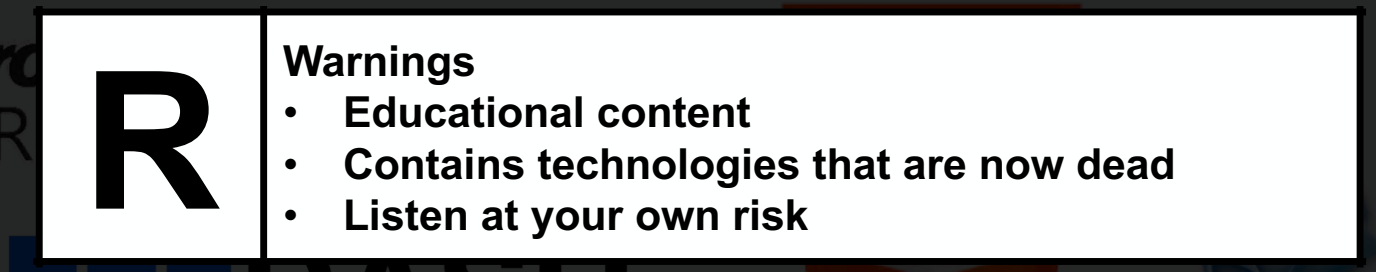
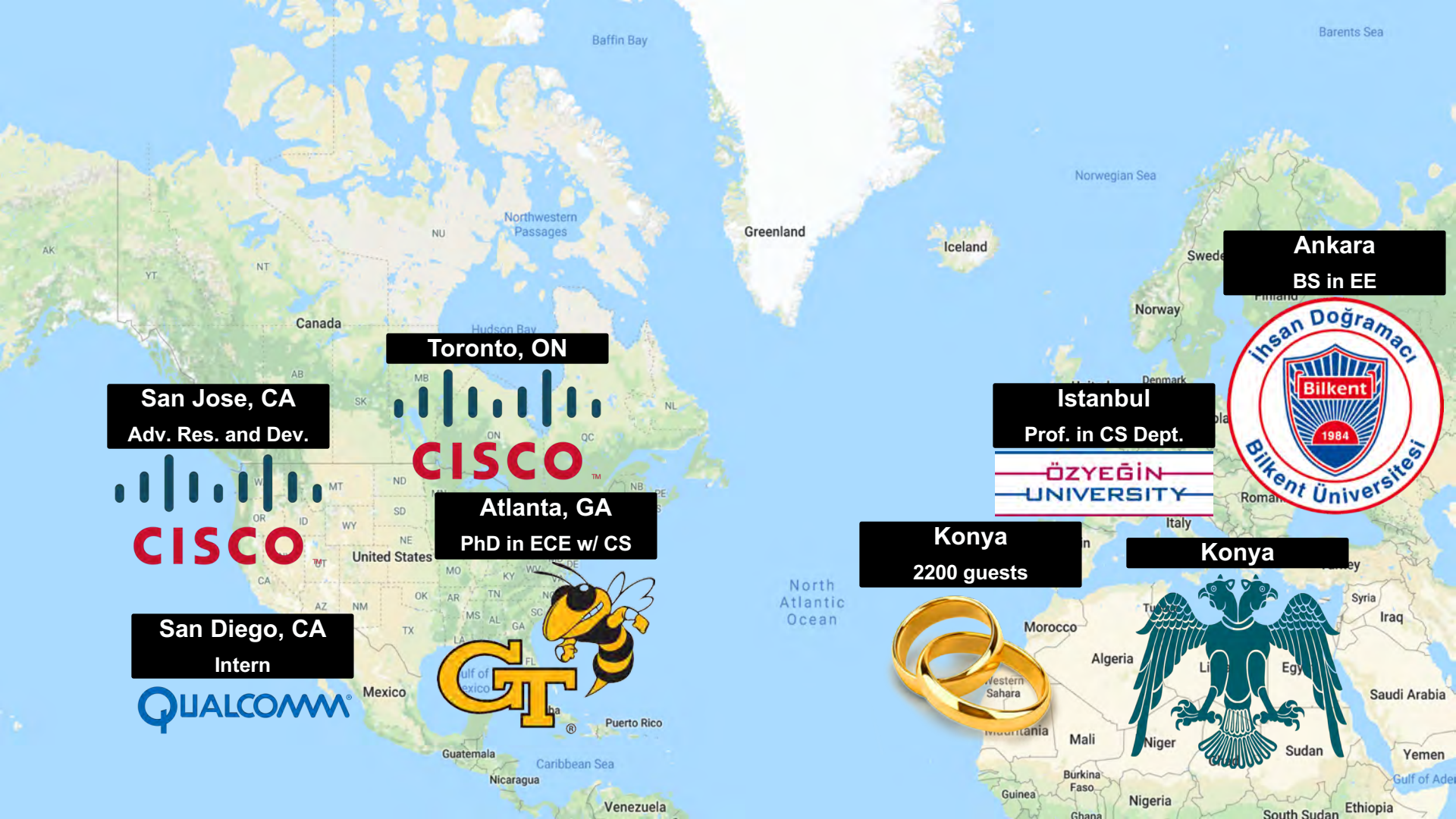


Algorithms and Formats for Adaptive Streaming

Ali C. Begen, Streaming Artichoke

Viewer Discretion is Advised

The following content may contain elements that are not suitable for some audiences

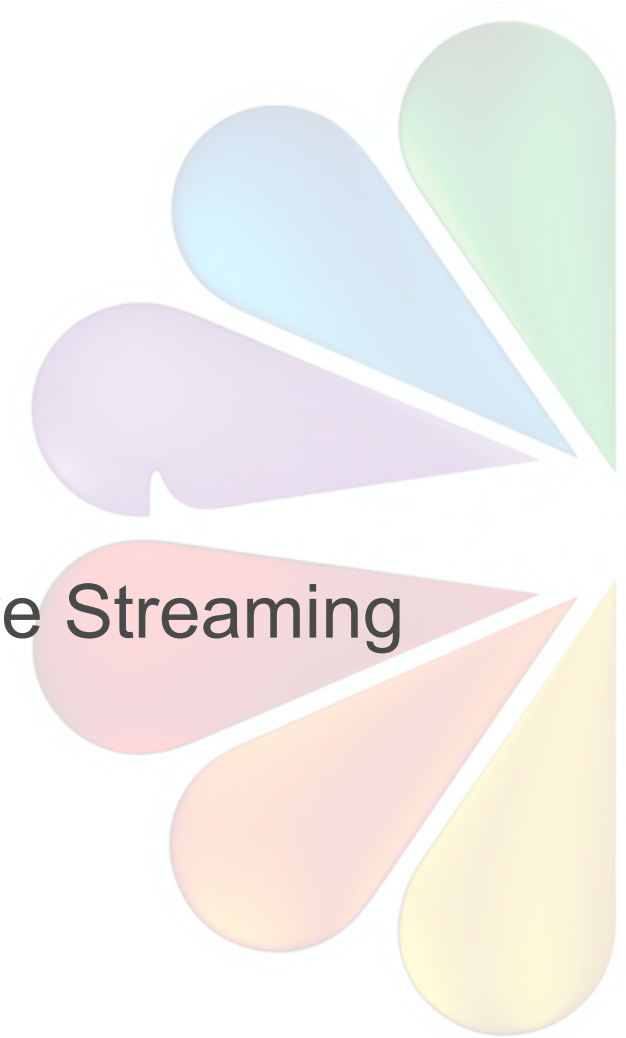

Baffin Bay
Barents Sea
Norwegian Sea
Greenland
Iceland
Northwestern Passages
Canada
Hudson Bay
United States
Mexico
Puerto Rico
Guatemala
Caribbean Sea
Nicaragua
Venezuela
North Atlantic Ocean
Norway
Italy
Algeria
Morocco
Western Sahara
Mauritania
Mali
Niger
Sudan
Nigeria
Burkina Faso
Guinea
Ghana
Ethiopia
South Sudan
Saudi Arabia
Yemen
Iraq
Syria
Gulf of Aden
Toronto, ON
San Jose, CA
Adv. Res. and Dev.
Atlanta, GA
PhD in ECE w/ CS
San Diego, CA
Intern
Ankara
BS in EE
Istanbul
Prof. in CS Dept.
Konya
2200 guests
Konya

Topics to Cover

- Download vs. streaming
- Common issues in scaling and multi-screen/hybrid delivery
- Status of MPEG DASH
- Improving QoE in streaming

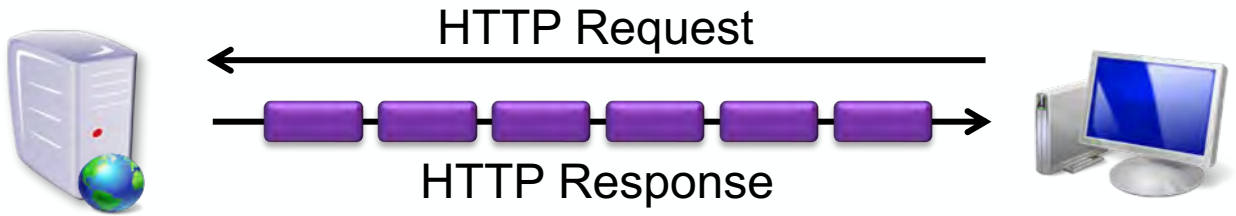
Algorithms and Formats for Adaptive Streaming

Download vs. Streaming

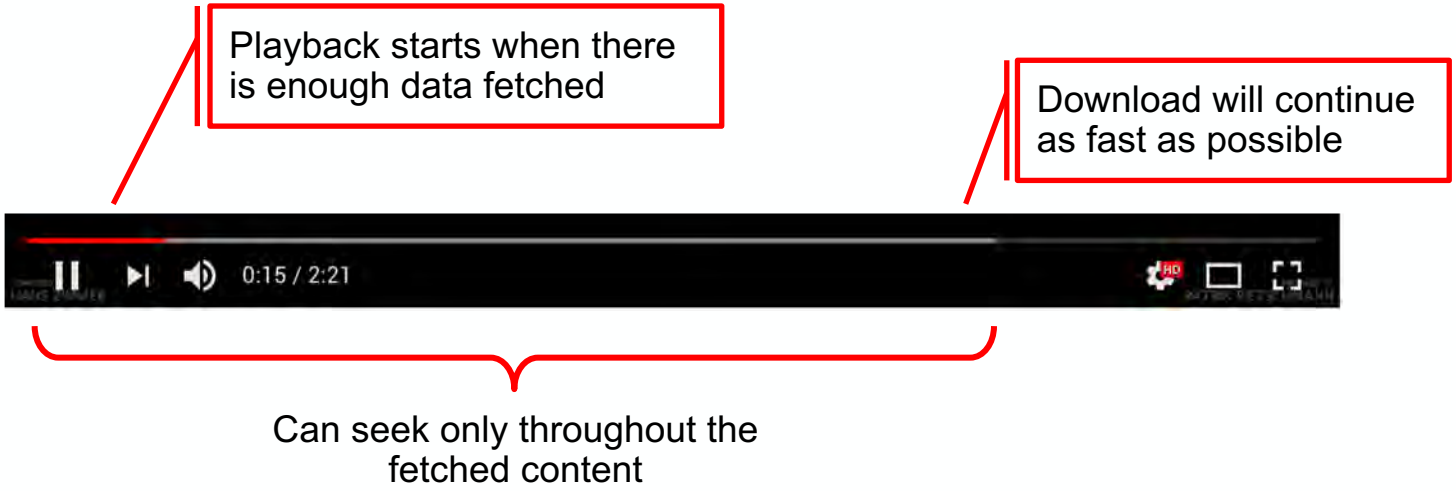


Progressive Download

One Request, One Response



Progressive Download Scenario



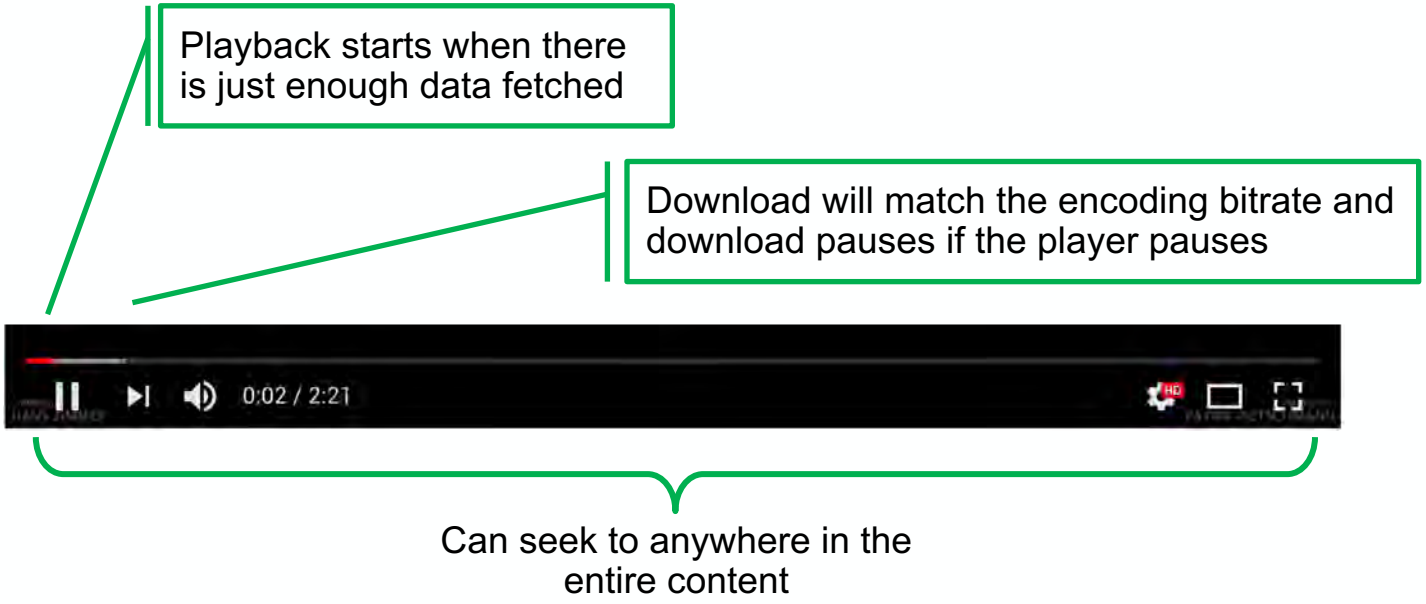
What is Streaming?

