Unit 2:
Video Streaming

Outline

1. Video Streaming
   a. Overview of HTTP-based video streaming
      i. Basics
      ii. Some history
   b. Quality of Experience
      i. Basics
      ii. QoS metrics impacting the video QoE
   c. Streaming strategies
      i. Strategies
      ii. Who chooses the strategy ?
      iii. Impact of streaming strategy on QoE
   d. HTTP Adaptive Streaming
      i. Basics
      ii. Workflow
      iii. Performance
      iv. Advanced Transport Options

Sources

- G. Urvoy-Keller and L. Sassatelli, *Video streaming*, lecture M2 Ubinet, UNS, 2014
- M. Siekkinen, *Video streaming*, lecture Aalto Univ., 2014
- H. Riiser, *Adaptive Bitrate Video Streaming over HTTP in Mobile Wireless Networks*, PhD thesis, Univ. of Oslo, 2013
- Z. Morley Mao, *Multimedia Networking*, lecture, lecture Univ. of Michigan
- *Video Communications and Video Streaming Over Internet: Issues and Solutions*, lecture Sharif Univ. of Technology
- A. C. Begen and T. Stockhammer, *HTTP Adaptive Streaming: Principles, Ongoing Research and Standards*, ICME 2013
- S. Akhshabi, S. Narayanaswamy, A. C. Begen, C. Dovrolis, *An experimental evaluation of rate-adaptive video players over HTTP*, Elsevier SP, Oct. 2011
- And others along the way…

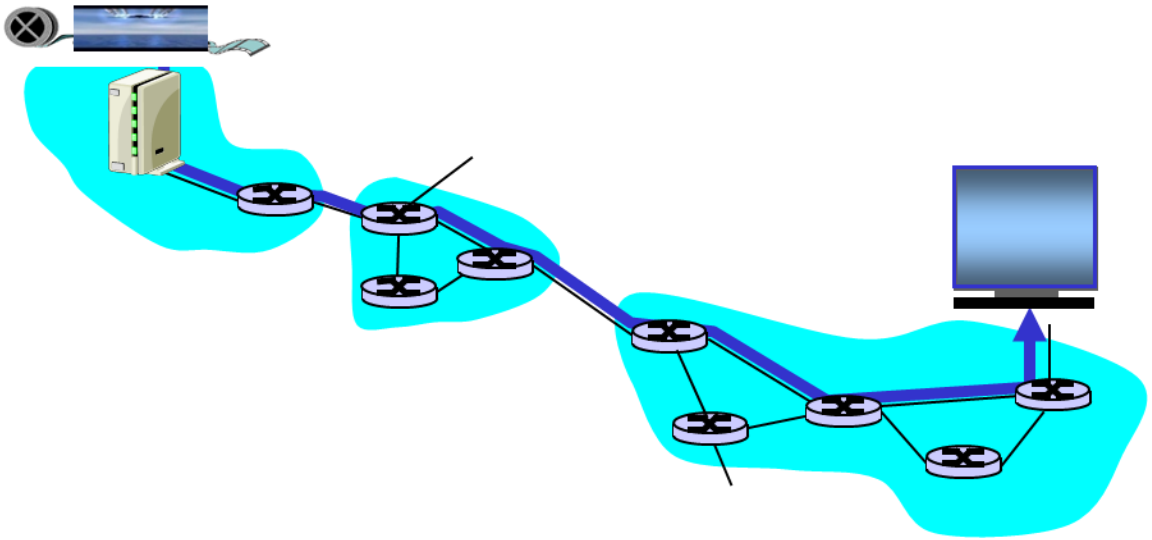
Overview of HTTP-based video streaming

1. **Basics**

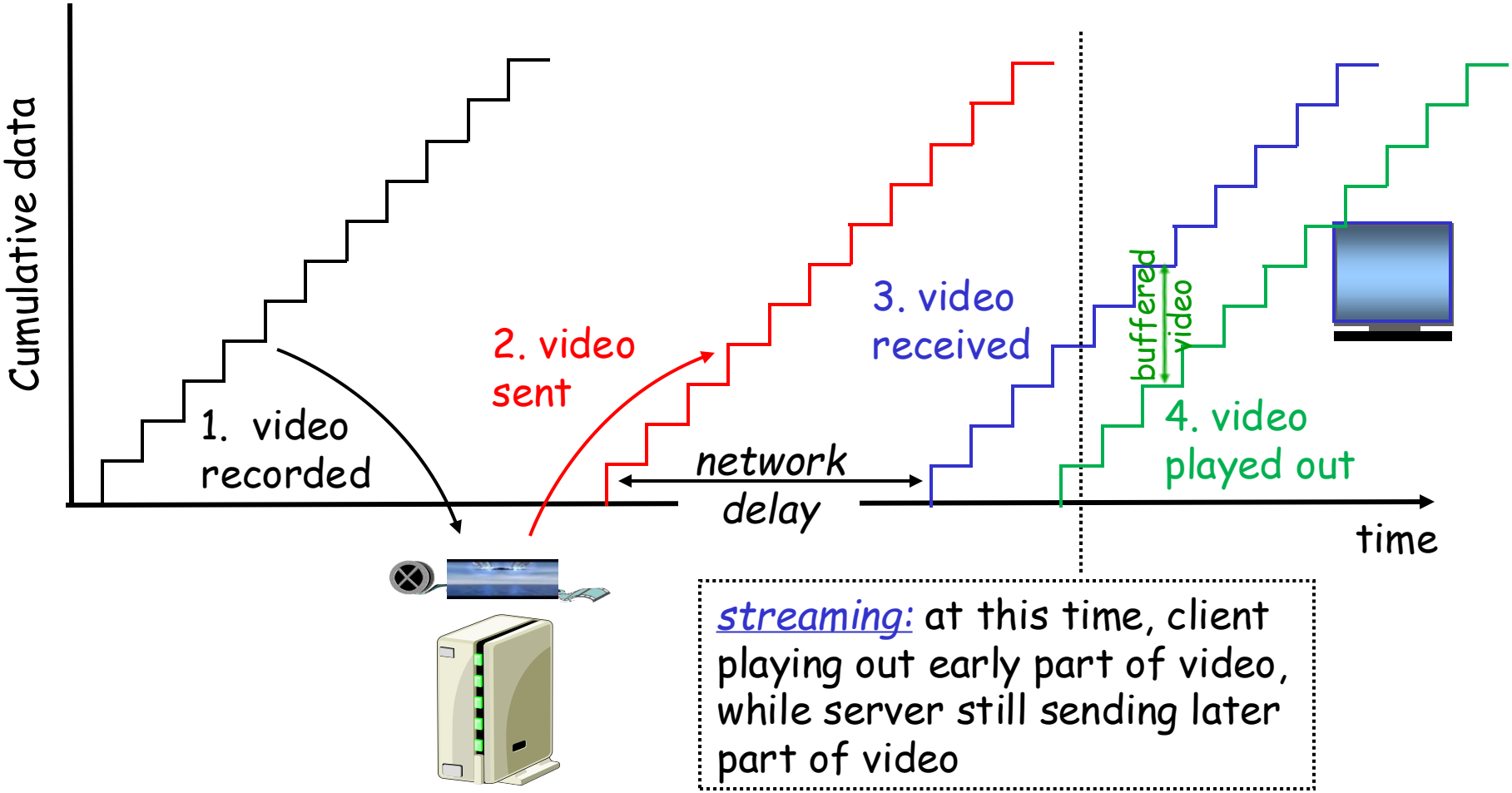
2. Some history

What is streaming?

- **Streaming**: the media stored at a source is transmitted to the client. The client playout starts before all the data has arrived.
- Timing constraint for still-to-be transmitted data: in time for playout
- Notation:
  - "video rate"="bitrate"=the rate at which the played video has been encoded
  - "bandwidth"="downloading rate"="sending rate"= the data rate achieved between the source and destination nodes



Streaming Stored Multimedia: What is it?



streaming: at this time, client playing out early part of video, while server still sending later part of video

Video Streaming Applications

1) Live, interactive video

   Video teleconferencing, video phones, etc.

2) Live, non-interactive video

   Course lectures, news, sporting events, conferences

3) Stored, non-interactive video

   Movies, distance learning, Web videos, etc.

- In this course, we focus on 3): streaming of stored data.

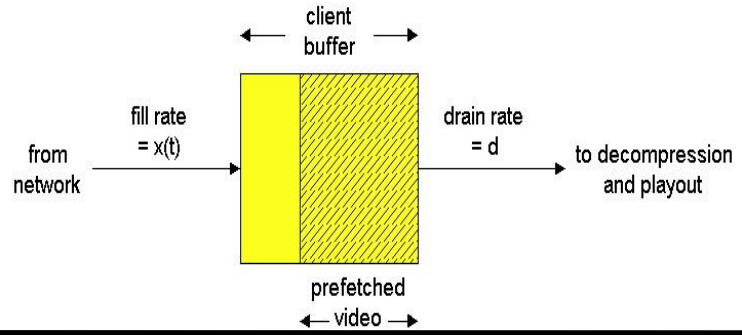
QoS constraints for video streaming

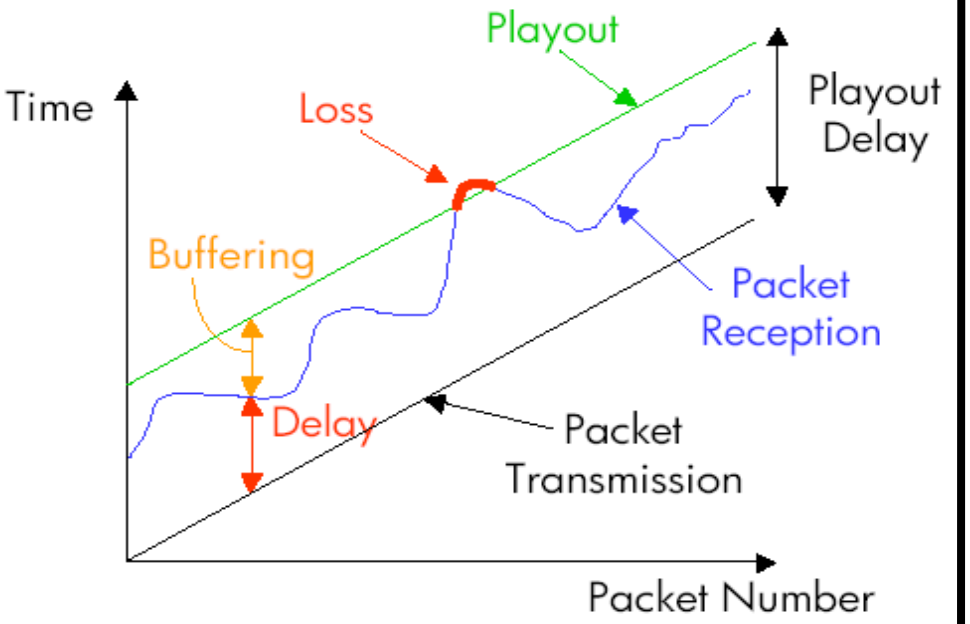
- Problem: Internet only offers best-effort service
- No guarantees on:
  - Bandwidth
    - Can not reserve bandwidth in Internet today
    - Available bandwidth is dynamic
    - If transmit faster than available bandwidth, then congestion occurs, packet loss, and drop in video quality
    - If transmit slower than available bandwidth, then sub-optimal video quality
    - Goal: Match video bit rate with available bandwidth
  - Loss rates
  - Delay jitter
- Specifically, these characteristics are **unknown** and **dynamic**
- Goal: Design a system to reliably deliver high-quality video over the Internet

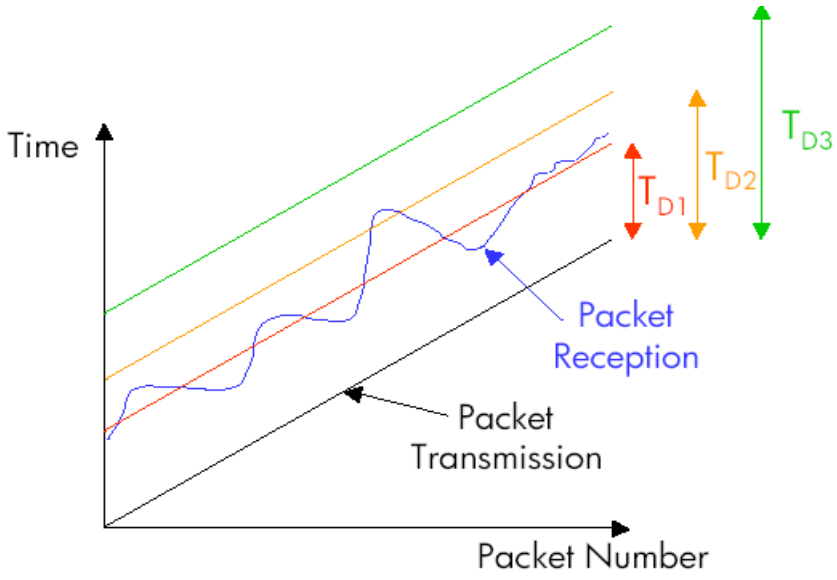
Overcoming Internet best-effort quality: Playout buffer

