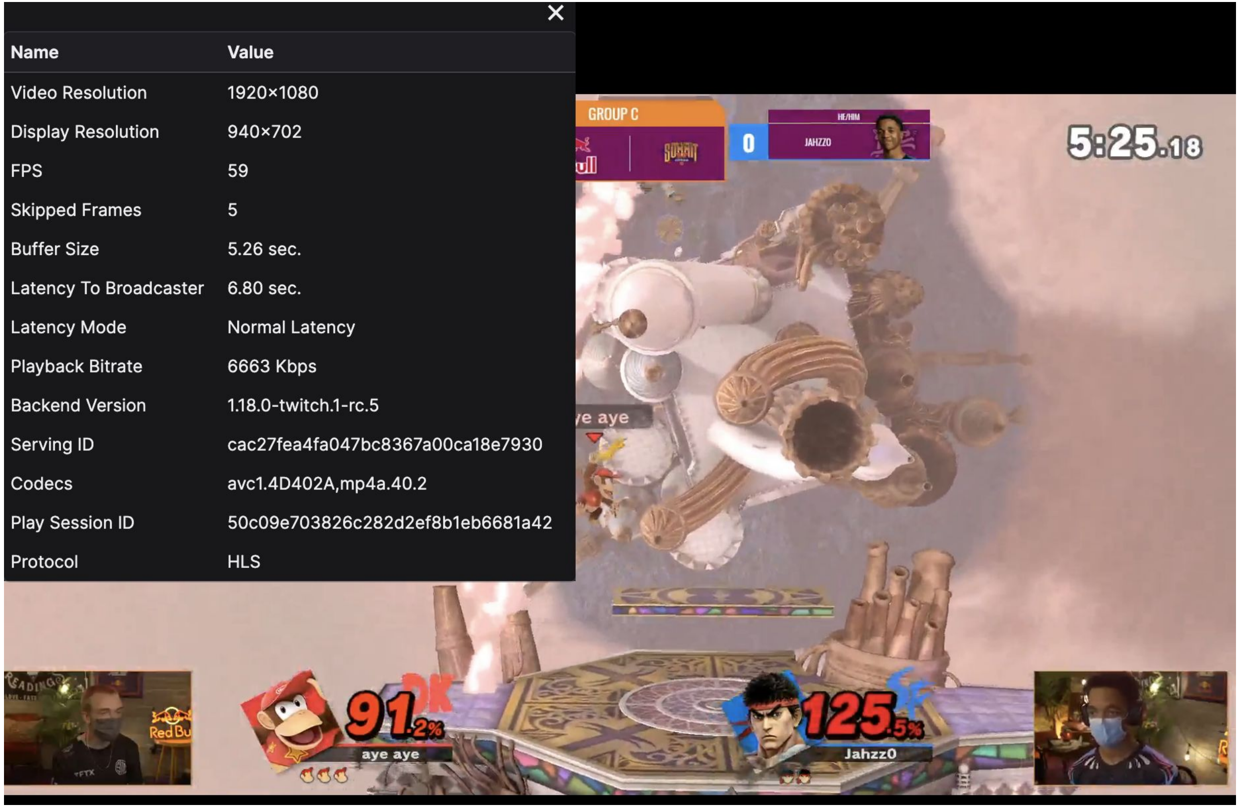
- Generate chunks ahead of time
    - “Processing video”
- CDNs to store these chunks
    - Preemptively push future chunks to cache
- How to choose chunk length?
    - Small duration → adaptation is dynamic, but more overhead
    - Large duration → less flexible and reactive, but lower overhead

YouTube - Stats for nerds



```
       Video ID / sCPN  iXqGcZbEgAQ / NZNF B80M ND9F
      Viewport / Frames 790x444*2.00 / 0 dropped of 209
  Current / Optimal Res 1280x720@30 / 1280x720@30
   Volume / Normalized  95% / 95% (content loudness -1.6dB)
                Codecs  vp09.00.51.08.01.01.01.01.00 (247) / opus (251)
                 Color  bt709 / bt709
      Connection Speed                                          56894 Kbps
      Network Activity                                          0 KB
         Buffer Health                                          33.19 s
          Mystery Text  s:4 t:6.81 b:0.000-40.001 P L pl_i:232 pbs:453
                  Date  Thu Apr 06 2023 09:44:41 GMT-0400 (Eastern Daylight Time)
```

Twitch - what’s different when it comes to live streams?



Name	Value
Video Resolution	1920×1080
Display Resolution	940×702
FPS	59
Skipped Frames	5
Buffer Size	5.26 sec.
Latency To Broadcaster	6.80 sec.
Latency Mode	Normal Latency
Playback Bitrate	6663 Kbps
Backend Version	1.18.0-twitch.1-rc.5
Serving ID	cac27fea4fa047bc8367a00ca18e7930
Codecs	avc1.4D402A,mp4a.40.2
Play Session ID	50c09e703826c282d2ef8b1eb6681a42
Protocol	HLS

Delay between streamer and chat.

Cannot create chunks.

Video conference?