Streaming is transmission of a continuous content from a server to a client and its simultaneous consumption by the client

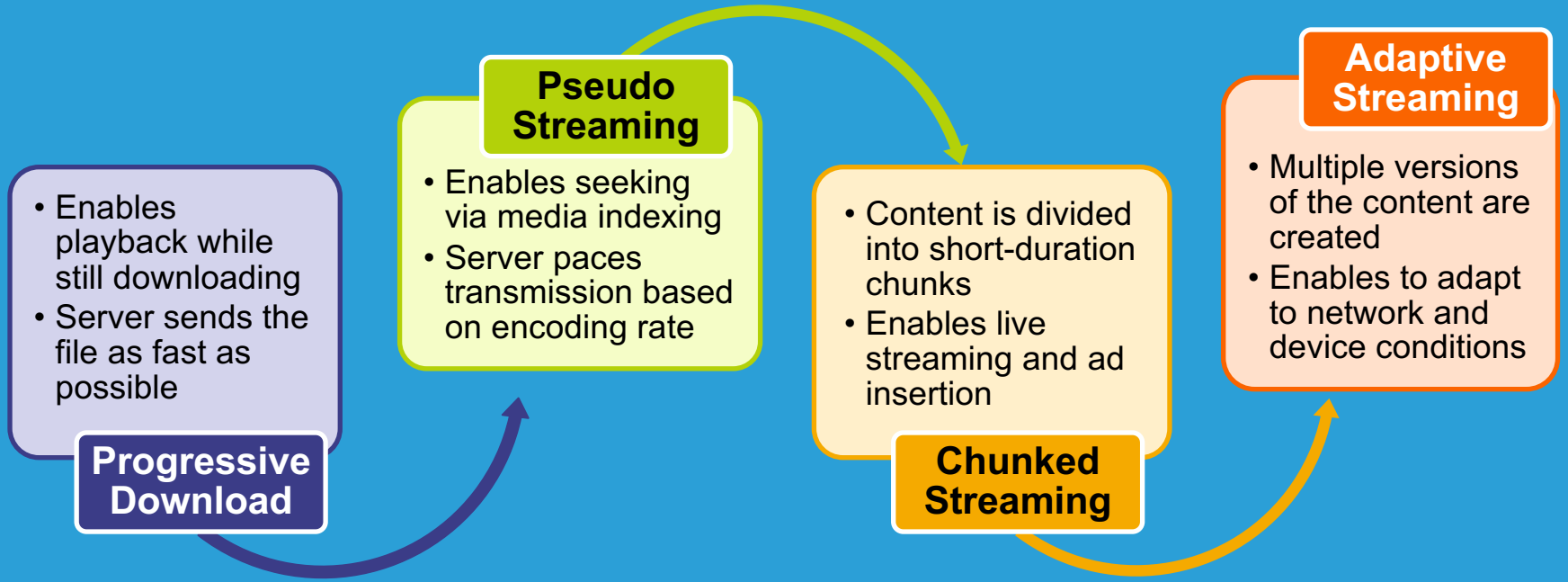
Two Main Characteristics

1. Client consumption rate may be limited by real-time constraints as opposed to just bandwidth availability
2. Server transmission rate (loosely or tightly) matches to client consumption rate

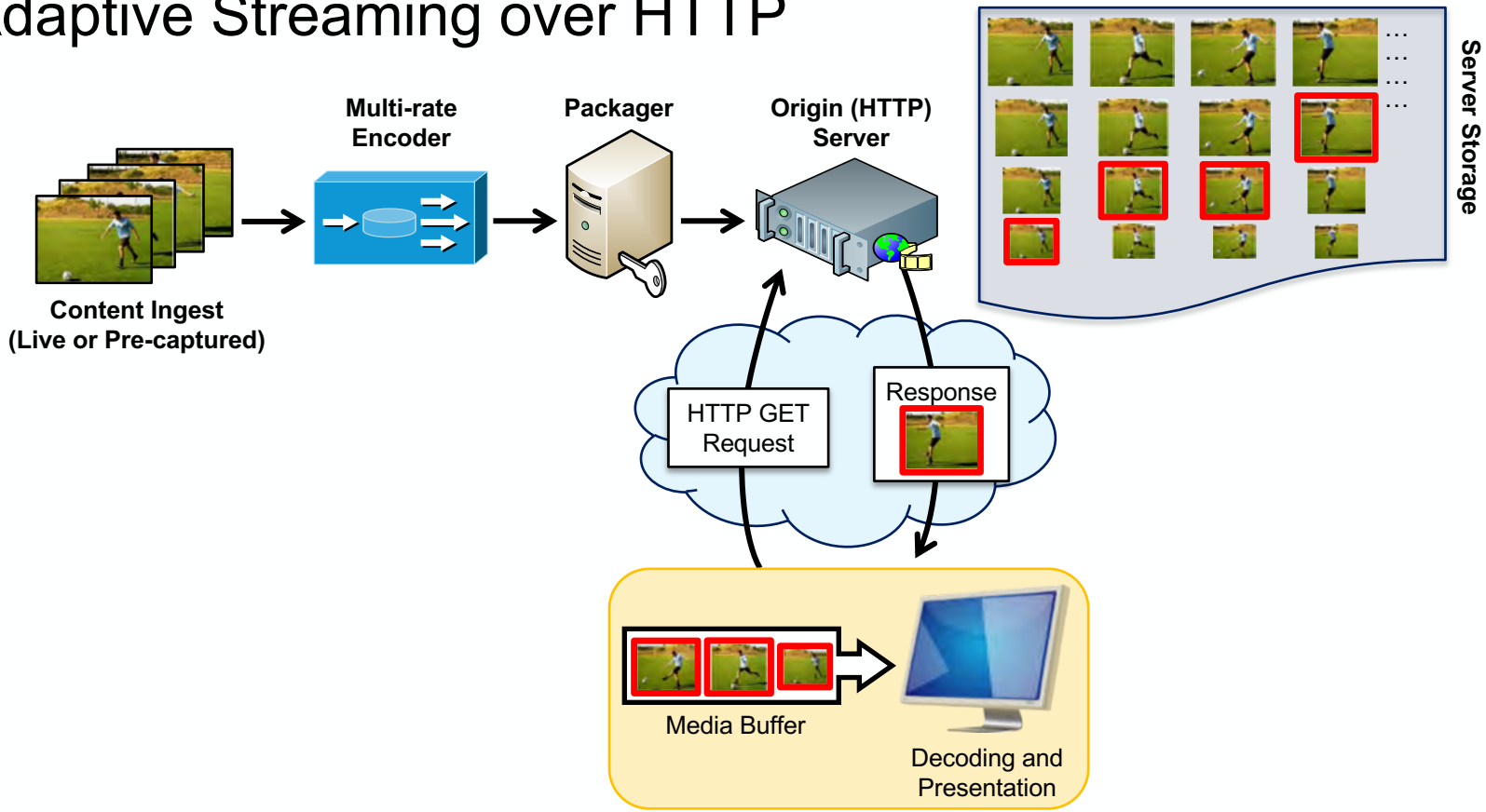
Streaming Scenario



Video Delivery over HTTP



Adaptive Streaming over HTTP



HTTP Adaptive Streaming

Adapt Video to Web Rather than Changing the Web

- Imitation of streaming via short downloads
  - Downloads small chunks to minimize bandwidth waste
  - Enables to monitor consumption and track the streaming clients
- Adaptation to dynamic conditions and device capabilities
  - Adapts to dynamic conditions in the Internet and home network
  - Adapts to display resolution, CPU and memory resources of the streaming client

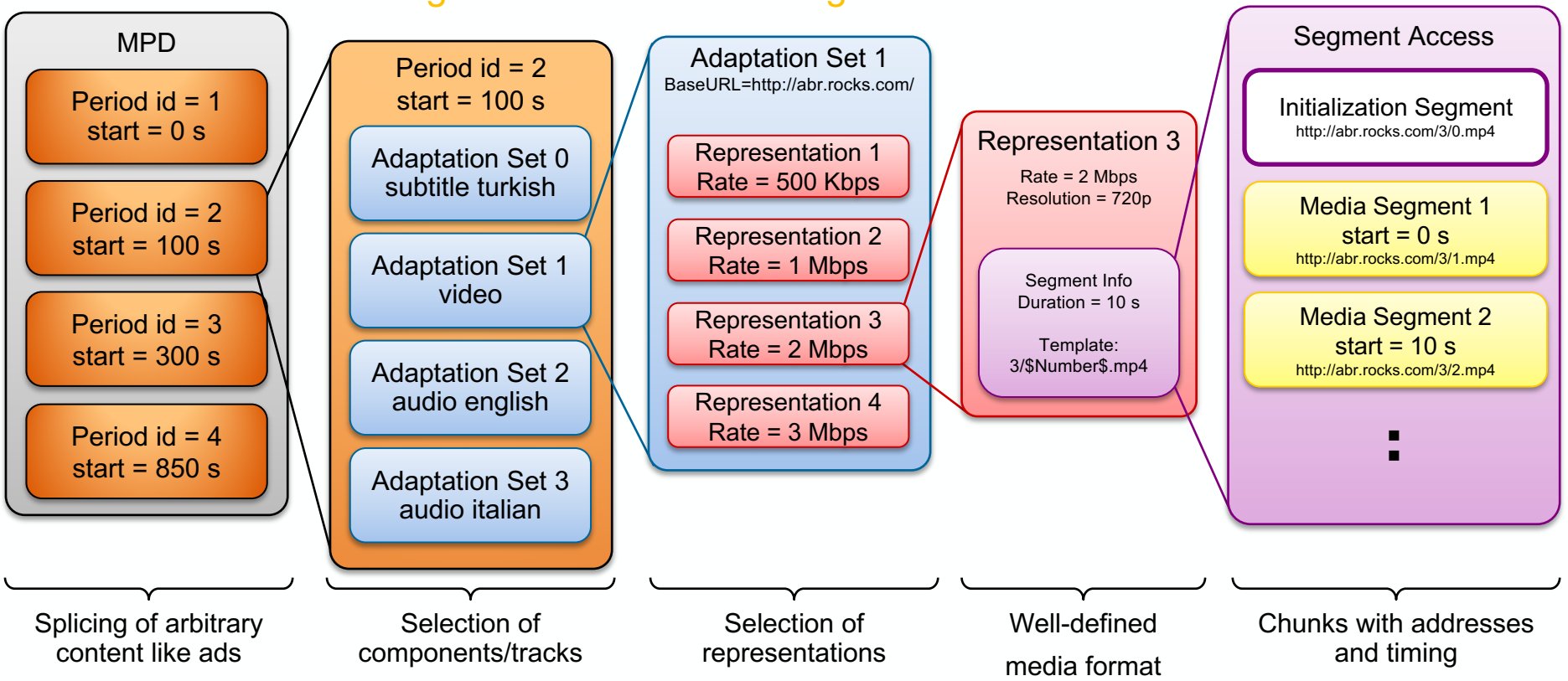
  → Facilitates “any device, anywhere, anytime” paradigm
- Improved quality of experience (not necessarily mean improved average quality)
  - Enables faster start-up and seeking, and quicker buffer fills
  - Reduces skips, freezes and stutters
- Use of HTTP
  - Well-understood naming/addressing approach, and authentication/authorization infrastructure
  - Provides easy traversal for all kinds of middleboxes (e.g., NATs, firewalls)
  - Enables cloud access, leverages the existing (cheap) HTTP caching infrastructure