- Meant to compensate for bandwidth variations, delay jitter and enables retransmission of lost packets.
- Corresponds to adding an offset to the playout time of each packet
  - If (packet delay < offset) then OK
  - Buffer packet until its playout time
  - If (packet delay > offset) then problem
- Playout buffer typically 30 secs to several minutes of playback

Overcoming Internet best-effort quality: Playout buffer







Effect of different buffering delays

A few words about video compression

Examples:

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in the Internet, < 1 Mbps)

Recent eve\olution:

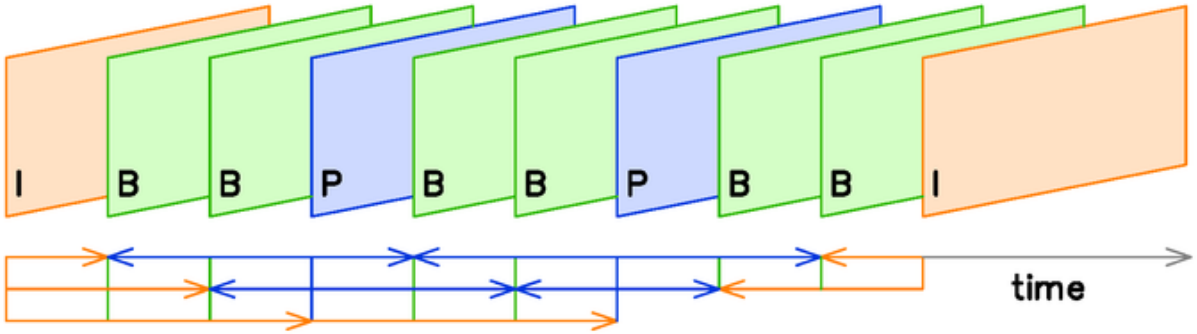
- **Ultra High Definition, a.k.a. 4K** (because it packs 4 times as many pixels as 1080p): in deployment in video distribution.
- Main content provider like Netflix and YouTube have started to offer it, which is increasing the demand for streaming.
- Required bandwidth for 4K videos: **15Mbps** -> the residential access is quickly growing -> **4K is putting a heavy strain on ISPs**

Extensions:

- Layered (scalable) video
  - Adapt layers to available bandwidth
  - Multiple Description Codes (MDC)
  - Scalable Video Coding (SVC): Annex G extension of the H.264/MPEG-4 AVC video compression standard

A few words about video compression

- **Advanced Video Coding (AVC), H.264/MPEG-4 AVC**
  - Basic principle: to compare different parts of a frame of video to find areas that are redundant, both *within a single frame* as well as *subsequent frames.*
  - The redundant areas are then replaced ("predicted") with a short description instead of the original pixels.
- Three major picture (or frame) types:
  - **I-frame (Intra-coded picture):** The complete image, don't require other video frames to decode, but they are the heaviest ones.
  - **P-frame (Predicted picture):** It holds *only the changes in the image from the previous I frame*. Then we need data from previous frames to decompress
  - **B-frame (Bidirectional predicted picture):** It holds only the differences between the current frame and both the preceding and following P frames to get the highest amount of data compression.



A few words about video compression

- **Scalable Video Coding (SVC),** Annex G extension of the H.264/MPEG-4 AVC
  - Basic principle: To allow dropping packets (degrading quality) to reduce the bandwidth required
  - To do that, B frames are encoded hierarchically thanks to a hierarchy of temporal enhancement layers.
  - At least all the frames of the base layer are required. Then, each extra layer "enhances" the quality.

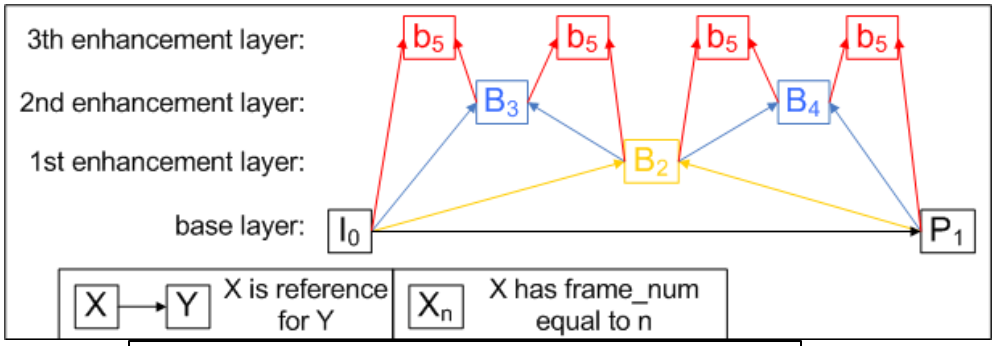


Image from: http://multimedialab.elis.ugent.be/bflavor/samples.html

- **High Efficiency Video Coding (HEVC),** H.265/MPEG-H Part 2
  - Likely successor to the widely used AVC (H.264).
  - In comparison to AVC, HEVC offers about double the data compression ratio at the same level of video quality thanks to improved predictions and larger pattern comparisons
  - It supports resolutions up to 8192×4320, including 8K UHD.

Overview of HTTP-based video streaming

1. Basics

2. **Some history**

A brief history of video streaming

- Used to be admitted that real-time video over a best-effort network like the Internet would have to be streamed using a datagram protocol: giving the streaming application packet-level control
- -> this meant in practice that it should be carried by UDP, not TCP, e.g., *RTP (Real-time Transport Protocol) over UDP*:
  - RTP streaming systems can have full control over packet retransmission -> can trade packet loss for delay (e.g., audio and video conferencing).
    - Problem with RTP: new media codecs cannot easily be supported.
  - RTP requires out-of-band signaling (RTSP-Real Time Streaming Protocol, and SIP, Session Initiation Protocol)

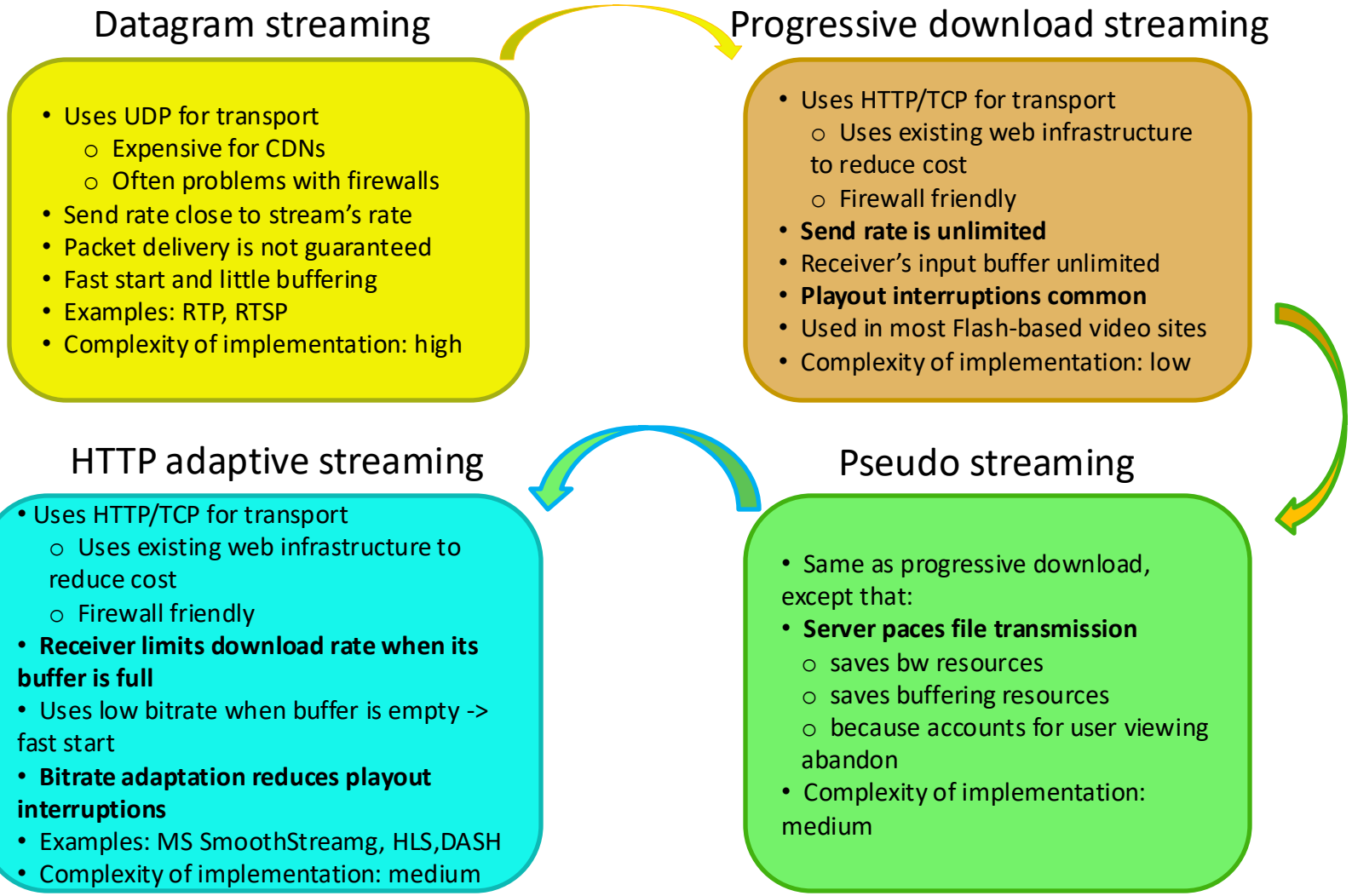
A brief history of video streaming

- In addition to this, protocols based on datagram streaming suffer from:
  - Packet-level control -> very complicated implementation as needs to deal with flow and congestion control, packet loss, out-of-order delivery, etc.
  - Firewalls and NAT routers frequently cause problems with UDP
  - Higher cost of the infrastructure as Content Delivery Networks (CDNs) require specialized solutions for caching and load balancing (almost all deployed infrastructure optimizations target HTTP because of its massive popularity)
- => most of the industry adopted progressive download streaming using HTTP
- Its simplicity has made it the streaming protocol most commonly used today (e.g., by YouTube).

Whereby HTTP-based streaming

- With this approach, the client simply downloads a media stream as a file in a normal media container format such as MP4, and plays back the video while it is downloading.
- Benefits:
  - firewall traversing
  - all CDNs support it (can automatically take advantage of transparent web caching)
- Downsides:
  - playout interruptions are more likely to occur
  - larger buffer is required (limiting progressive streaming's suitability for real-time communication)
  - multicast is not an option (no longer considered a big loss, since there is no widely available multicast infrastructure)
- -> The biggest arguments for datagram protocols for non-interactive streaming is now relatively irrelevant

Evolution from datagram streaming to adaptive HTTP streaming



Quality of Experience

1. **Basics**

2. QoS metrics impacting the video QoE

QoE Basics

- Traditionally, Quality of Service (QoS) metrics were used to estimate quality in networks:
  - *throughput, delay, jitter*
  - *Easy to determine: simple measurements on the network*
- But QoS metrics are not really measuring the quality that the client ***perceives***, i.e., the Quality of Experience (QoE):
  - *"QoE is the degree of delight or annoyance of the user of an application or service" [*Video Quality Expert Group (VQEG)*]*
- QoE is subjective & based on complex functions of the QoS metrics
  1. video quality
  2. variations of the video quality
  3. ratio of rebufferings
  4. startup delay

QoE Basics

- Many metrics to capture quality
  - Traditional QoS metrics being replaced by video delivery based metrics (QoE)
- QoE is crucial for video providers and telcos:
  - Direct impact on user engagement → ***revenue (through ads etc.)***
  - Currently lots of efforts to model and improve it
- Different stakeholders
  - ISP: The one providing client's access to Internet
    - often the one customers blame for poor QoE
  - Video service provider: YouTube, Netflix, …
  - CDN operator: YouTube, Akamai, Limelight networks, Level 3, …
  - Client: you, the one who matters
  - Device manufacturer: hardware performance
  - Client software provider: OS and player software