Dead, Surviving, Maturing and Newborn Technologies

- Move Adaptive Stream (Long gone, but some components are in Slingbox)
  - http://www.movenetworks.com
- Microsoft Smooth Streaming (Legacy)
  - http://www.iis.net/expand/SmoothStreaming
- Adobe Flash (Almost dead)
  - http://www.adobe.com/products/flashplayer.html
- Adobe HTTP Dynamic Streaming (Legacy)
  - http://www.adobe.com/products/httpdynamicstreaming
- Apple HTTP Live Streaming (The elephant in the room)
  - https://tools.ietf.org/html/rfc8216
  - https://datatracker.ietf.org/doc/draft-pantos-hls-rfc8216bis
- MPEG DASH and CMAF (The standards)
  - http://mpeg.chiariglione.org/standards/mpeg-dash
  - http://mpeg.chiariglione.org/standards/mpeg-a/common-media-application-format

An Example DASH Template-Based Manifest

List of Accessible Segments and Their Timings



An Example HLS Playlist-Based Manifest

master.m3u8

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=232370,CODECS=
gear1/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=649879,CODECS=
gear2/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=41457,CODECS="
gear0/prog_index.m3u8
```

gear1/prog_index.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
.
.
.
#EXT-X-ENDLIST
```

Source: https://developer.apple.com/streaming/examples/ and https://www.gpac-licensing.com/2014/12/01/apple-hls-comparing-versions/

Example Representations

Vancouver 2010

	Encoding Bitrate	Resolution
Rep. #1	3.45 Mbps	1280 x 720
Rep. #2	1.95 Mbps	848 x 480
Rep. #3	1.25 Mbps	640 x 360
Rep. #4	900 Kbps	512 x 288
Rep. #5	600 Kbps	400 x 224
Rep. #6	400 Kbps	312 x 176

Sochi 2014

	Encoding Bitrate	Resolution
Rep. #1	3.45 Mbps	1280 x 720
Rep. #2	2.2 Mbps	960 x 540
Rep. #3	1.4 Mbps	960 x 540
Rep. #4	900 Kbps	512 x 288
Rep. #5	600 Kbps	512 x 288
Rep. #6	400 Kbps	340 x 192
Rep. #7	200 Kbps	340 x 192

PyeongChang 2018

	Encoding Bitrate	Resolution
Rep. #1	18 Mbps	4K (60p)
Rep. #2	12.2 Mbps	2560x1440 (60p)
Rep. #3	4.7 Mbps	2K (60p)
Rep. #4	3.5 Mbps	1280x720 (60p)
Rep. #5	2 Mbps	1280 x 720
Rep. #6	1.2 Mbps	768 x 432
Rep. #7	750 Kbps	640 x 360
Rep. #8	500 Kbps	512 x 288
Rep. #9	300 Kbps	320 x 180
Rep. #10	200 Kbps	320 x 180

Source: Vertigo MIX10, Alex Zambelli's Streaming Media Blog, Akamai, Comcast

HAS Working Principle

Smart and Selfish Clients

- Client fetches and parses the manifest
- Client uses the OS-provided HTTP stack (HTTP may run over TCP or QUIC)
- Client uses the required decryption tools for the protected content

HTTP Server

Request

Response

(One can also multicast media segments)

Client

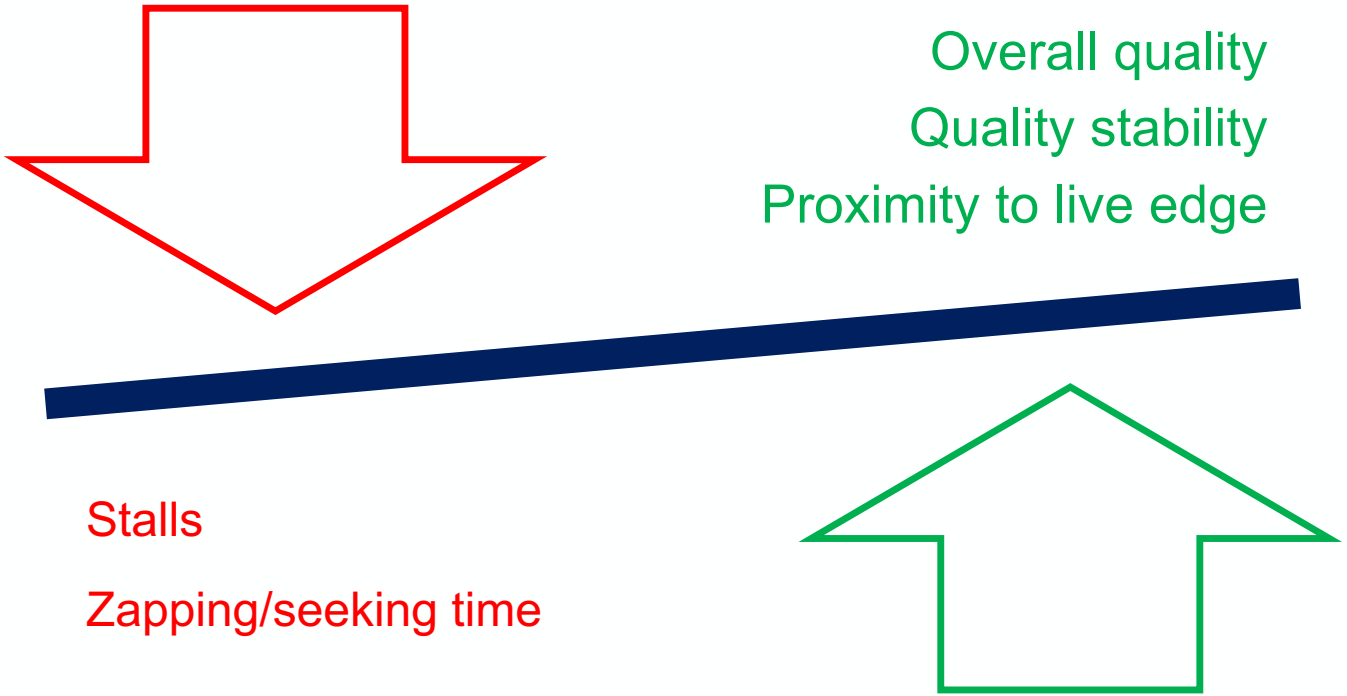
Client monitors and measures

- Size of the playout buffer (both in bytes and seconds)
- Chunk download times and throughput
- Local resources (CPU, memory, window size, etc.)
- Dropped frames

Client performs adaptation

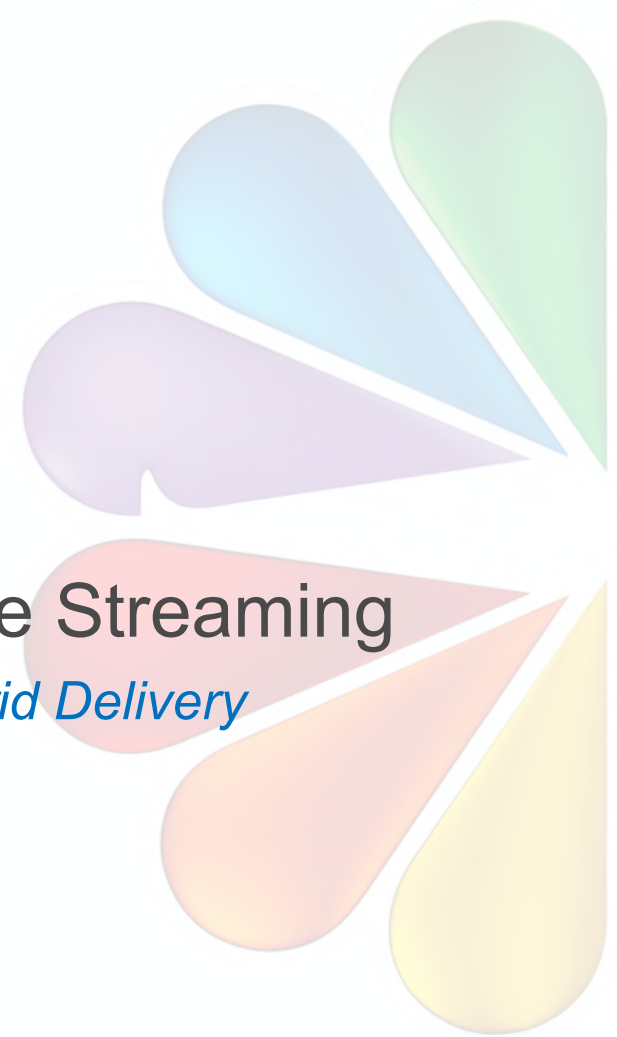
Client measures and reports metrics for analytics

Tradeoffs in Adaptive Streaming



Algorithms and Formats for Adaptive Streaming

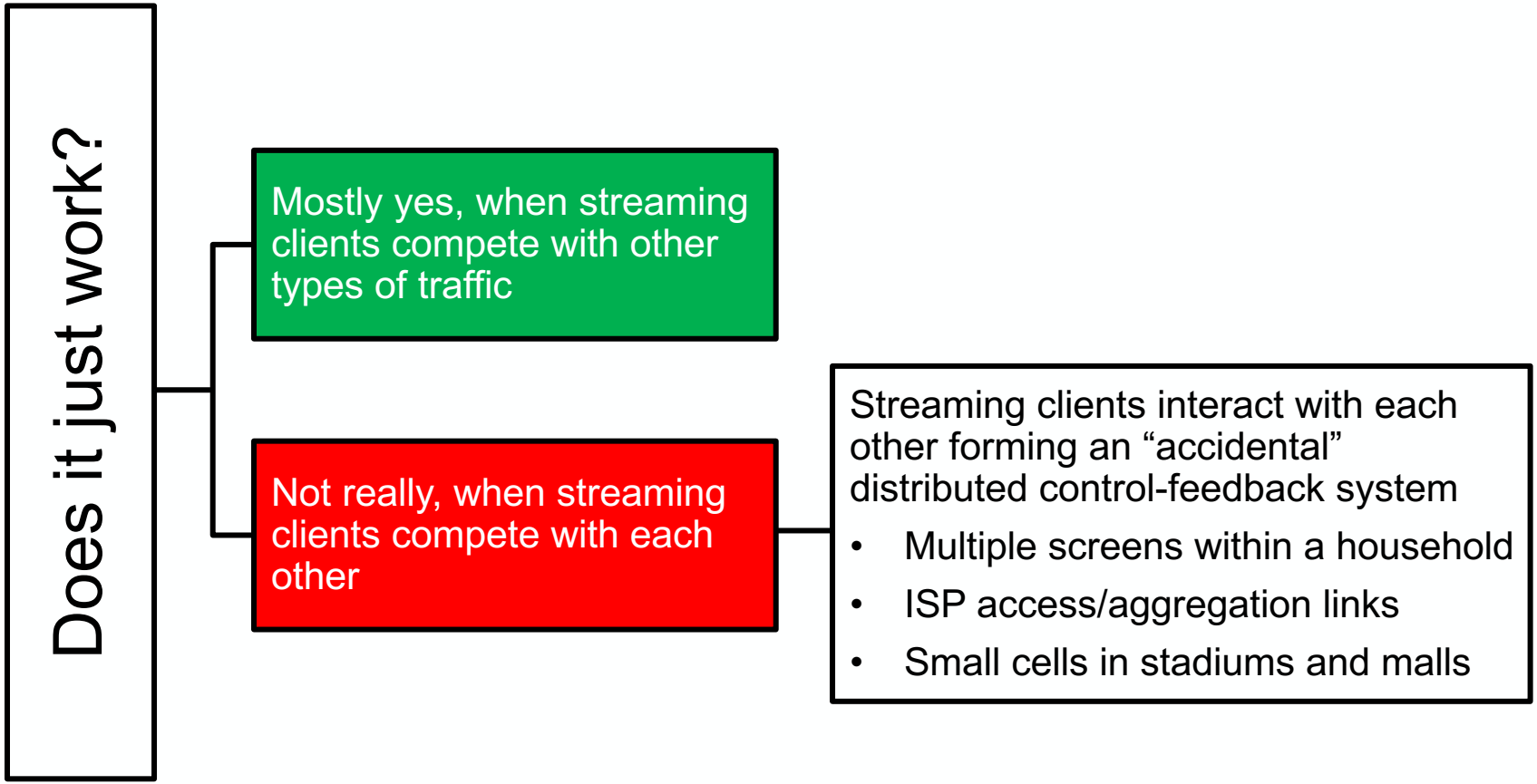
Common Issues in Scaling and Multi-Screen/Hybrid Delivery



Streaming over HTTP – The Promise

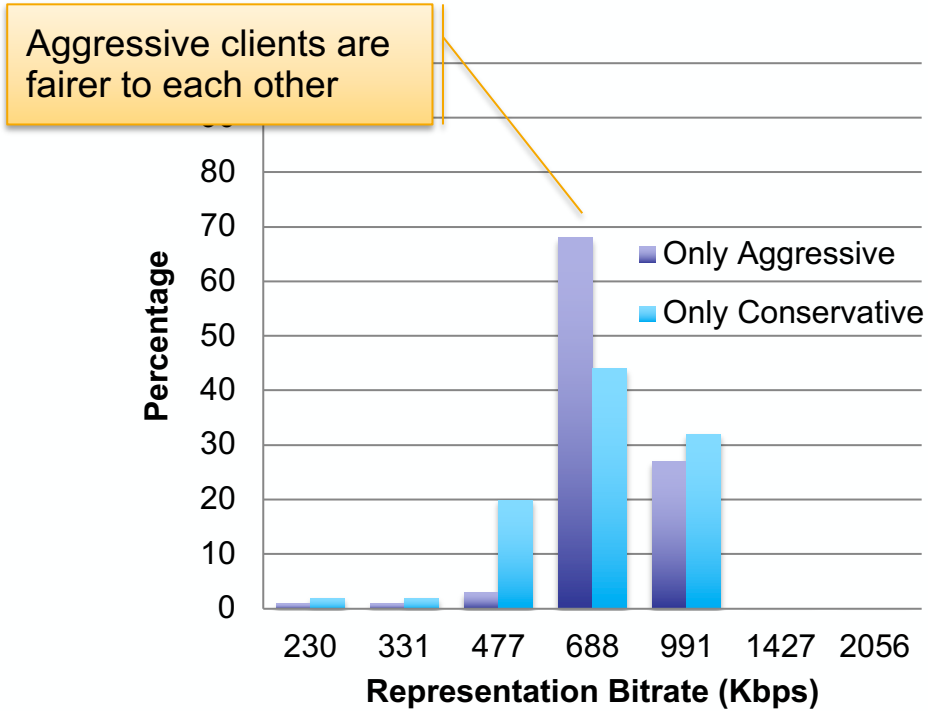
- Leverage tried-and-true Web infrastructure for scaling
  - Video is just ordinary Web content!
- Leverage tried-and-true TCP
  - Congestion avoidance
  - Reliability
  - No special QoS for video

THERE
IT SHOULD
JUST WORK

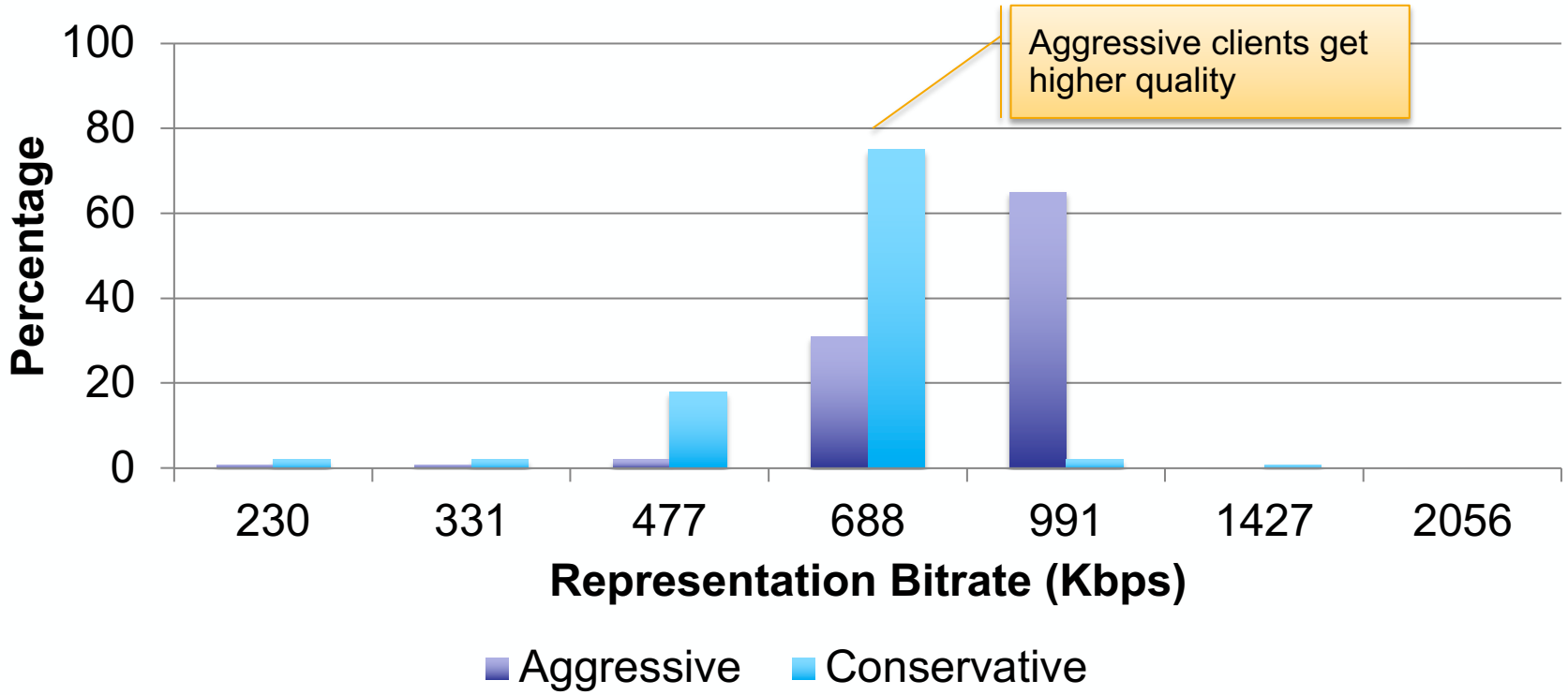

Does it just work?
Mostly yes, when streaming clients compete with other types of traffic
Not really, when streaming clients compete with each other
Streaming clients interact with each other forming an “accidental” distributed control-feedback system
• Multiple screens within a household
• ISP access/aggregation links
• Small cells in stadiums and malls

100 Simulated Clients Sharing a 100 Mbps Link

- Aggressive Clients
  - Stay at bitrate R provided that recent download speeds are at least 0.9xR
- Conservative Clients
  - Stay at bitrate R provided that recent download speeds are at least 1.2xR
- Parameters
  - Chunks: 2 s
  - Minimum buffer threshold: 4 s
  - Maximum buffer: 45 s

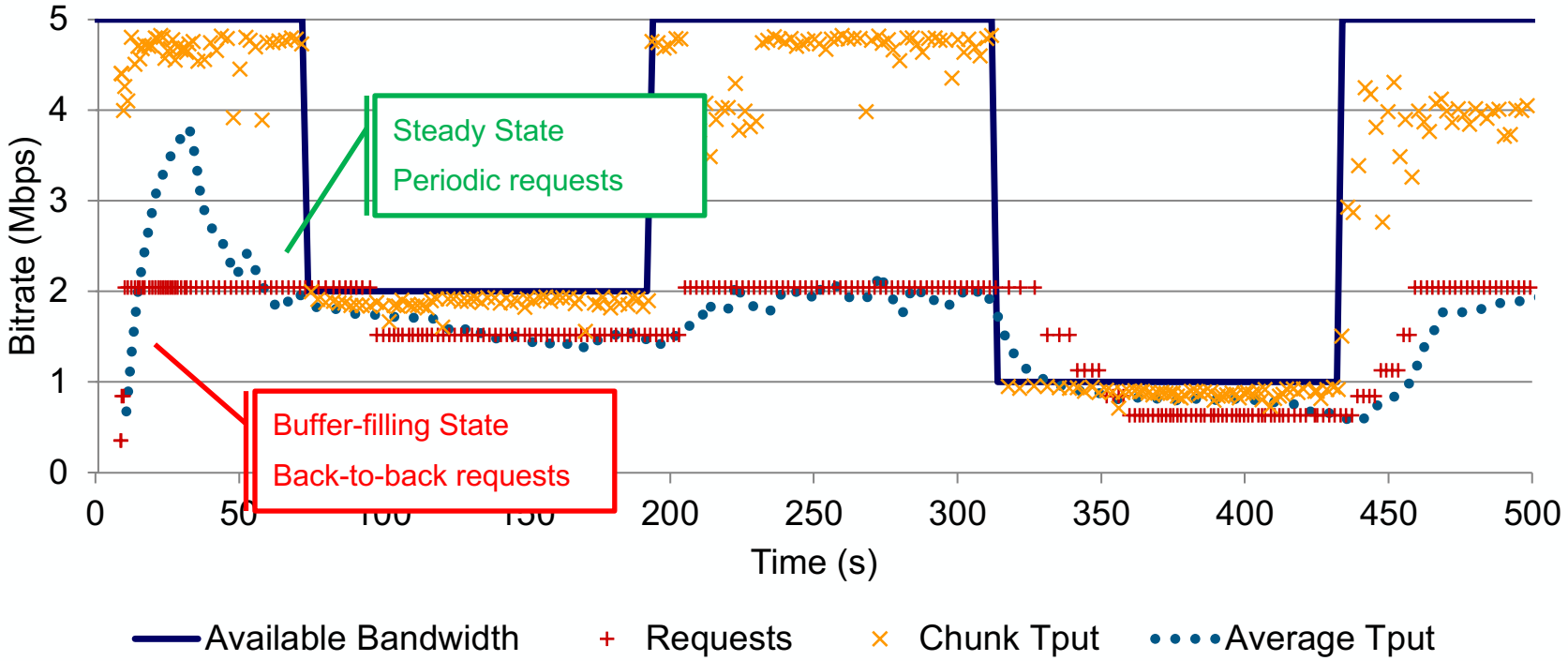


50+50 Mixed Simulated Clients Sharing a 100 Mbps Link



Demystifying a Streaming Client

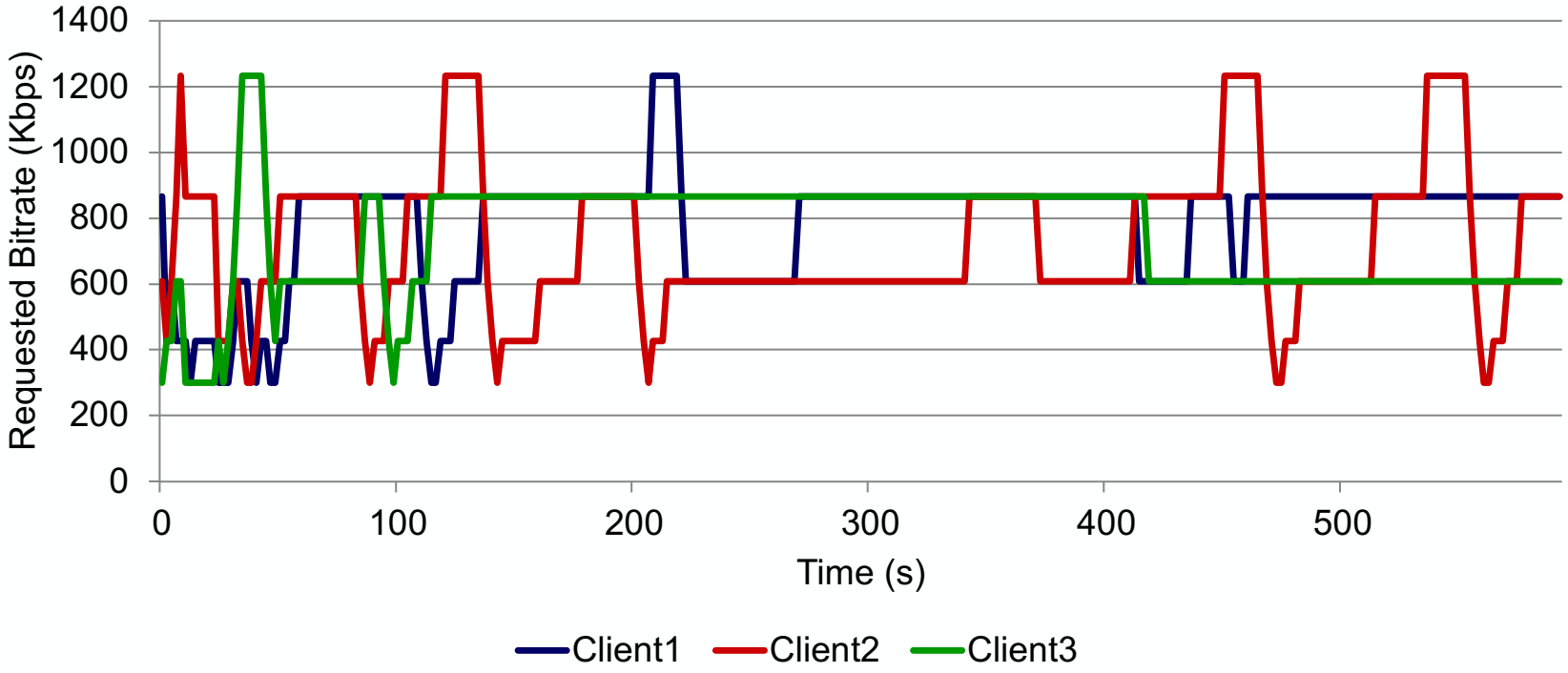
A Single Microsoft Smooth Streaming Client under a Controlled Environment