QoE Basics

- What determines QoE, i.e. who is in control?
  - Jointly the behavior of all the above stakeholders
  - Outcome of a rather complex process
- Many factors influence QoE
  - Different stakeholders
    - Video CDN operations are a crucial one
  - Streaming strategy is another important one
    - *We will see that in more detail in next section*

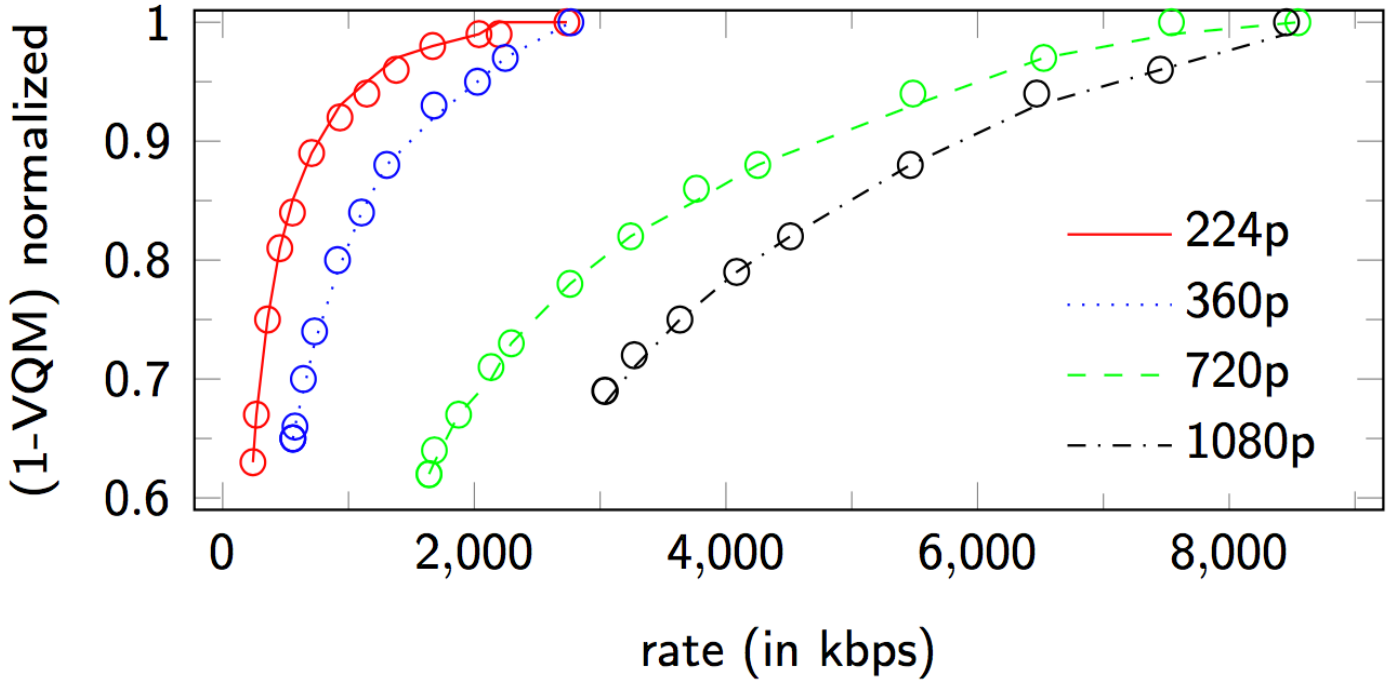
Quality of Experience

1. Basics

2. **QoS metrics impacting the video QoE**

QoS metrics impacting the video QoE

- QoE definition as weighted sum of QoS metrics:
  - QoE = Video quality + video quality switches + rebuffering time (# rebuffering events) + startup delay
- Video quality metrics:
  - Based on video resolution, frame rate, target encoding rate,….



QoS metrics impacting the video QoE

- Video quality classification :
  - Objective video quality metrics:
    - Typically *Full Reference Methods (FR):* They compute the quality difference by comparing the original video signal against a reference video
    - *Examples:*
      - *Peak Signal-to-Noise Ratio (PSNR):* every pixel from the source is compared against the corresponding pixel at the received video, frame by frame
      - *Video Quality Metric VQM*: More complex calculations, standardized in ITU-Recommendations
  - Subjective video quality metrics:
    - They aim to automatically estimate the average viewer's opinion on the video quality
    - *Mean Opinion Score (MOS):* Video sequences shown to a group of viewers and their opinion is recorded and averaged
- Video quality switches (in the case of adaptive streaming)
  - Rate and amplitude of quality switches

QoS metrics impacting the video QoE

- Buffering events
  - If playback buffer runs dry, playback pauses -> annoys the user
  - Happens if download rate is smaller than encoding rate for too long
  - Initial buffering attempts to avoid such events
  - Studies suggest this is the most important reason for abandoning viewing
- Initial startup delay (a.k.a. joining time)
  - Time it takes for playback to begin after clicking play
  - Not as fatal as buffering events for audience retention

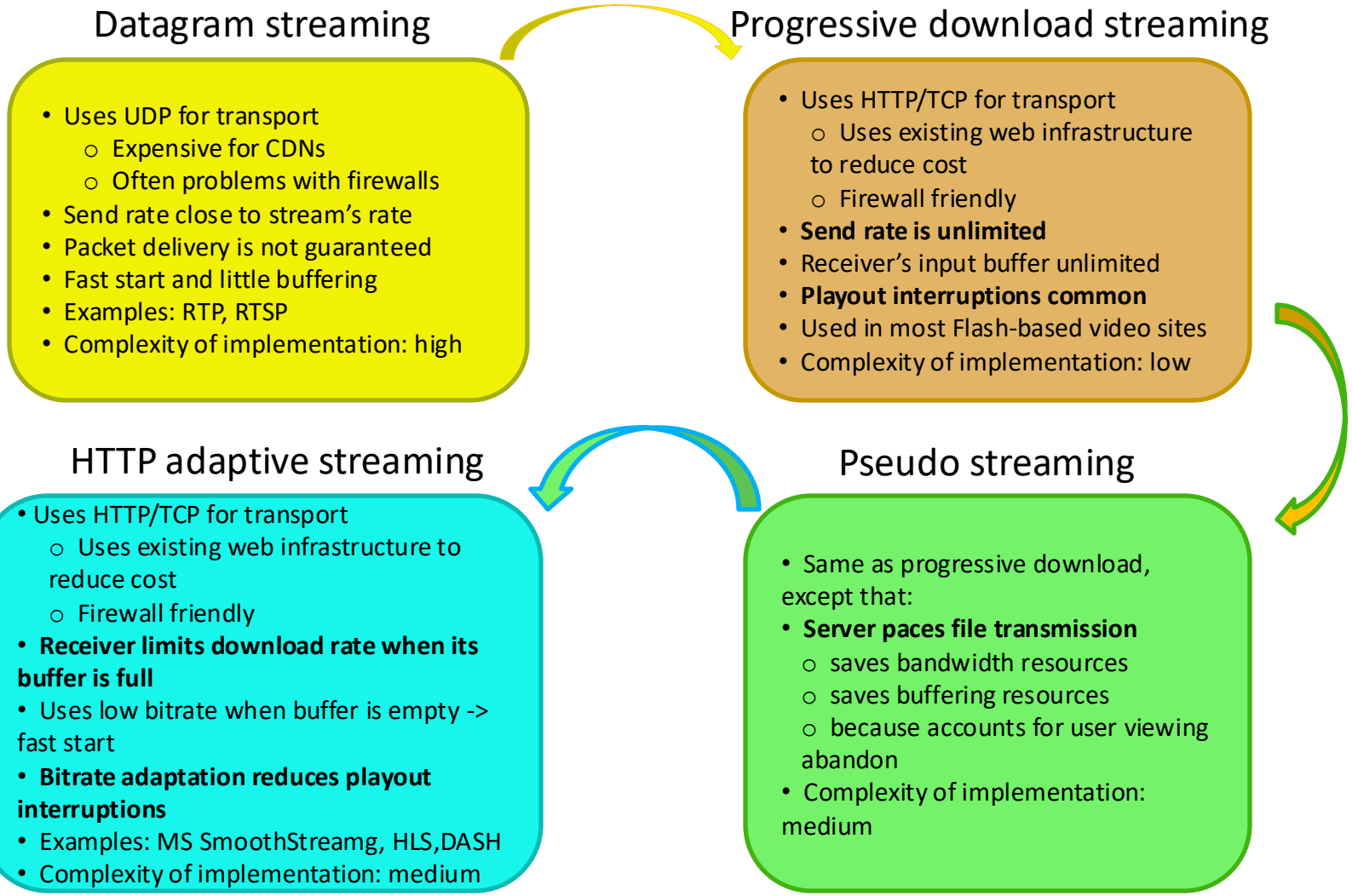
Delivery related QoE metrics

- Several factors influence the probability of experiencing buffering event
  - Amount of initially buffered content
  - Instantaneous TCP bulk transfer capacity from server to client
  - *Content download strategy, i.e., streaming strategy*

Streaming strategies

1. **Strategies**
2. Who chooses the strategy ?
3. Impact of streaming strategy on QoE

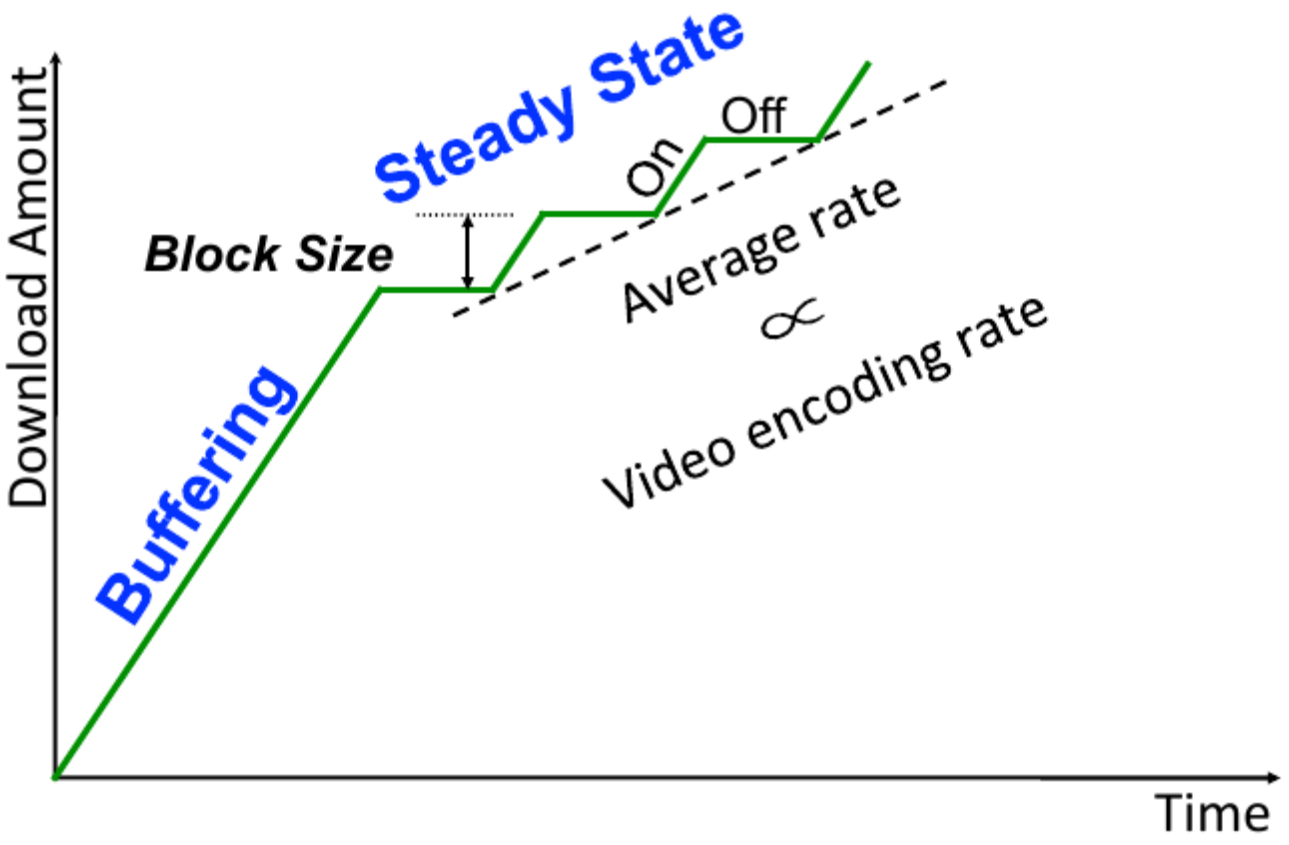
Evolution from datagram streaming to adaptive HTTP streaming