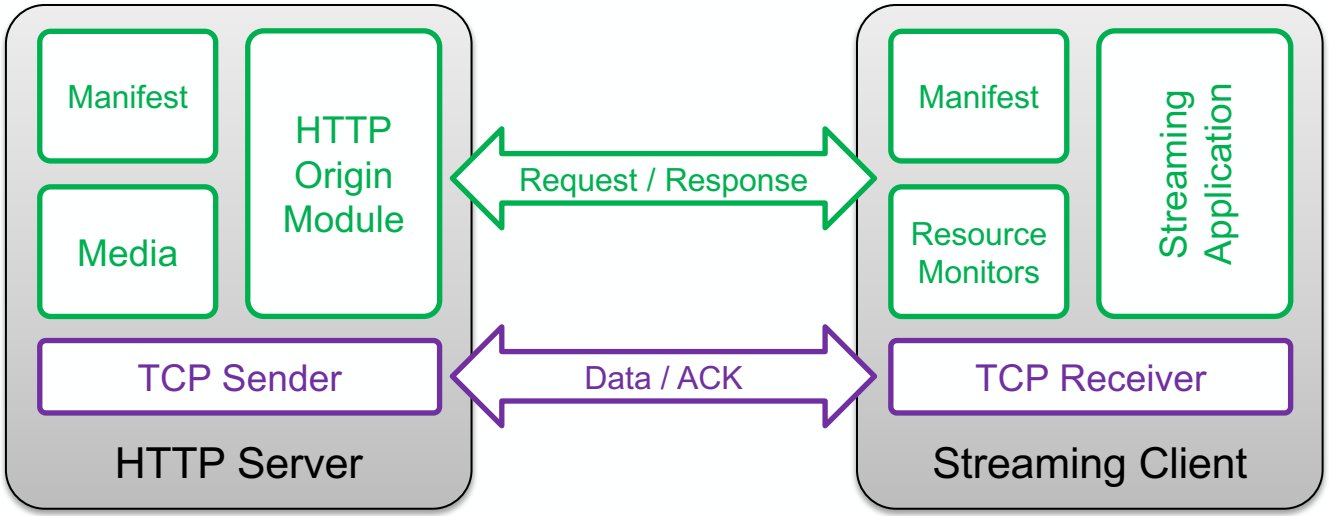
Reading: “An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP,” ACM MMSys 2011

Selfishness Hurts Everyone

10 (Commercial) Streaming Clients Sharing a 10 Mbps Link



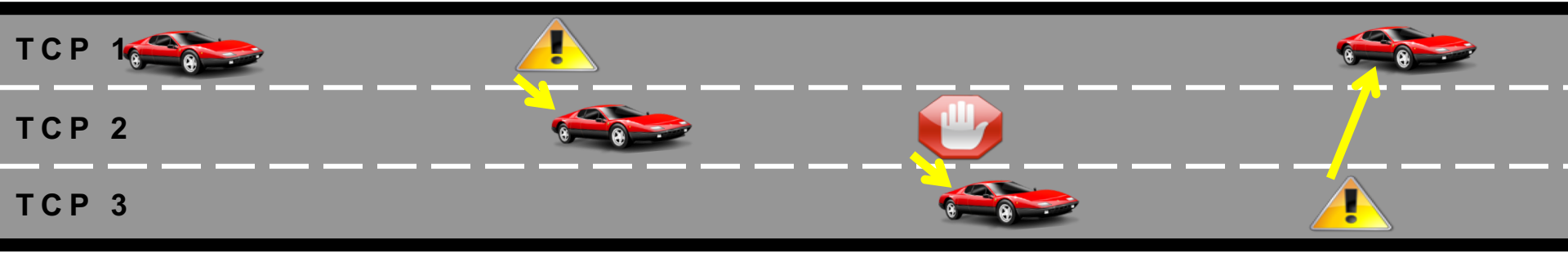
Inner and Outer Control Loops



There could be multiple TCPs destined to the same or different servers

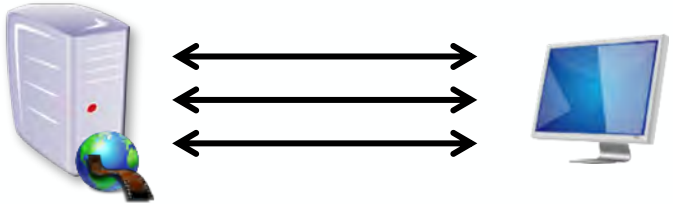
Streaming with Multiple TCP Connections

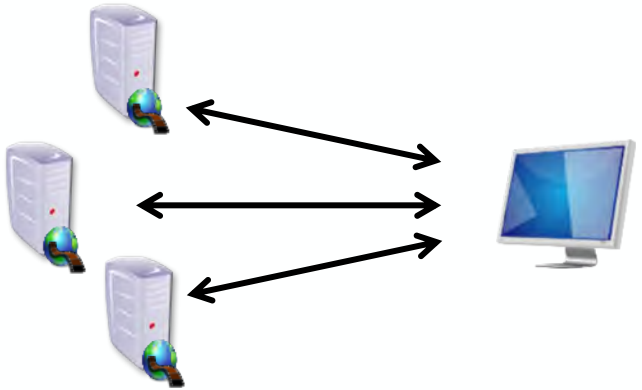
Similar to Driving on a Multi-Lane Highway



TCP Fairness ≠ Fair Streaming

Streaming with Multiple TCP Connections





- Using multiple concurrent TCPs
  - Can help mitigate head-of-line blocking
  - Allows fetching multiple (sub)segments in parallel
  - Allows to quickly abandon a non-working connection without having to slow-start a new one

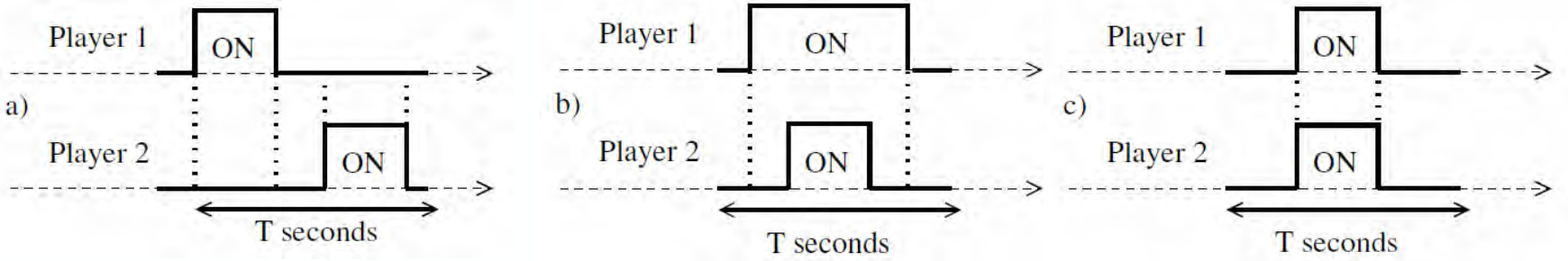
System performance deteriorates very quickly if many clients adopt this approach without limiting the aggregated bandwidth consumption

Understanding the Root Cause

Two Competing Clients

- Depending on the timing of the ON periods:
  - Unfairness, underutilization and/or instability may occur
  - Clients may grossly overestimate their fair share of the available bandwidth

Clients cannot figure out how much bandwidth to use until they use too much (Just like TCP)



Reading: “What happens when HTTP adaptive streaming players compete for bandwidth?,” ACM NOSSDAV 2012

How to Solve the Issues?

Fix the clients and/or the transport

- Use a better adaptation algorithm like PANDA or BOLA
- Use machine learning or deep learning like Pensieve
- Improve the HTTP/TCP stack, try out the alternatives
- Adopt ideas from game/consensus theory (GTA)

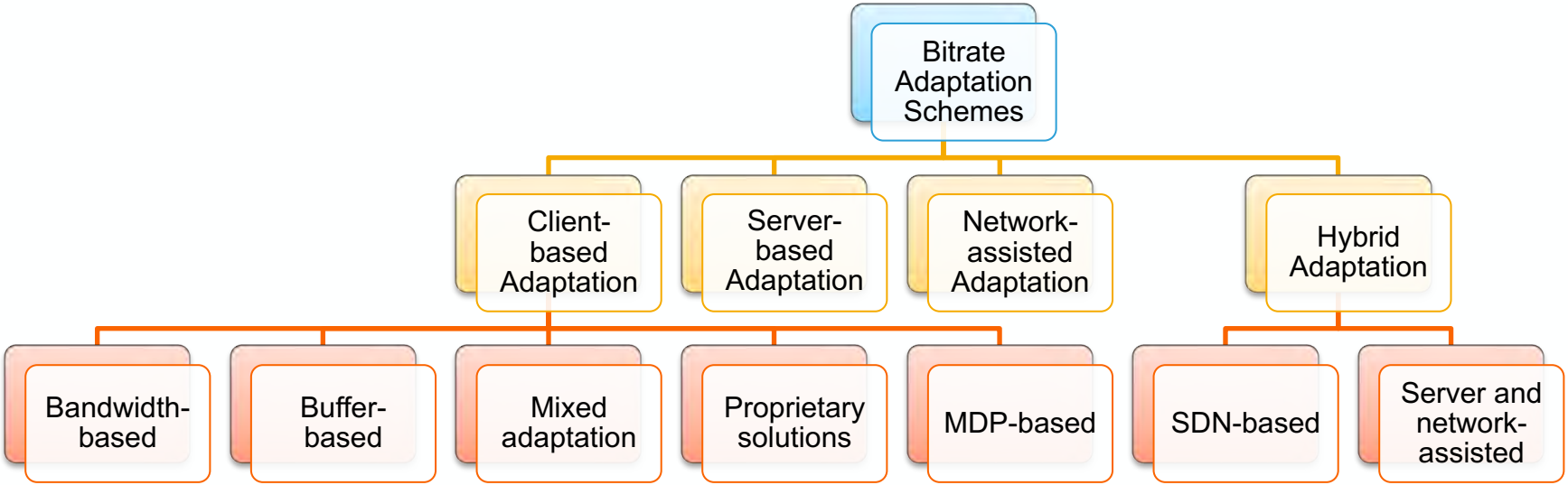
Get support from the network

- QoS in the core/edge
- SDN

Enable a control plane

- Assist the clients and network elements thru metrics and analytics

Bitrate Adaptation Schemes



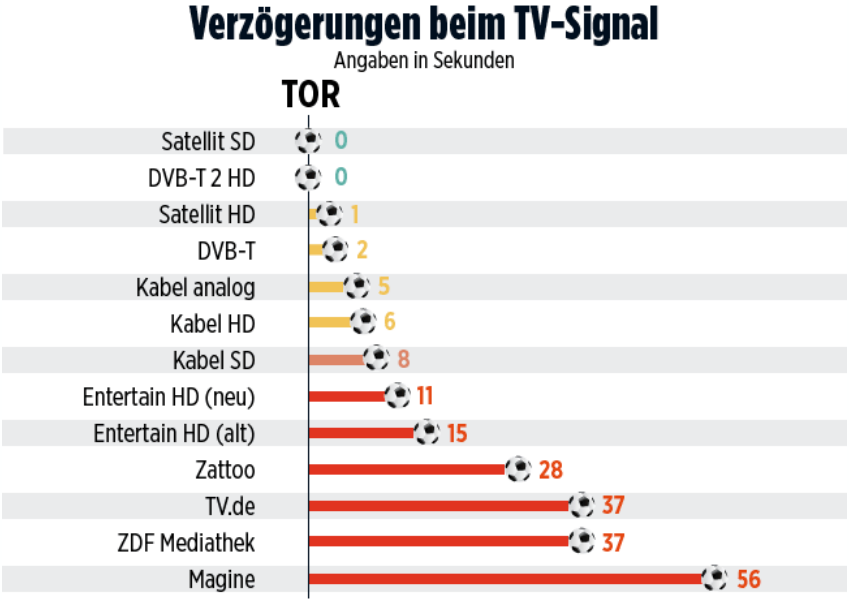
Reading: “A survey on bitrate adaptation schemes for streaming media over HTTP,” *IEEE Commun. Surveys Tuts.*, to appear

High-Level Comparison between Different Schemes

Pick Your Poison

- The client-based adaptation schemes
  - Show a good performance in single and few-client scenarios
  - Largely fail in multi-client or mixed-client scenarios
- The server-based adaptation schemes
  - Require custom servers
  - More effective in eliminating the bitrate oscillation problems
  - Less scalable due to increased complexity on the servers
- The network-assisted adaptation and hybrid schemes
  - Show a good performance in both small and large populations
  - Require modifications on the clients, server and/or network devices
  - Pose practicality issues for deployment

Why is Latency Important?



info.BILD.de | Quelle: heise online

Latency in One Slide

Contributors to the Latency

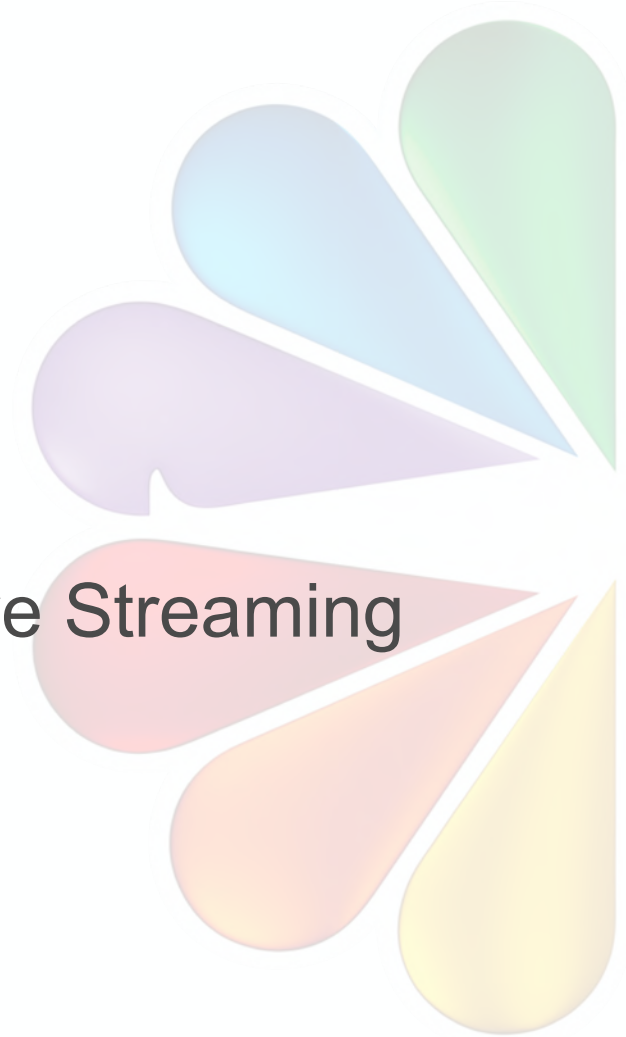
- Video encoding pipeline duration
- Ingest and packaging operations
- Network propagation delays
- CDN buffer delays
- Media segment duration
- Player behavior
  - Buffering
  - Playhead positioning
  - Resilience

Current Implementations

- Smooth Streaming: 2-second segments, usually 10 seconds of latency
- DASH: 2-second segments supported by most players, usually 8 to 10 seconds of latency
- HLS
  - Until mid-2016: 10-second segments, usually 30 seconds of latency
  - Since mid-2016: 6-second segments, usually 18 to 20 seconds of latency
  - Safari Mobile in iOS11: Autostart for live streams and support for short segments
  - App Store & iOS applications: 6-second segments are recommended but not mandatory

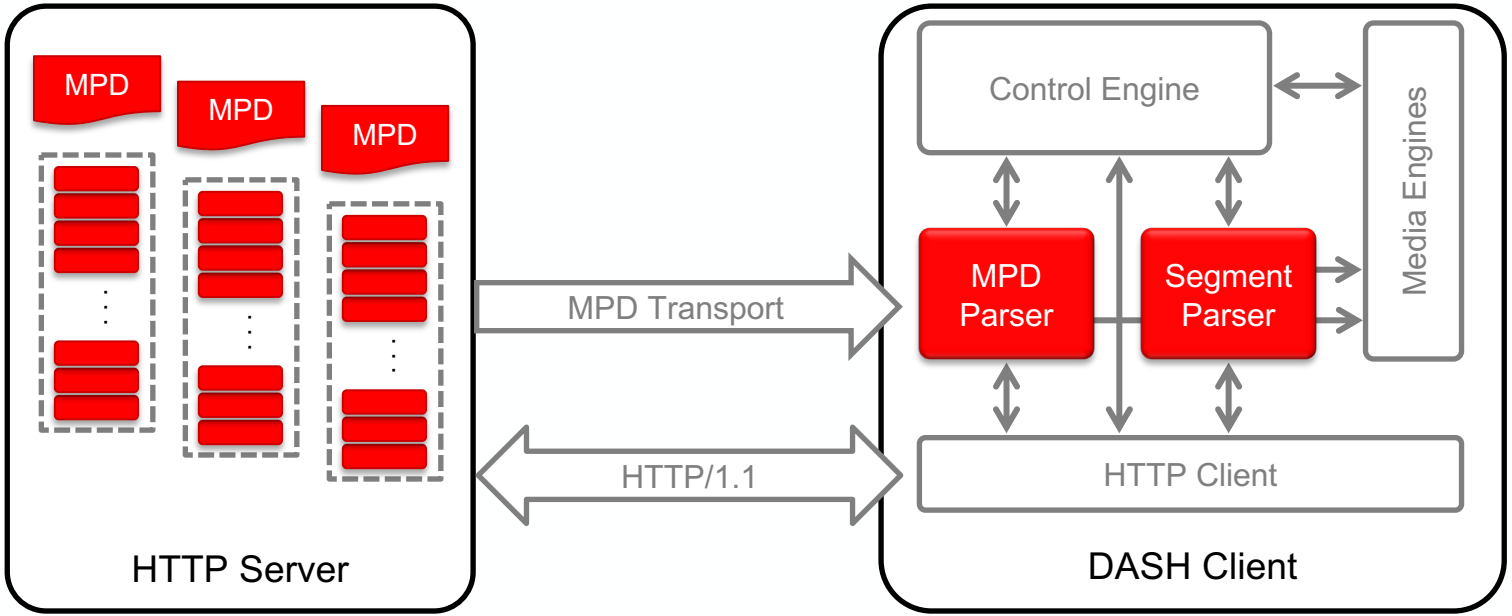
Algorithms and Formats for Adaptive Streaming

Status of MPEG DASH



Scope of MPEG DASH

Shown in Red



Brief History of DASH at MPEG

23009-1: Media Presentation Description and Segment Formats

- Amd. 1: NTP sync, extended profiles
- Amd. 2: SRD, URL parameter insertion, role extensions
- Amd. 3: External MPD link, period continuity, generalized HTTP header extensions/queries
- Amd. 4: TV profile, MPD chaining/resetting, data URLs in MPD, switching across adaptation sets
- 3rd edition (FDIS) in ballot
- Amd. 5 (WiP): Device information, quality equivalence descriptor, timed text roles, announcing popular content, flexible IOP signaling, early available periods, signaling missing/alternative segments

23009-2: Conformance and Reference Software

- 2nd edition was published in Oct. 2017
- Amd. 1 (WiP): SAND conformance rules

23009-3: Implementation Guidelines (Informative)

- 3rd edition (WD) is in progress

Brief History of DASH at MPEG

23009-4: Segment Encryption and Authentication

- 2nd edition (FDIS) in ballot

23009-5: Server and Network Assisted DASH (SAND)

- 1st edition, published in Feb. 2017

23009-6: DASH over Full Duplex HTTP-Based Protocols (FDH)

- 1st edition, published in Dec. 2017

23009-7: Delivery of CMAF Contents with DASH (Informative)

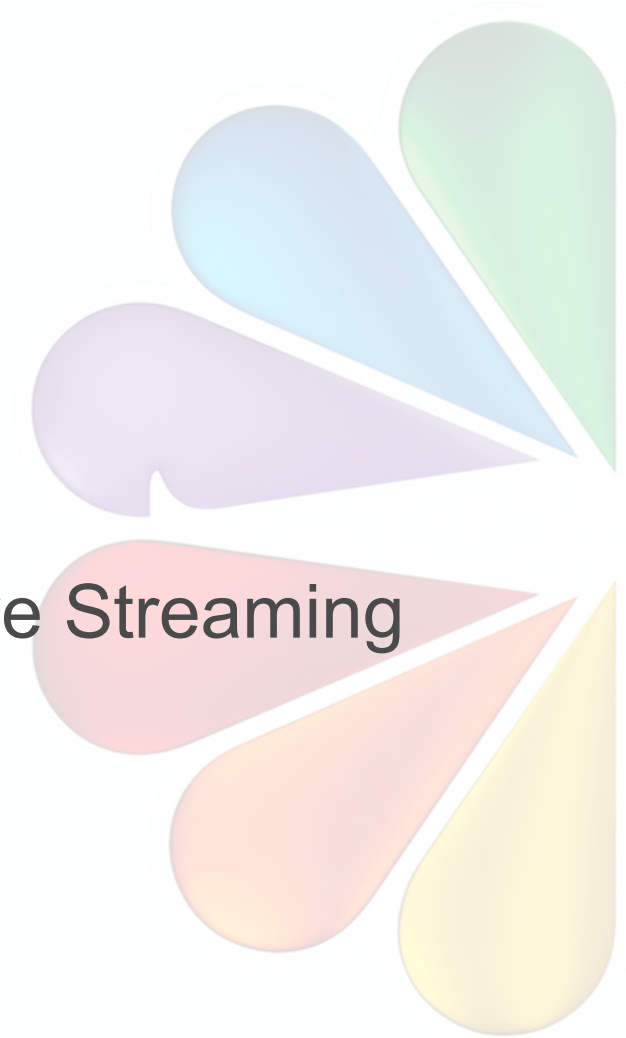
- 1st edition (WD) is work in progress

Ongoing Work as of MPEG 123 (July 2018)

- Technologies under Consideration (w17812)
  - Usage of HEVC tile tracks in DASH
  - Annotation and client model for content selection
  - Signaling for quality control
  - Announcing popular content in DASH
  - Using segment templates for forensic watermarking
  - Using DASH MPD chaining for mid-roll ads
  - DASH playlist description
  - Mixed MPD
  - Event processing model
  - Patch method for MPD updates

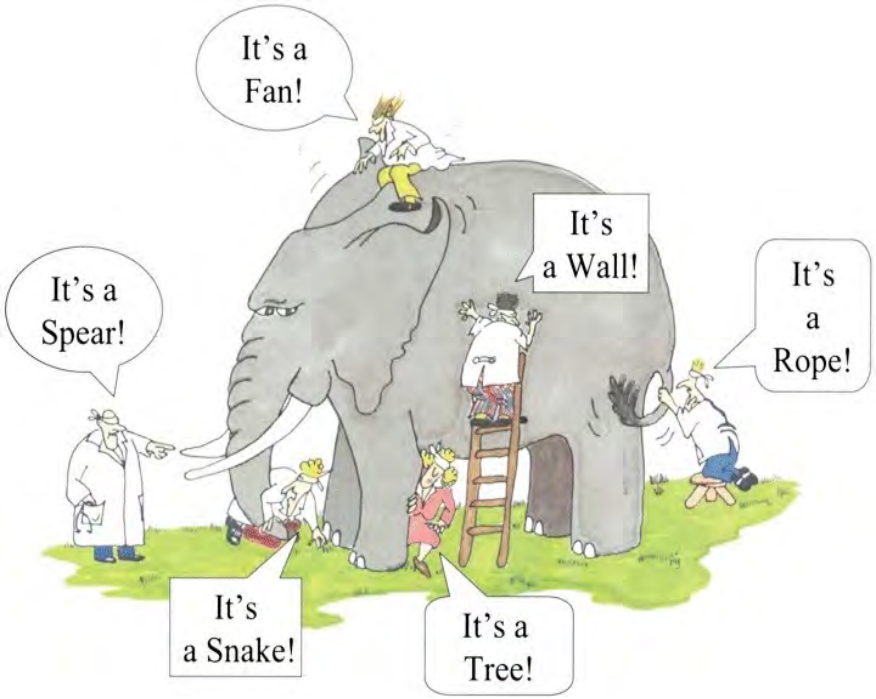
Algorithms and Formats for Adaptive Streaming

Improving QoE in Streaming



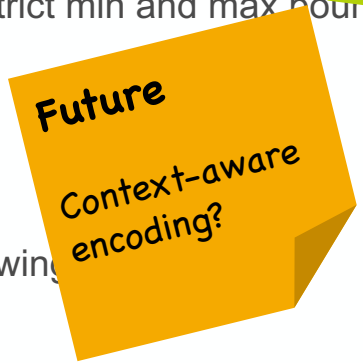
What is Quality?