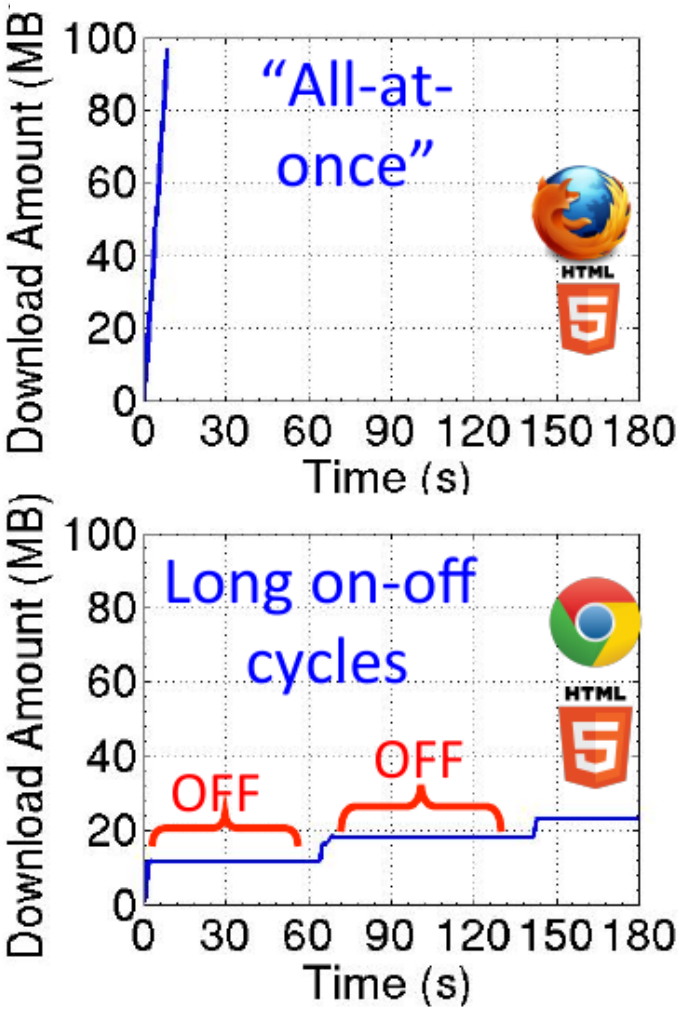
Streaming strategy

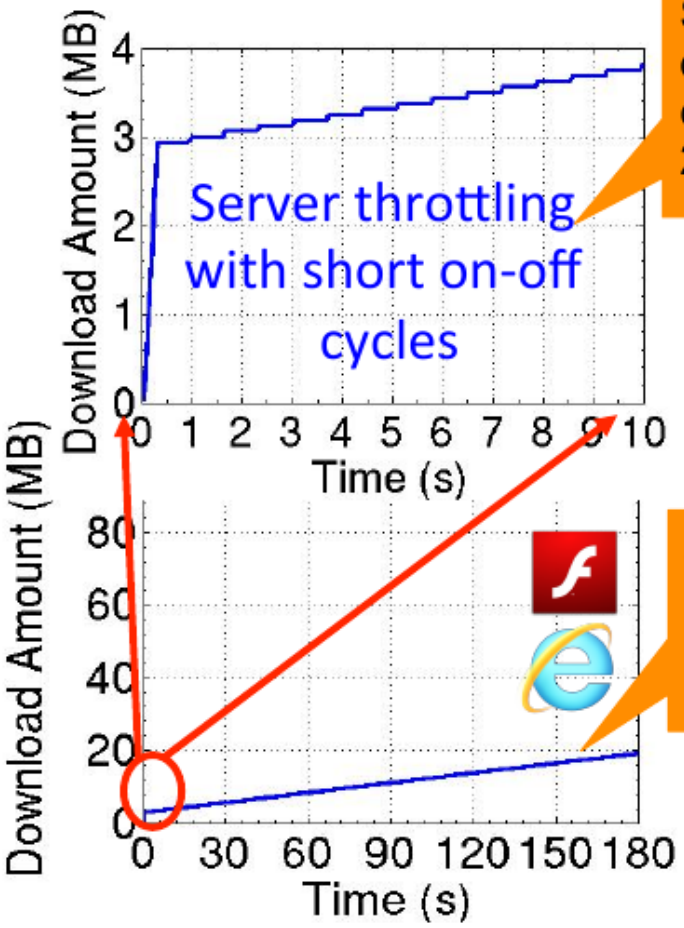
- Turns out that a number of different strategies are used
  - Interplay between server and client application
- We'll borrow some results from earlier studies
  - *A. Rao et al: Network Characteristics of Video Streaming Traffic. CoNEXT 2011.*
  - *M. Hoque et al: Mobile Multimedia Streaming Techniques: QoE and Energy Saving Perspective. Pervasive and Mobile Computing 2014.*

Generic behavior: « pseudo streaming »



Identified streaming strategies



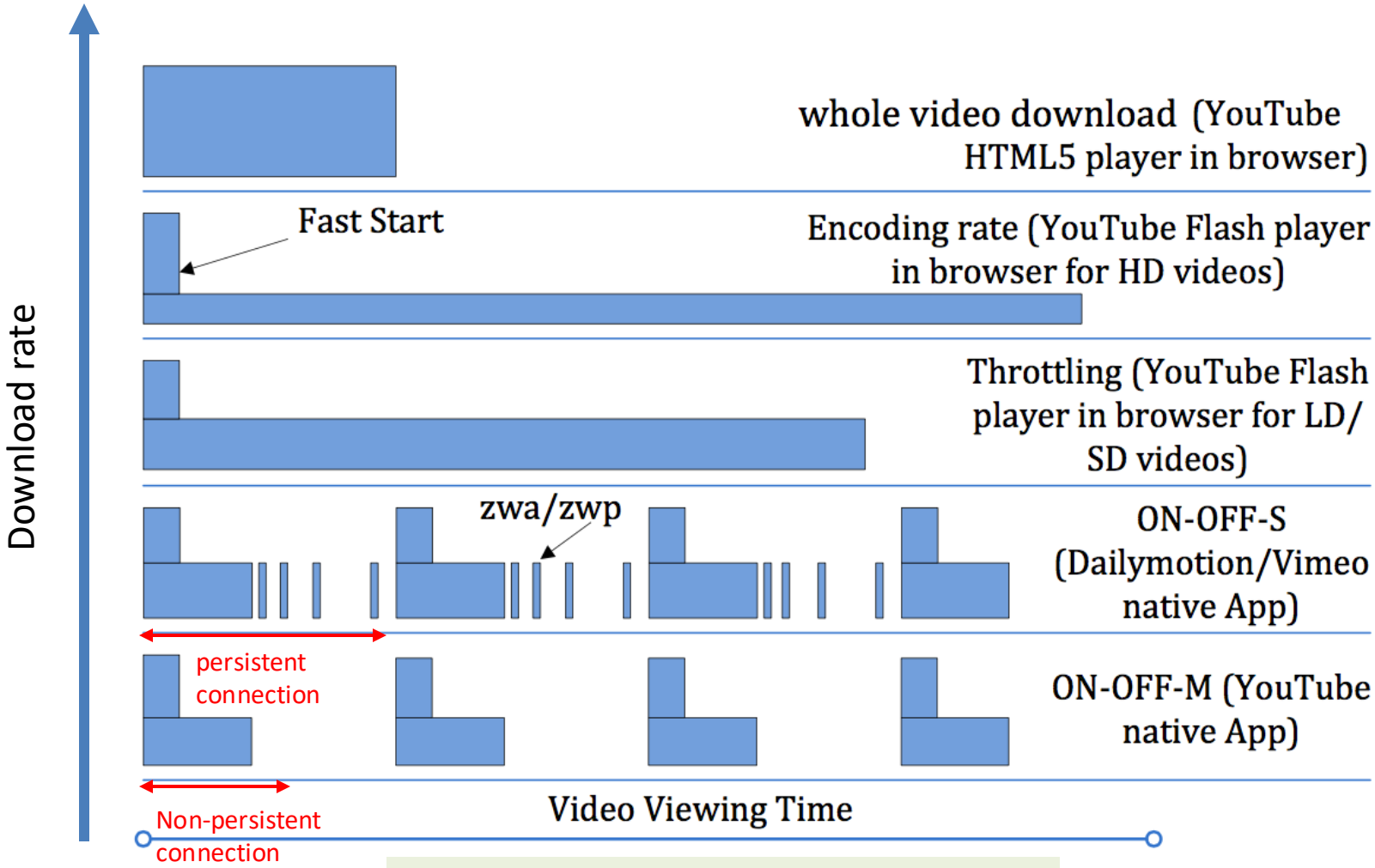


Server enforces a download rate control (e.g: 1.5x or 2x encoding rate)

Also *encoding rate* streaming possible: dl at encoding rate in steady state phase

Streaming strategies vastly different

Identified streaming strategies



Streaming strategies vastly different

Siekkinen, Matti, Mohammad Ashraful Hoque, and Jukka K. Nurminen. "Using Viewing Statistics to Control Energy and Traffic Overhead in Mobile Video Streaming." (2014).

Streaming strategies

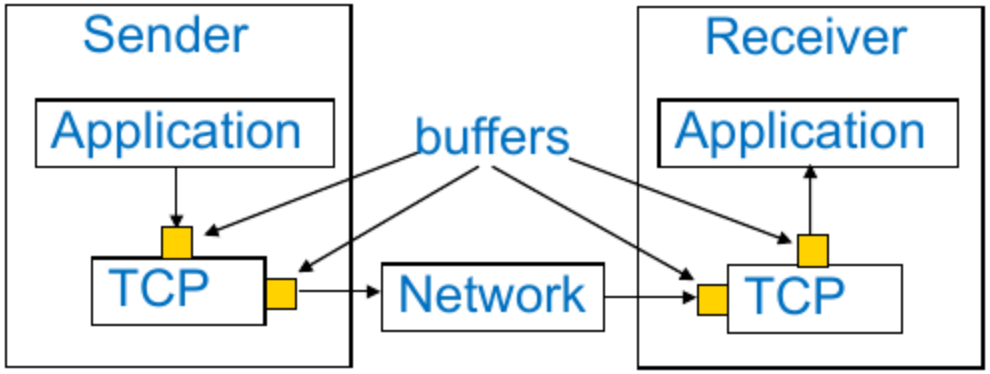
1. Strategies

2. **Who chooses the strategy ?**

3. Impact of streaming strategy on QoE

Who chooses the strategy?

- Let's quickly review the interplay between applications and TCP



Who chooses the strategy?

- All-at-once and server throttling are **enforced by server**
  - But all-at-once may be explicitly requested by client application (HTTP request parameter)
- Long ON-OFF cycles are (usually) **caused by client application**
  - Client application periodically stops reading from TCP socket
  - TCP receive buffer fills up -> TCP flow control activates and forbids server side TCP to send more packets
  - Transfer paused until application reads again from socket
- Encoding rate streaming seems to be **unintentional**
  - Client's application buffer fills up -> receive TCP buffer fills up -> TCP flow control activates
  - Client application reads data from socket at encoding rate -> new data transferred at this rate

Streaming strategies

1. Strategies

2. Who chooses the strategy ?

3. **Impact of streaming strategy on QoE**

Impact of streaming strategy on QoE

- Amount of buffered data directly relates to the probability of a buffering event
  - Level of protection against transient network issues
- But providers and clients need to consider other issues too
  - Unnecessarily served content upon abandoning: **half videos are abandoned before half viewing**
  - **->** Bandwidth waste
  - Capped data plans
  - Energy efficiency

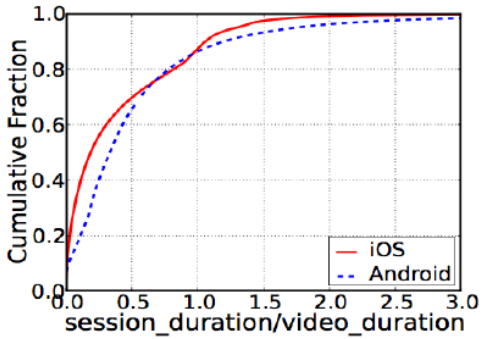
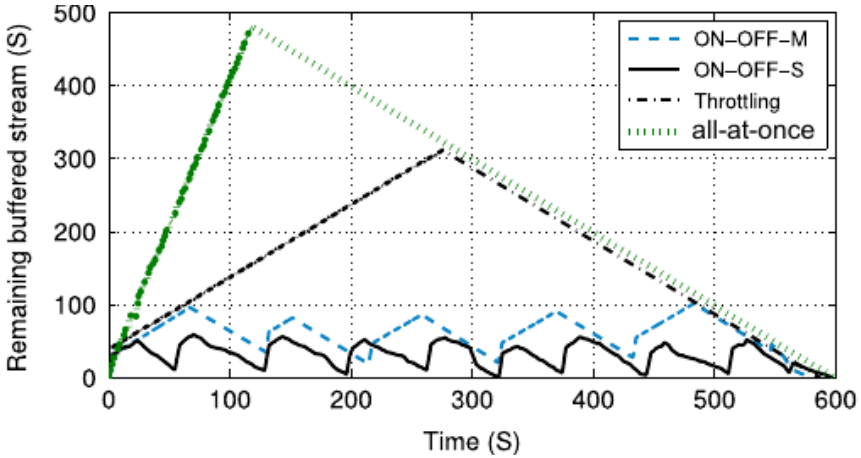