Many Definitions Do Exist



Nomenclature of Rate Control

- ABR: Adaptive bitrate
  - Misnomer, refers to adaptive streaming over HTTP
- CBR: Constant bitrate encoding
  - The decoder's buffer is filled at a constant rate
- (True) VBR: Variable bitrate encoding
  - The decoder's buffer is filled at a non-constant rate
- Capped VBR: How VBR is implemented in practice
  - The decoder's buffer is filled at a non-constant rate with strict min and max bounds
- Title/content-based (or content-aware) encoding
  - Choosing the bitrate ladder based on the content
- Context-aware encoding
  - Advanced optimizations based on viewer, display and viewing

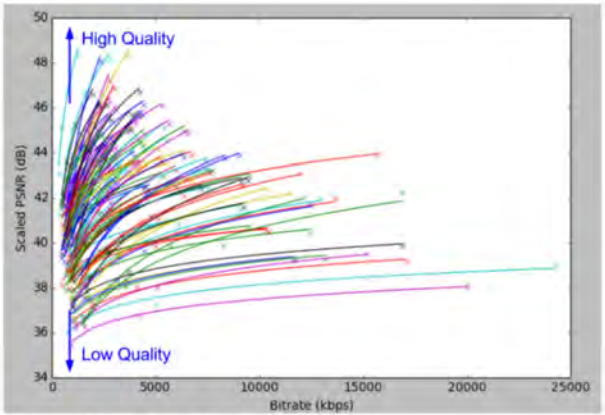


Content-Based (Content-Aware) Encoding

Picking the Bitrate Ladder Based on the Content

Increased bandwidth brings new opportunities

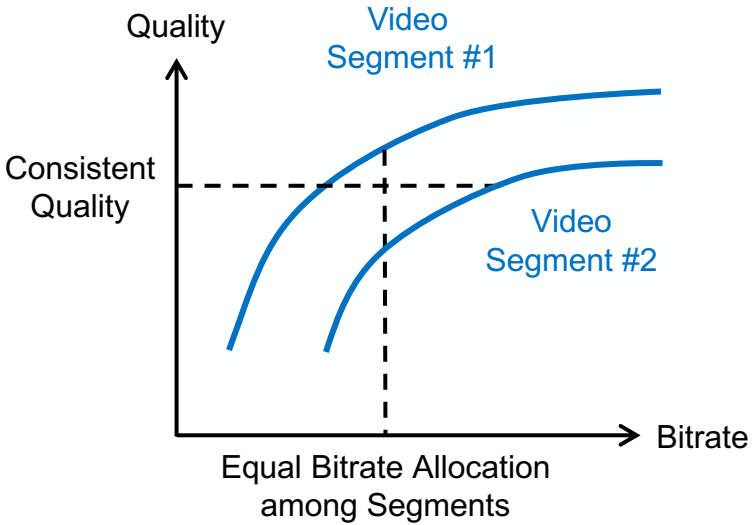
Drives overall maturity of IP/ABR delivery technologies, introduces new opportunities – Per Title Encode Optimization



Source: Netflix Tech Blog, December 2015

Content-aware encoding gives us fairness in quality as opposed to fairness in bitrate

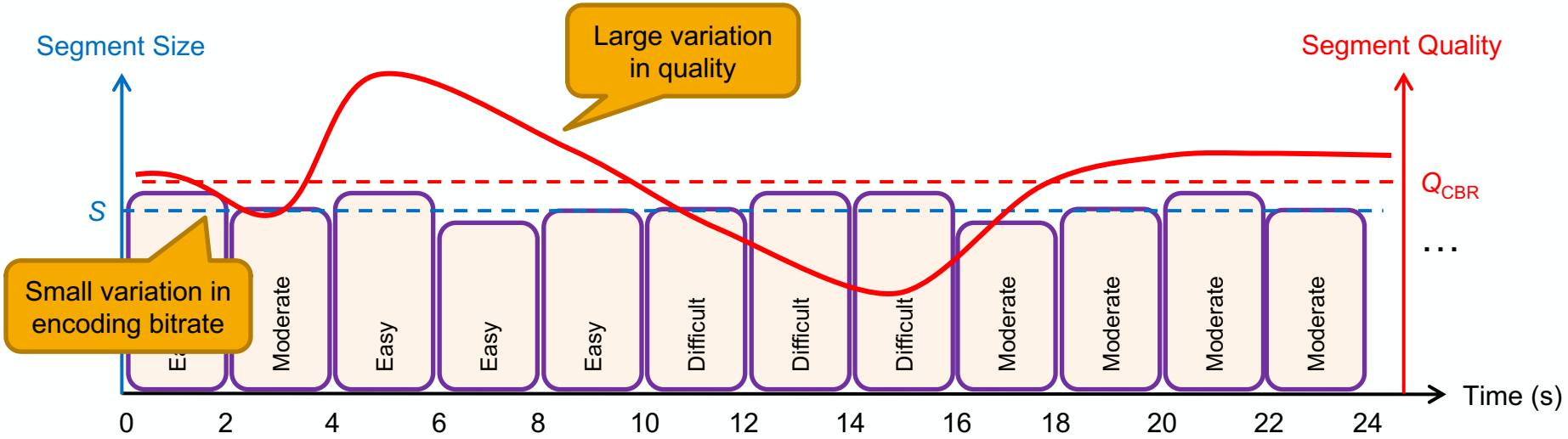
Segments Have Different Complexities



- If the following holds true
  - Segments are ~CBR encoded
  - Client fetches segments based on bitrate information only
- Then, viewer QoE will vary because of
  - Low-motion/complexity vs. high-motion/complexity scenes
  - Upshifts and downshifts dictated by the adaptation logic

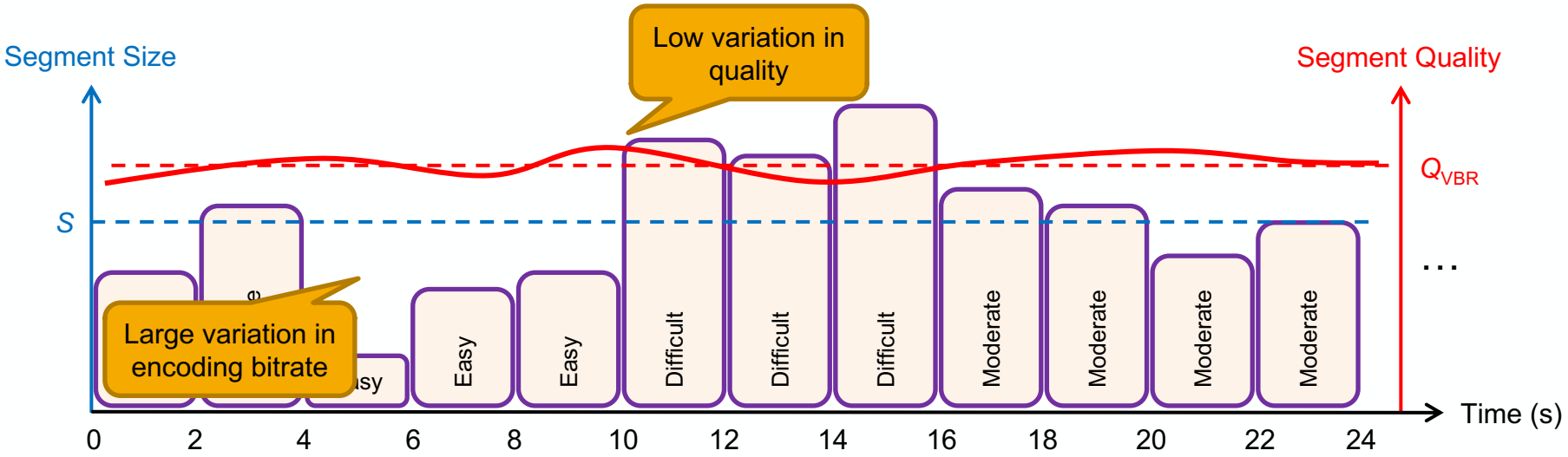
Adaptation Feature Does Not Deliver Consistent Quality

Guidelines Limited Bitrate Variability to (Mostly) 10% So Far



If there is something worse than having to watch a video at a lousy quality, it is to watch that video with varying quality

What If We Encode in a More Subtle Fashion?

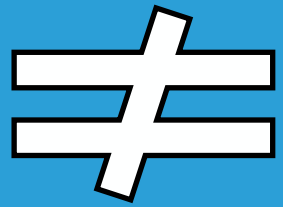


While we spend the same total amount of bits, we not only increase average quality but also reduce quality variation

HLS authoring spec for ATV allows 2x capping rate for VoD. For linear content, variability is limited to 10-25% range.

Generating VBR-encoded segments is easy, but streaming them is not!