Fig. 1: Ratio Between Session Duration and Video Duration (CDF)



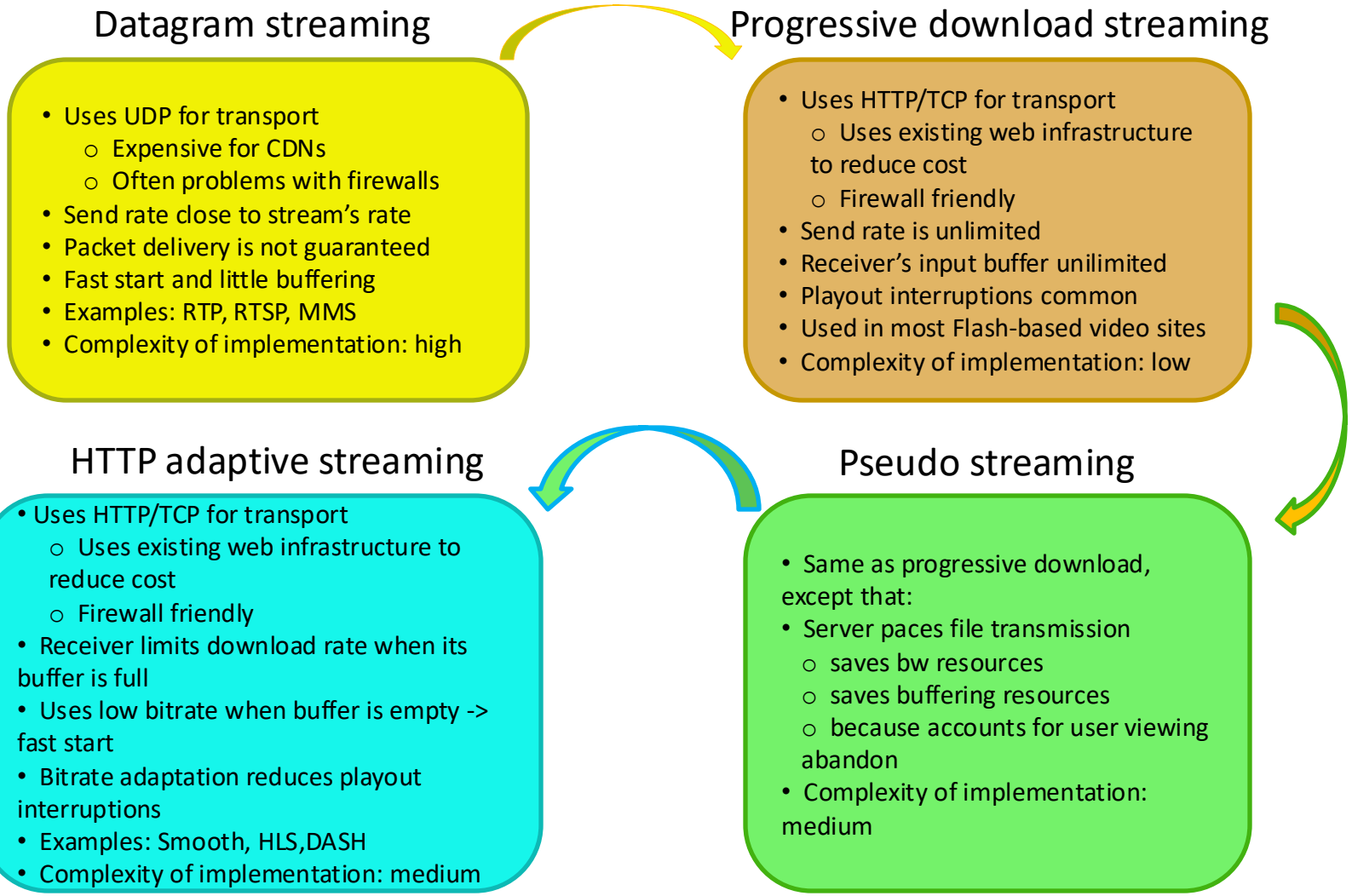
Strategy	No ON-OFF	Long ON-OFF	Short ON-OFF
Engineering Complexity	Not required	Explicit support at Application Layer	
Receive buffer occupancy	Large	Moderate	Small
Unused bytes on user interruption	Large amount	Moderate amount	Small amount

Y. Liu, F. Li, L. Guo, B. Shen and S. Chen, "A Comparative Study of Android and iOS for Accessing Internet Streaming Services," in PAM, Hong Kong, 2013.

HTTP Adaptive Streaming

1. **Basics**
2. Workflow
3. Performance
4. Advanced Transport Options

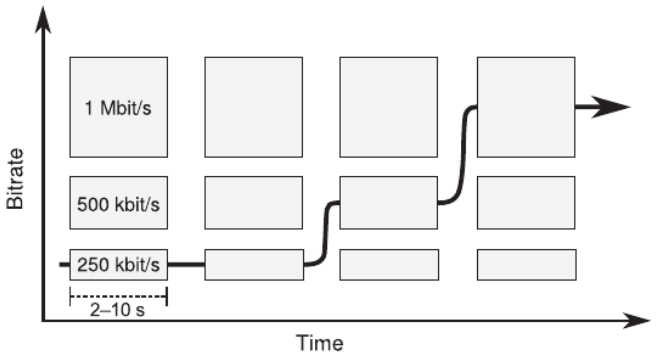
Evolution from datagram streaming to adaptive HTTP streaming



HTTP Adaptive Streaming: Motivation

- Idea: make the requested bitrate of the video fit the (possibly varying) network resources
- For strained networks: wireless (4G/5G), and wired networks (4K exacerbates) – shared networks like HFC or DSL backhaul are likely to face bandwidth issues

- "available resources": usually network bandwidth (can also possibly CPU load, battery capacity, and screen size)
- client-side adaptation by far the most popular in recent systems: all the information that is relevant when choosing which quality to use is available to the client
- **HAS offers a solution for the most significant problem with streaming over HTTP: fluctuating bandwidth**
- -> Being able to switch seamlessly to a stream with a lower bitrate whenever the bw or buffer is too low makes HTTP much more usable for video streaming, especially on mobile devices.
- -> HAS bound to generalize



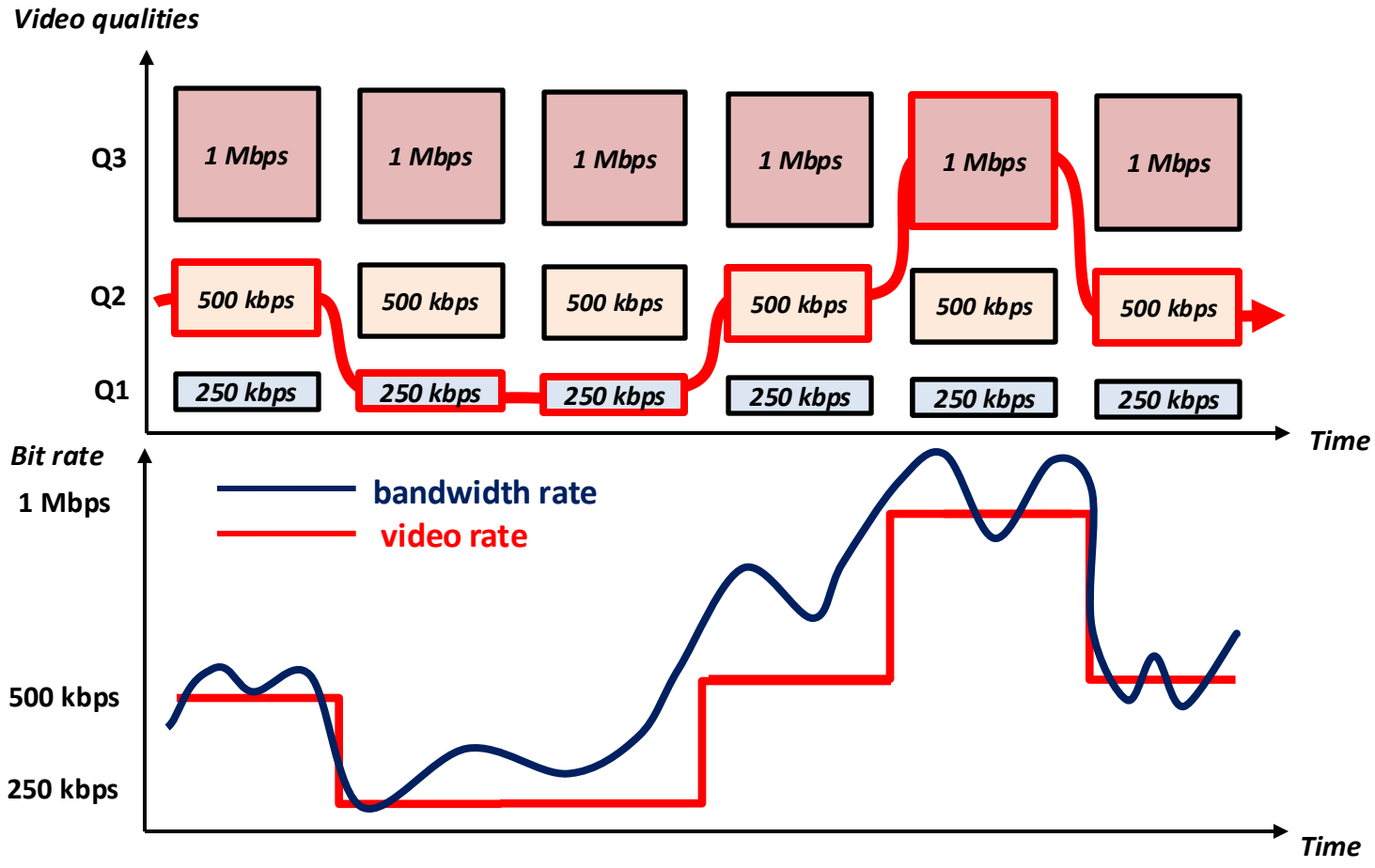
HAS: principle

- A stream is split into a sequence of segments downloaded individually, instead of performing one large file download per stream.
- Each segment is typically 2–10 seconds of the stream
- The segment data can be either a multiplexing container format that mixes data from several tracks (audio, video, subtitles, etc.), or it can contain data from just a single track, requiring the receiver to download and process segments from several tracks in parallel.
- The video track is available in multiple different bitrates, each representing a different quality level. The quality can only change on segment boundaries.

- HAS flavors: DASH, Microsoft Smooth Streaming, HLS,…
- -> DASH and HLS now
- How the client chooses the bitrate to request: up to its software, no need to make it uniform.

HAS: principle

- A video is composed of **segments** (2–10 s)
- Each segment is encoded in different qualities (bit rates, resolutions)



HAS: principle

MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH): International standardized solution of HAS.

Client-side adaptation policies from Bitmovin LibDASH: *[https://github.com/bitmovin/libdash]*

- **Libdash**: **official reference software of the ISO/IEC MPEG-DASH standard** and is an open-source library that provides an object orient (OO) interface to the MPEG-DASH standard, developed by Bitmovin.
- **Bitmovin:** multimedia technology company which provides services that transcode digital video and audio to streaming formats using cloud computing, and streaming media players.

HAS: principle

Policies:

1. **Rate-based (RB):**
   - Simply selecting the highest possible video representation (visual quality) taking as reference the measured speed from previous chunk download.
2. **Buffer-based (BB):**
   - Based on different threshold level monitoring the buffer.
   - The more the buffer is filled the higher is the quality selected.
   - The quality switch is done one step for each new request, in other words is not possible to jump form the lowest to the highest representation in one step
3. **Buffer and rate-based (BRB):**
   - Mix of the previous two.
   - Based on the concept of RB but the previous chunk download speed is weighted using a factor that depends on the amount of filled buffer.
   - If the buffer is depleting, the previous chunk download speed will be considered to be less than the measured one, then a lower representation is selected.

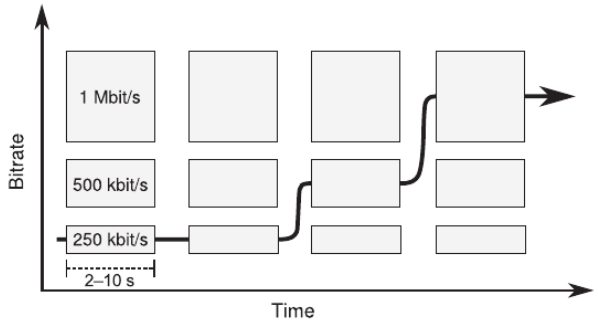
HTTP Adaptive Streaming

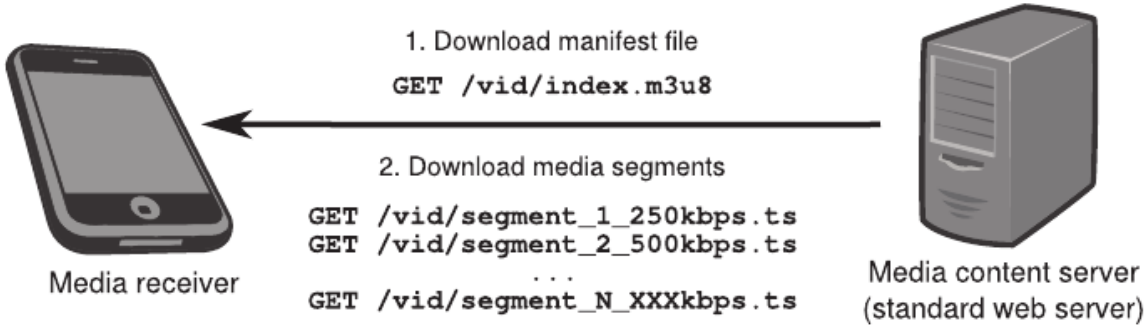
1. Basics
2. **Workflow**
3. Performance
4. Advanced Transport Options

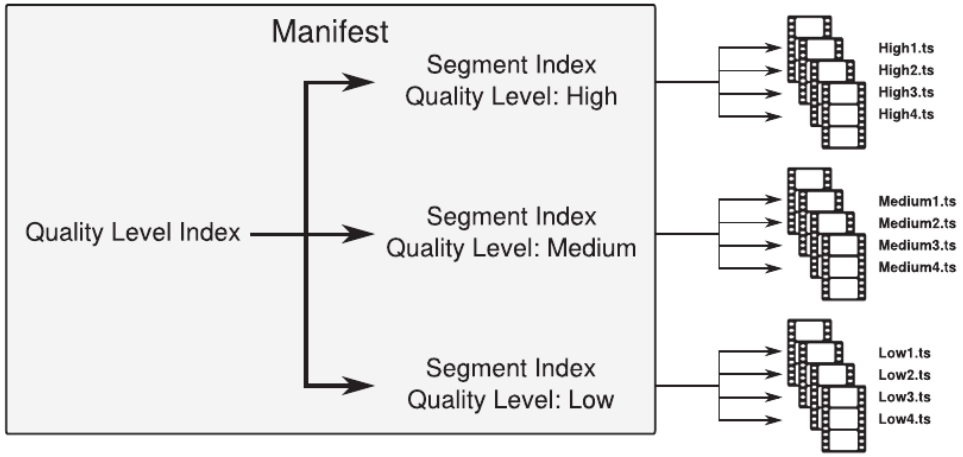
HAS: manifest file

- Each segment is separately downloadable using its own URL.
- Standard HTTP GET requests are sent for every segment, and the URLs for these requests contain what is needed to uniquely identify the segment to be downloaded (timestamp of the first video frame in the segment, bitrate values, …).
- To reduce the entire stream to a single URL, most adaptive formats use a manifest file to describe the stream's structure. The manifest file includes:
  - General info (total stream duration, encryption info, if it is VOD or Live,…)
  - Which types of streams are available (e.g., audio, video, subtitles, etc.)
  - URLs for each media segment, and info about the segments' durations and start times
  - Quality levels are available for each stream
- -> a media player needs only the URL to the manifest file to start playing video, because all the segment URLs will be known after it has downloaded and parsed the manifest

Workflow of HAS and manifest file







HTTP Adaptive Streaming

1. Basics
2. Workflow
3. **Performance**
4. Advanced Transport Options

Performance of HAS

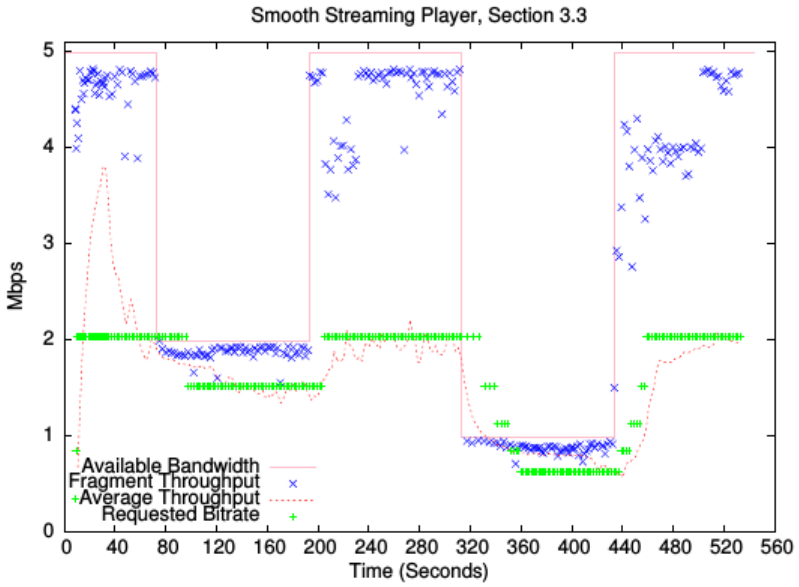


Figure 3: Per-fragment throughput, average TCP throughput and the requested bitrate for the video traffic under persistent avail-bw variations. Playback starts at around t=10 s, almost 3 s after the user clicked PLAY.

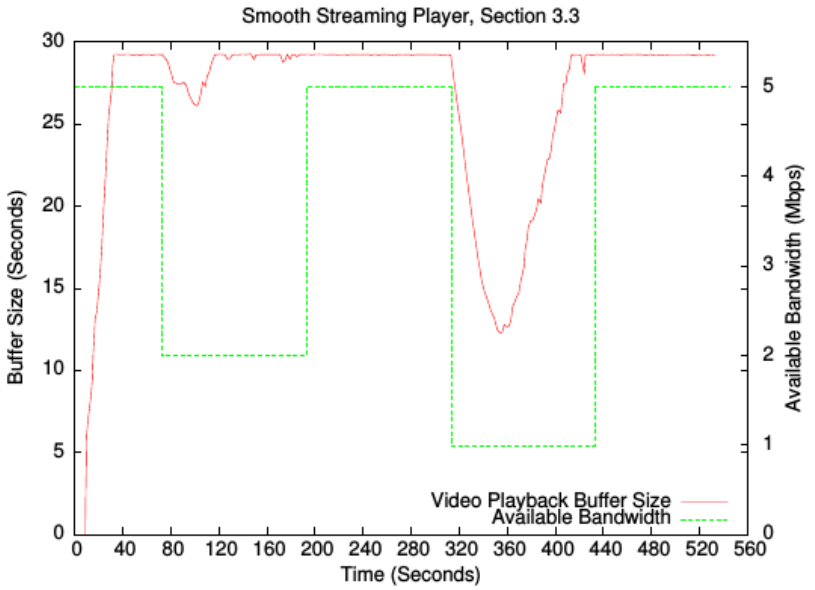


Figure 4: Video playback buffer size in seconds under persistent avail-bw variations.

Performance of HAS

- 1) 1st first player uses the highest profile (P2.75)
- 2) The 2 players could have shared the 4 Mbps bottleneck by switching to P1.52, however, they do not: player 2 oscillates between lower profiles.
- 3) The oscillations continue for both players.
- 4) Stable because lowest rate
- 5) The two players start oscillating in a synchronized manner.

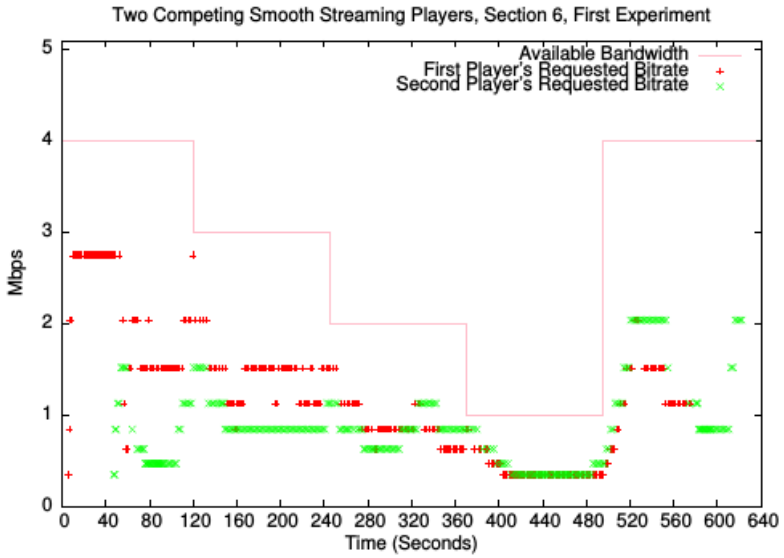
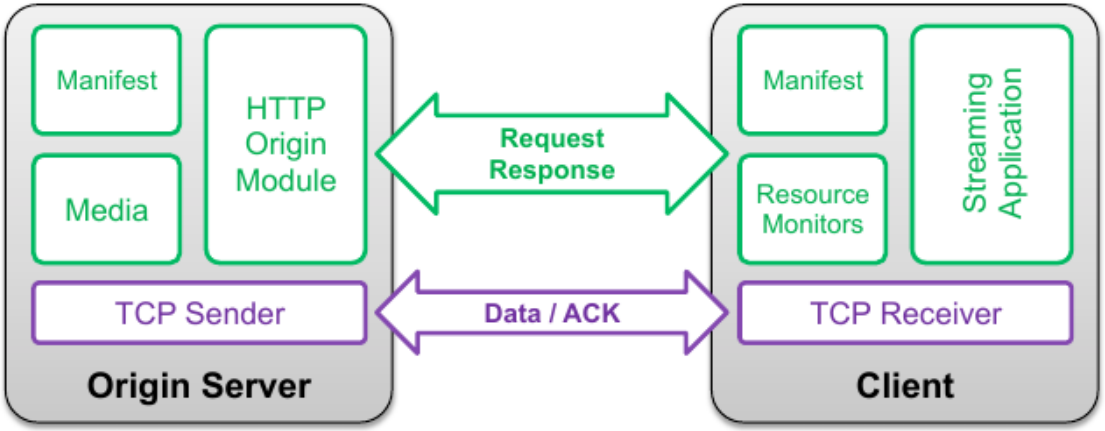


Figure 16: Two Smooth Streaming players compete for avail-bw. The players start the playback at around t=7 s and t=57 s, respectively.

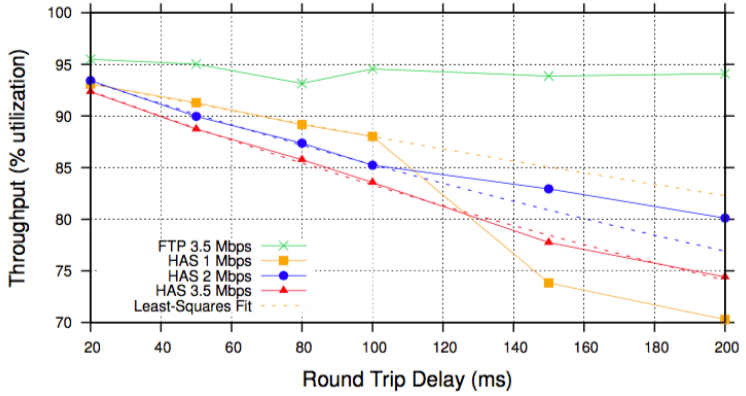
Performance of HAS



- The interplay between TCP loop control and HAS loop control is intricate, specifically when the RDA bases its decision *solely on the measured downloading rate (rate-based policy)*.
  - Can generate inefficiency and unfairness
- New players (Netflix) are increasingly using the buffer state info *(buffer-based policy).*

Performance of HAS

- **TCP–HAS Interplay**
  - Long video downloads using TCP implies big `cwnd`
    - Long-lived TCP flows (like FTP) → Fast retransmit / Fast recovery more used → no dramatic `cwnd` reductions → packet loss not very detrimental
    - Short-lived TCP flows (like HAS) → On-off pattern (sequential HTTP requests every few seconds) → Slow Start more used, specially at start/end of a chunk transmission → huge `cwnd` reductions very often → packet loss very detrimental
  - This difference between HAS and FTP throughput represents the penalty for downloading a video via several shorter requests instead fewer longer ones.
  - *Then: HAS ratio of goodput-to-throughput worse than FTP ratio and depending heavily on the RTT.*



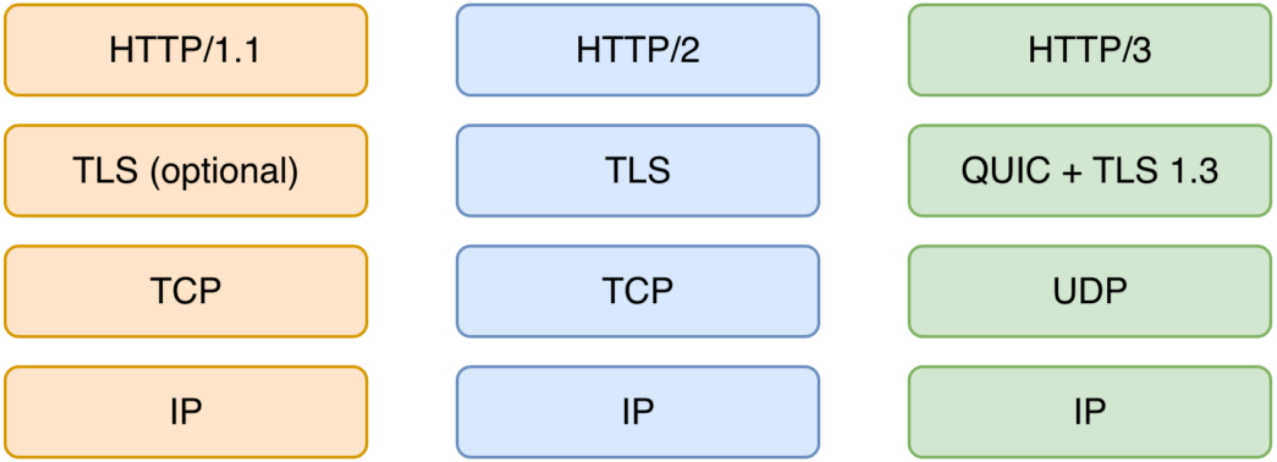
[1] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac, "Interactions between HTTP adaptive streaming and TCP," in ACM NOSSDAV, Jun. 2012.

HTTP Adaptive Streaming

1. Basics
2. Workflow
3. Performance
4. **Advanced Transport Options**

Advanced Transport Options

- **HTTP/2:**
  - HTTP/2 (2015) is a HTTP version based on Google's SPDY Protocol, which was implemented to speed up the loading of web pages.
  - It is a recent standard and is already starting to take over from the current protocol HTTP1.1 (1999) still used by most websites.
- **HTTP/3 over QUIC (Quick UDP Internet Connections):**
  - HTTP/3 (2022) is the last HTTP version based. HTTP/3 supersedes HTTP/2 by using as transport protocol QUIC (2015) instead of TCP.
  - QUICK is an entirely new transport protocol for the web developed on top of UDP instead of TCP by Google.
  - Deployed in every major Google Site on Desktop and Android Chrome.

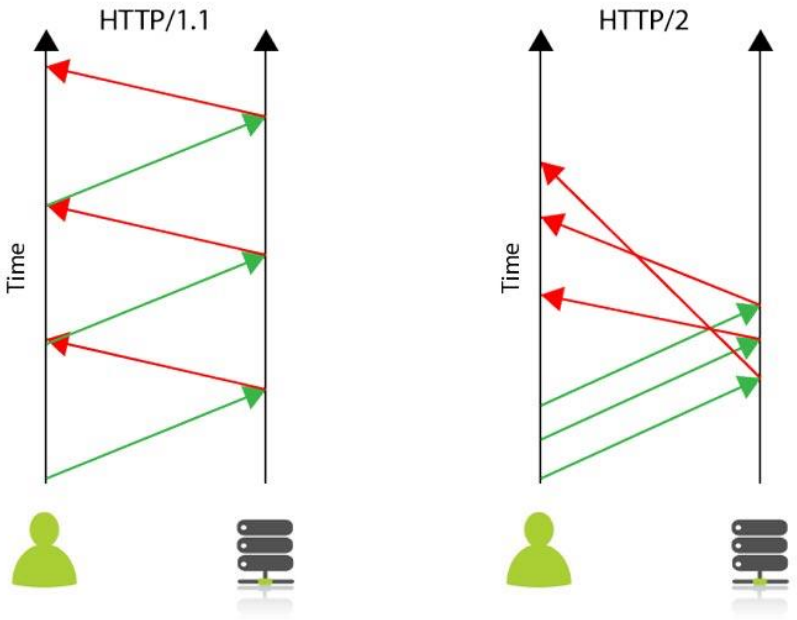


[*Google report*, QUIC: Next generation multiplexed transport over UDP.]
[https://zadroweb.com/blog/http3-introduction-web-protocols/]

Advanced Transport Options

HTTP/2: Multiplexing and concurrency

- Modern websites require the web browser to make a significant number of requests to render a web page (many objects: HTML, CSS, JavaScript, images).
- HTTP/1.0 allowed just one request to be made at a time, ***while HTTP/1.1 and HTTP/2 allows multiple requests***
- HTTP/1.1 allows
  - no pipelining (see picture)
  - ***synchronous pipelining***: multiple requests in parallel, but with "head-of-line blocking" problem:
    - at the client side, before a subsequent request can be made the results of some previous requests must have been received (objects have priorities).
    - the server must send its responses in the same order that the requests were received (connection remains FIFO (first-in, first-out)

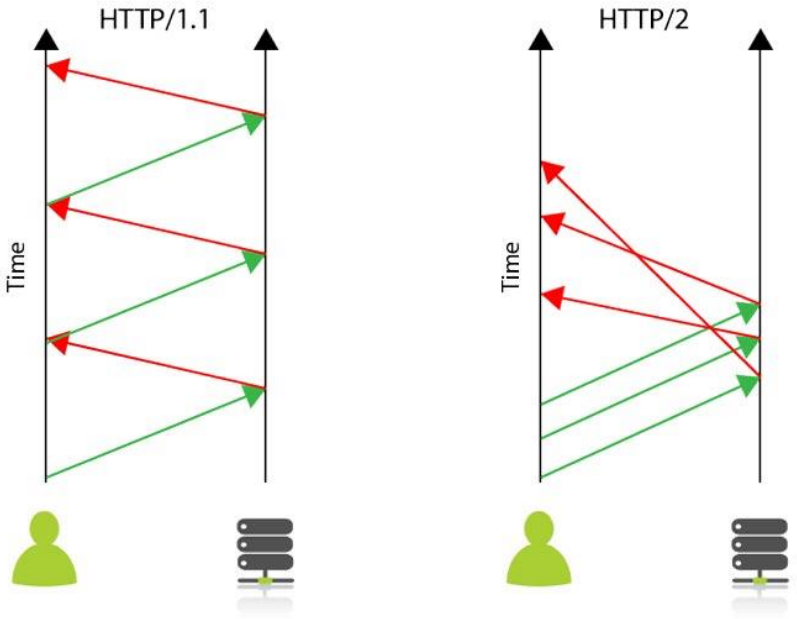


[https://www.thewebmaster.com/hosting/2015/dec/14/what-is-http2-and-how-does-it-compare-to-http1-1/

Advanced Transport Options

HTTP/2: Multiplexing and concurrency

- HTTP/2 allows ***asynchronous pipelining:***
- Head-of-line blocking is eliminated, the website can be loaded more quickly:
  - at the client side, the browser can indicate to the server the priorities for the objects. The objects with similar priorities can be grouped (multiplexed) in the same request.
  - at the server, the server can send the responses in a different order that the requests were received, but respecting the priorities.



[https://www.thewebmaster.com/hosting/2015/dec/14/what-is-http2-and-how-does-it-compare-to-http1-1/

Advanced Transport Options

HTTP/2: Header compression

- With HTTP/1.1 headers have to be provided for every object requested, which if you have a page with 100 requests can be time-consuming, and use bandwidth.
- HTTP/2 can send all the headers in a single connection, also utilizing compression. Instead of 100 round trips required to load the headers for all the objects, it can now be done in a single trip.

HTTP/2: Server push

- With HTTP/2 the server can send resources the client has not yet requested. This will be a substantial boost on high latency connections as it effectively could remove multiple RRTs to the server.
- With HTTP/2 PUSH, the server can take the initiative by having some rules that trigger push at certain moments.
- For example, when the server receives a request to index.html, it can also push styles.css and script.js without waiting for the respective request from browser.

HTTP/2 and Encryption (SSL)

- Despite a strong push to make HTTPS mandatory for the new Protocol, it did not get consensus. As a result "technically" HTTP/2 does not need HTTPS to function.
- That being said, two of the leading web browsers (Firefox and Chrome) said that they would only implement HTTP/2 over TLS (SSL).

[https://www.thewebmaster.com/hosting/2015/dec/14/what-is-http2-and-how-does-it-compare-to-http1-1/

Advanced Transport Options

HTTP/3 over QUIC: Quick UDP Internet Connections (QUIC)

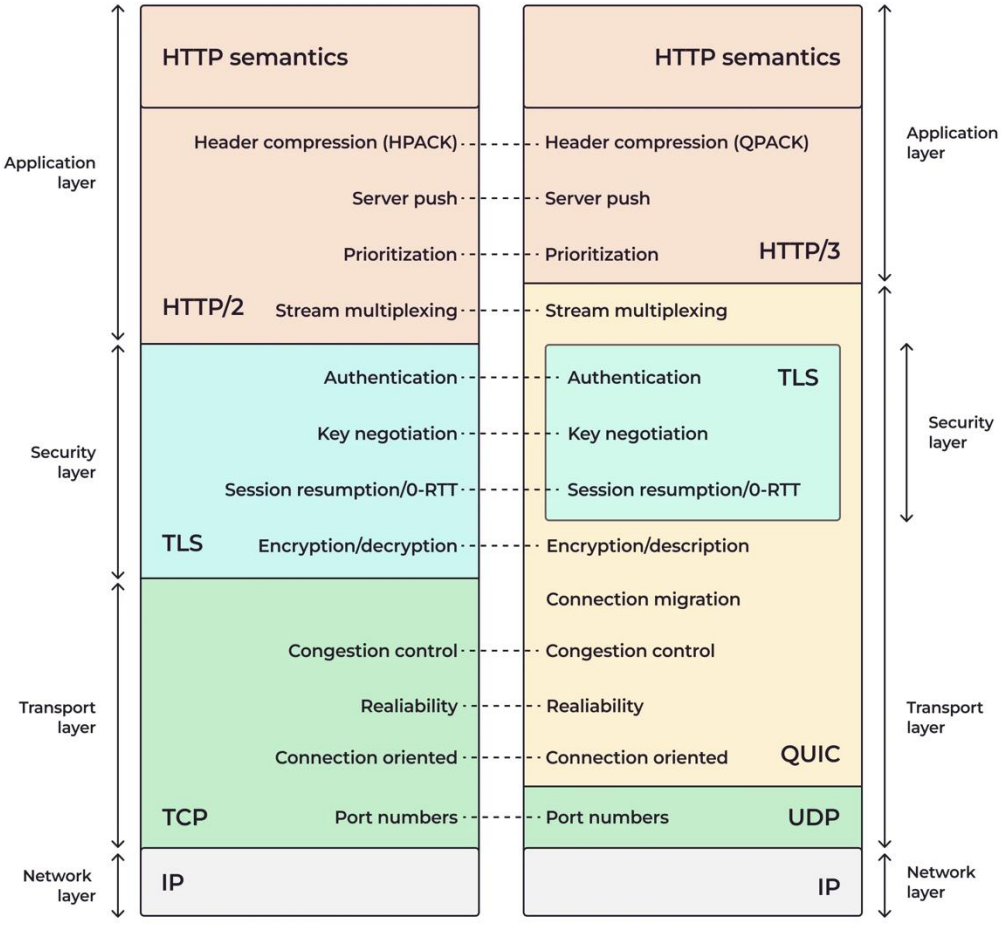
- The main added value of HTTP/3 w.r.t. HTTP/2 comes from QUCK, then we will focus on QUICK
- The common assumption is that HAS is deployed on top of the existing infrastructure, utilizing application (HTTP) and transport (TCP) protocols as is.
- But, Interestingly, standards such as DASH do not mandate the usage of any specific transport protocol.
- Then, finally, can we use UDP instead TCP under DASH?
- Yes, Google actually does it with QUIC.
- QUICs combine the *speed* of the UDP protocol with the *reliability* of the TCP protocol.
- QUIC uses 443 UDP port

[*Google report*, QUIC: Next generation multiplexed transport over UDP.]

Advanced Transport Options

Where does QUIC fit in?

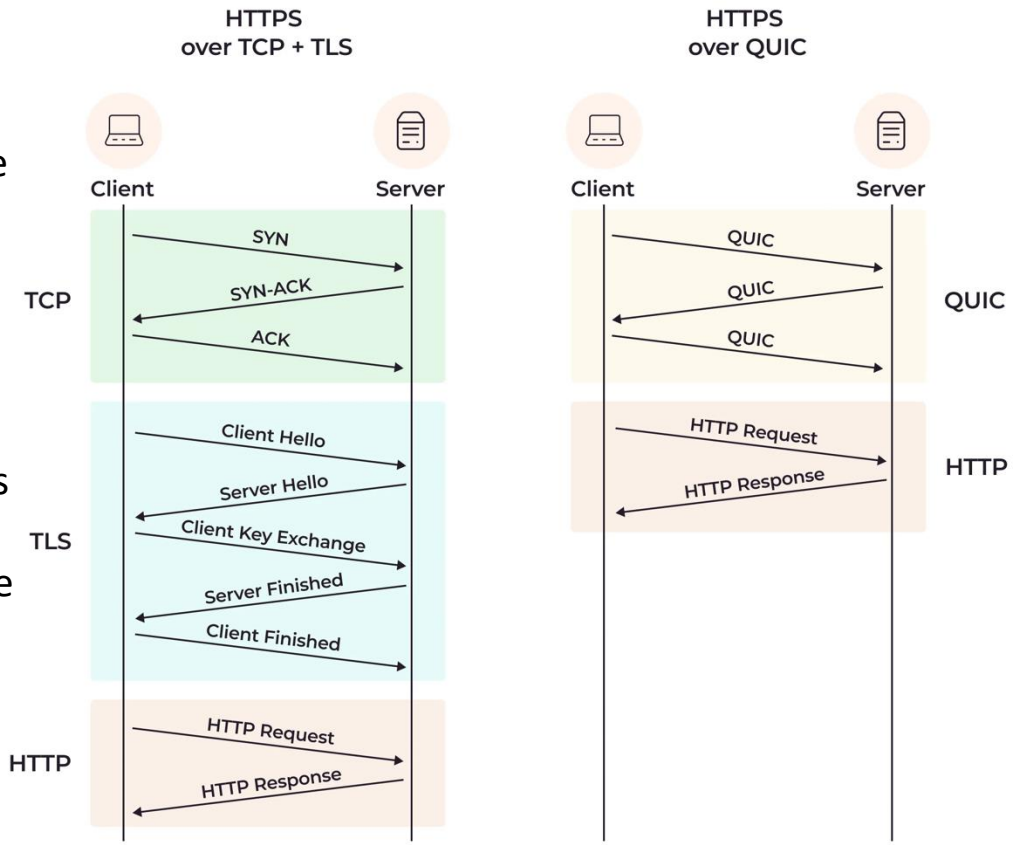
- If you look at the layers which make up a modern HTTPs connection, QUIC replaces the TLS stack and parts of HTTP/2.
- The QUIC protocol implements its own crypto-layer *so does not make use of the existing TLS 1.2.*
- It replaces TCP with UDP and on top of QUIC, a smaller *HTTP/2 API* is used to communicate with remote servers. The reason it's smaller is *because the multiplexing and connection management is already handled by QUIC.* What's left is an interpretation of the HTTP protocol.



[*Google report*, QUIC: Next generation multiplexed transport over UDP.]
[https://gcore.com/learning/what-is-http-3/]

Advanced Transport Options

- To start a TCP connection a *3-way handshake* is performed. This means additional round-trips for each starting connection.
- If on top of that you also need to negotiate TLS, to create a secure, encrypted, https connection, even more network packets have to be sent back and forth.
- UDP is a *fire and forget* protocol. A message is sent over UDP *assumed* to arrive at the destination. The benefit is less time spent on the network to validate packets, the downside is that in order to be reliable, something has to be built *on top of* UDP to confirm packet delivery.
- The QUIC protocol can start a connection *and* negotiate all the TLS (HTTPs) parameters in 1 or 2 or 3 packets (depends on if it's a new server you are connecting to or a known host).

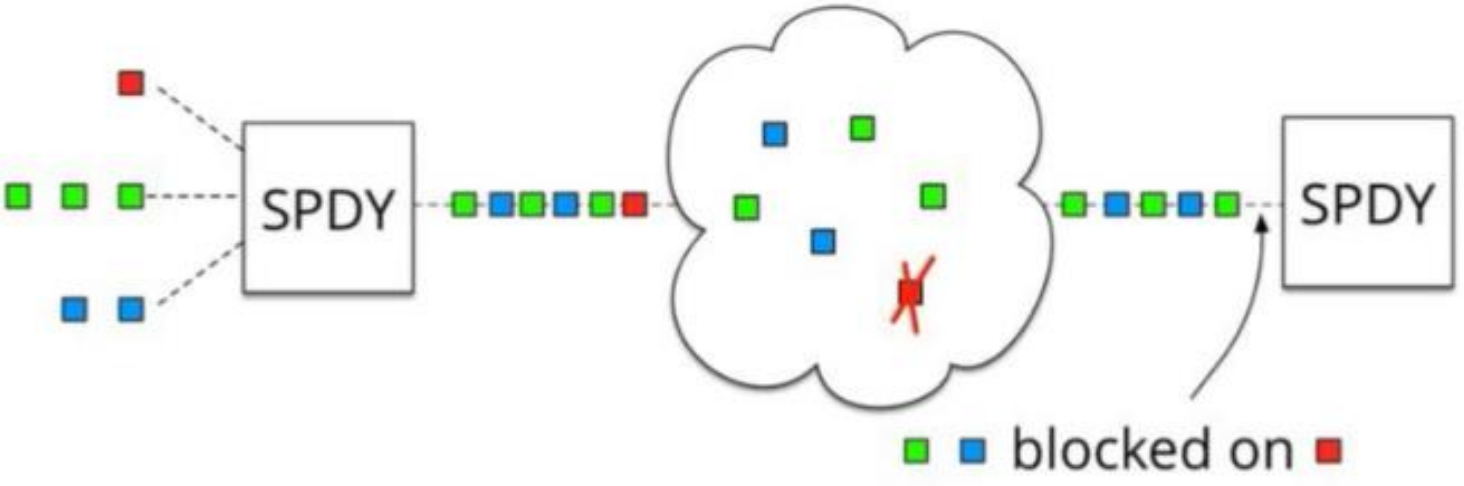


[*Google report*, QUIC: Next generation multiplexed transport over UDP.]
[https://gcore.com/learning/what-is-http-3/]

Advanced Transport Options

No TCP head-of-line blocking

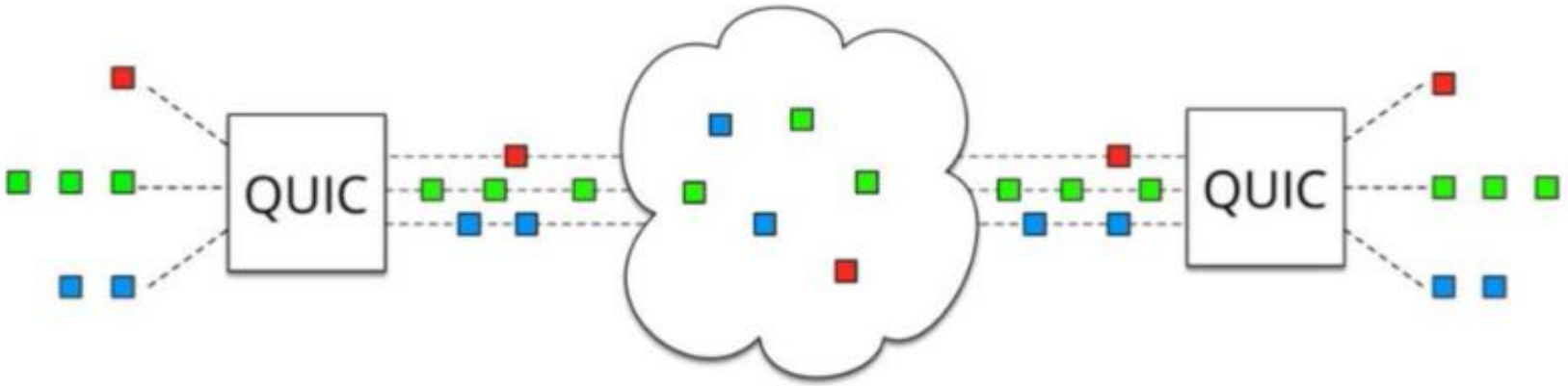
- With SPDY or HTTP/2 we have a *single* TCP connection being used to connect to a server instead of multiple connections for each asset. The TCP connection can independently request and receive resources, but there is a problem: the TCP head-of-line blocking of the single connection.
- In TCP, packets need to be processed in the correct order. If a packet is lost, it needs to be retransmitted. The TCP connection needs to wait (or "block") on that TCP packet before it can continue to parse the other packets.



Advanced Transport Options

No TCP head-of-line blocking

- In QUIC, this is solved by not making use of TCP anymore.
- UDP is not dependent on the order in which packets are received. While it's still possible for packets to get lost during transit, they will only impact *an individual resource* (as in: a single CSS/JS file) and not block the entire connection.
  - **Forward Error Correction:** Missing packets can be reconstructed. **T**he gain is not having to retransmit a lost packet, which would take a lot longer.
- *QUIC is essentially combining the best parts of SPDY and HTTP2 (the multiplexing) on top of a non-blocking transportation protocol (UDP).*



[*Google report*, QUIC: Next generation multiplexed transport over UDP.]

Advanced Transport Options

Congestion control and reliability

- Incorporates TCP best practices of flow and congestion control: TCP Cubic-fair with TCP FACK, TLP, F-RTO, Early Retransmit.

Connection migration

- With QUIC, since it's not using TCP, we don't need quadruplets (source IP/port → destination IP/port) to start / resume connections
- QUIC has implemented its own identifier for unique connections called the **Connection UUID**.
  - It's possible to go from WiFi to LTE and still keep your Connection UUID, so no need to renegotiate the connection or TLS. Your previous connection is still valid.

Client-side protection

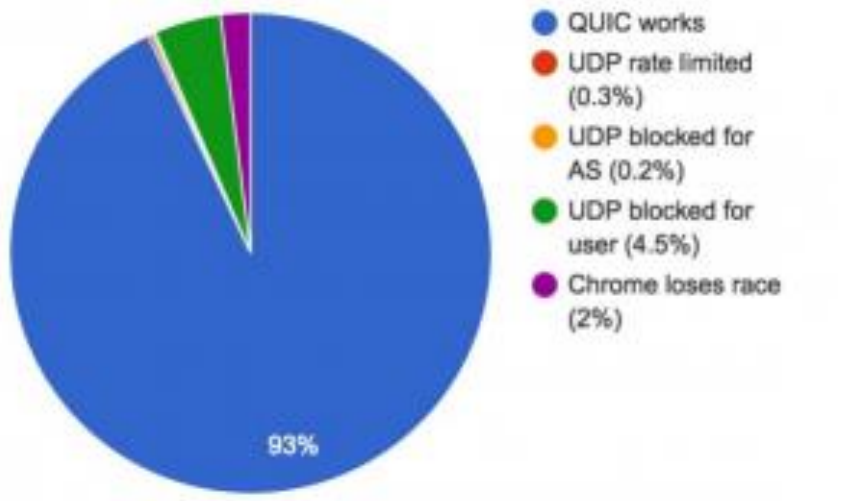
- What if UDP is blocked?
  - Chrome seamlessly falls back to HTTP/TCP
- What if the path MTU is too small?
  - QUIC handshake fails, Chrome falls back to TCP
- What if a client doesn't want to use QUIC?
  - Chrome flag / administrative policy to disable QUIC

[*Google report*, QUIC: Next generation multiplexed transport over UDP.]

Advanced Transport Options

QUIC: Does it work?

- Since QUIC is only supported on Google Services now, the server-side firewalling is probably OK.
- These numbers are client-side only: they show how many clients are allowed to do UDP over port 443.
- QUIC can be disabled in Chrome for compliance reasons. I bet there are a lot of enterprises that have disabled QUIC so those connections aren't even attempted.



(Source: QUIC Deployment Experience @Google)