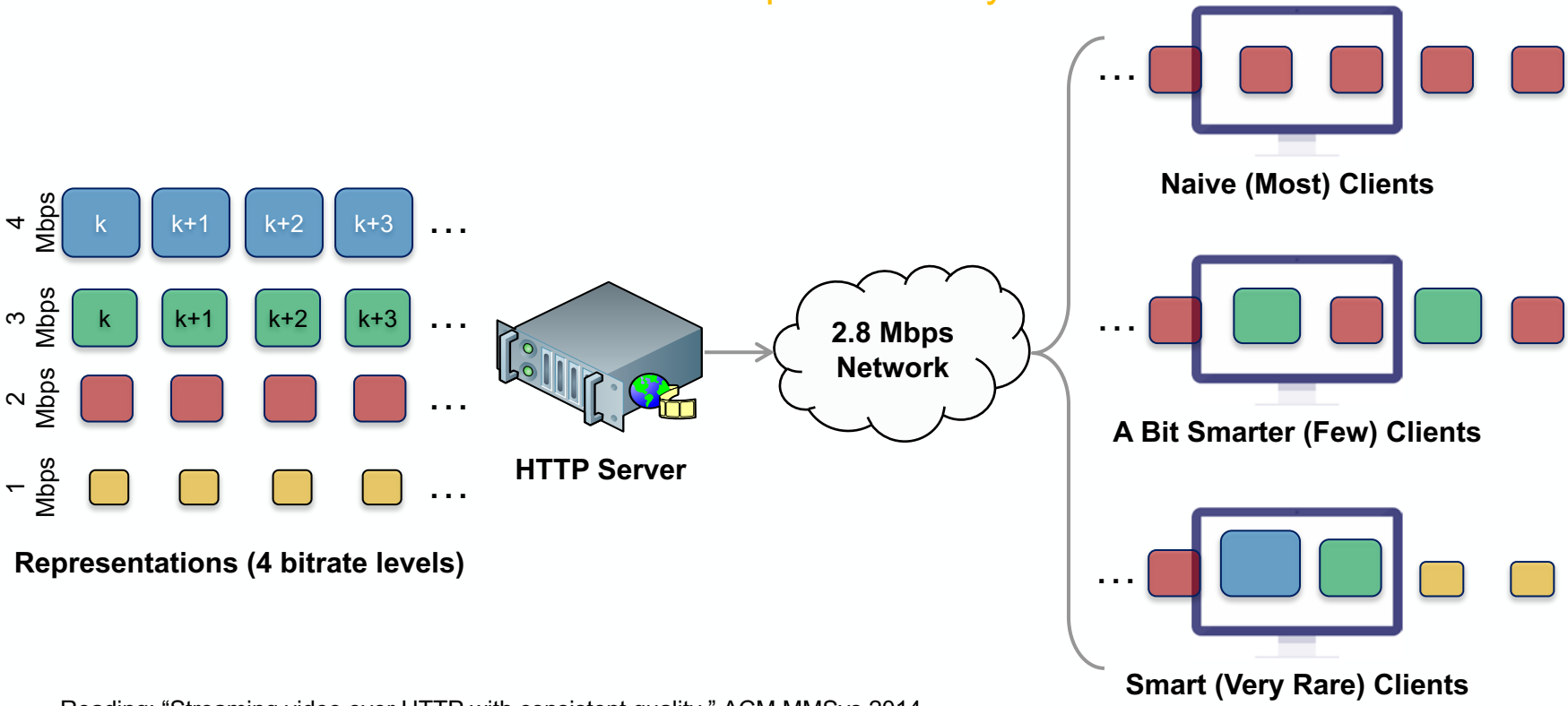
Content-aware Encoding



Content-aware Streaming

What If the Content is Already CBR Encoded

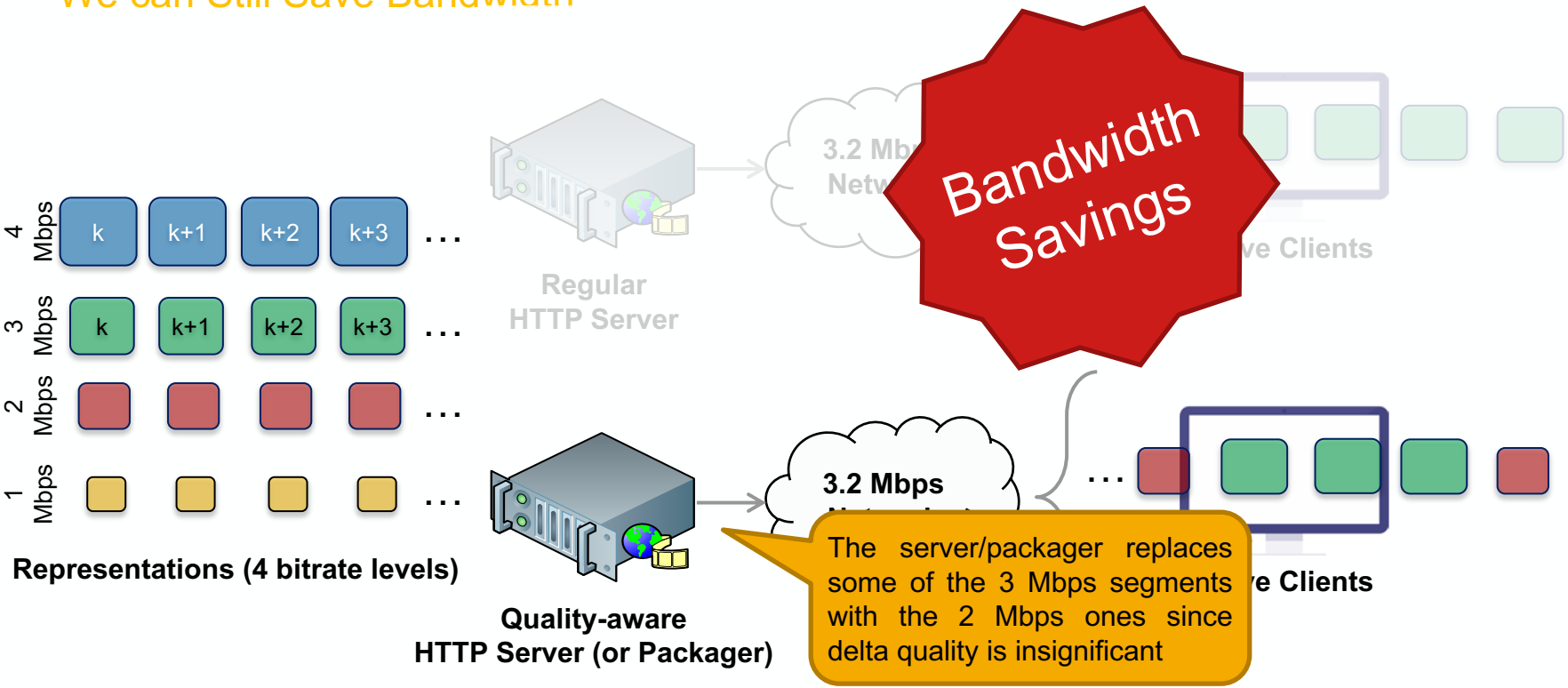
We can Still Save Bandwidth and/or Improve Quality



Reading: “Streaming video over HTTP with consistent quality,” ACM MMSys 2014

What If There is No Smartness in the Client

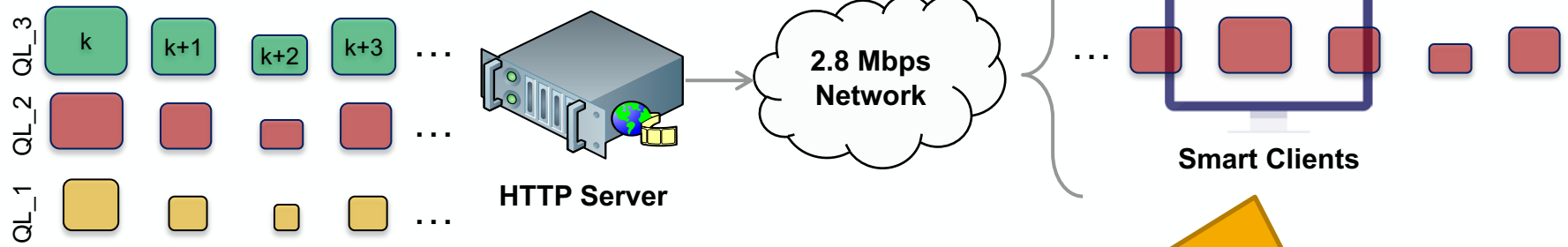
We can Still Save Bandwidth



Reading: “More juice less bits: content aware streaming,” ACM MMSys 2016

What If the Content is VBR Encoded

The resolution stays the same but the encoding rate varies in a given representation (per quality level)

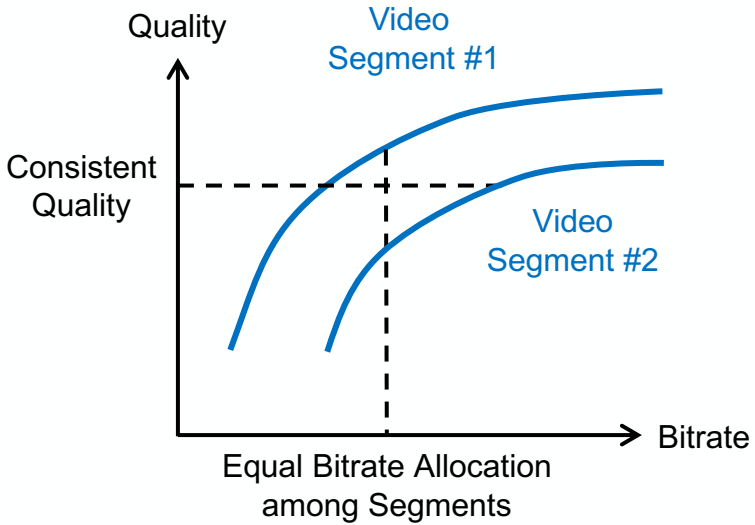


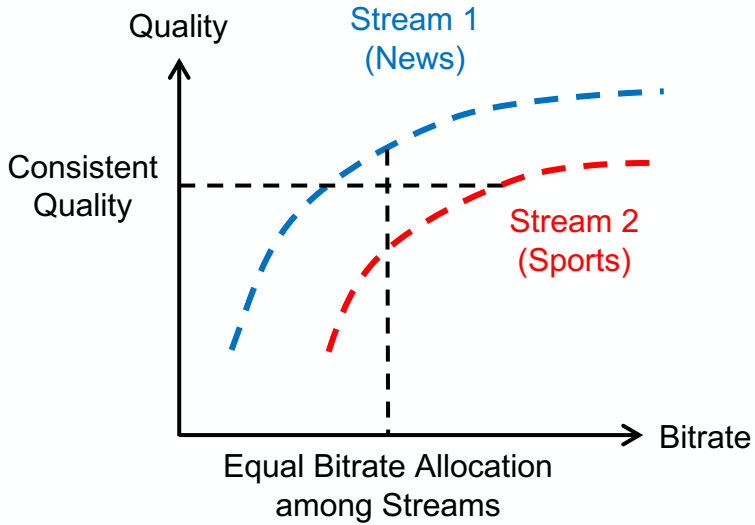
Representations (3 quality levels)

The client streams the highest consistent-quality video without draining its buffer while respecting the available bandwidth

Extending the Idea to Optimization across Streams

- Same principle applies to both:
  - In-stream: Temporal bit shifting between segments
  - Across-streams: Bit shifting between streams sharing a bottleneck link





Reading: "Spending quality time with the Web video," IEEE Internet Comput., 2016

Visit http://ali.begen.net for More Tutorials and Papers

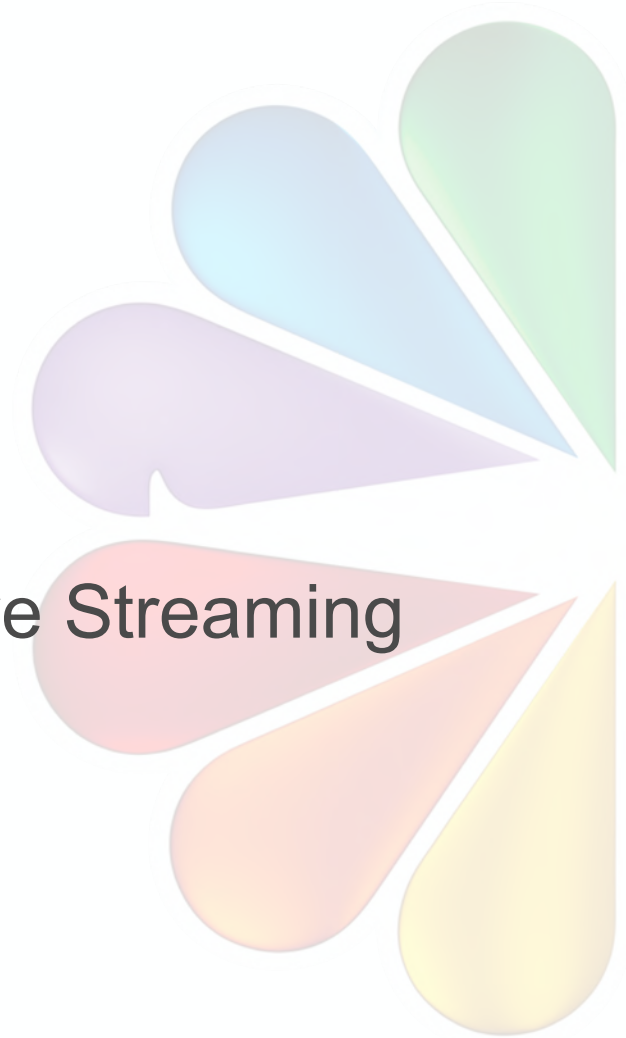
“Television! Teacher, mother, secret lover”

– Homer Simpson

Thanks to T. Stockhammer, J. Simmons, K. Hughes, C. Concolato, S. Pham, W. Law, N. Weil and many others for helping with the material

Algorithms and Formats for Adaptive Streaming

Backup Slides



Source Code for Client Implementations

- DASH Industry Forum
  - http://dashif.org/software/
- JW Player
  - https://github.com/jwplayer/jwplayer
- Other Open Source Implementations/Frameworks
  - http://dash.itec.aau.at/
  - http://gpac.wp.mines-telecom.fr/
  - https://github.com/google/shaka-player
  - https://github.com/video-dev/hls.js/
  - http://streaming.university/GTA/
- TNO's SAND Demo: https://github.com/tnomedialab/sand

DASH Datasets

- DASH (http://dash.itec.aau.at/)
  - http://www-itec.uni-klu.ac.at/dash/?page_id=207
- Distributed DASH
  - http://www-itec.uni-klu.ac.at/dash/?page_id=958
- Multi-Codec DASH
  - http://www-itec.uni-klu.ac.at/dash/?page_id=1619
- DASH SVC
  - http://concert.itec.aau.at/SVCDataset/
- UHD HEVC DASH
  - http://download.tsi.telecom-paristech.fr/gpac/dataset/dash/uhd/
- iVID-Datasets for AVC and HEVC
  - https://www.ucc.ie/en/misl/research/datasets/ivid_dataset/
- AVC and HEVC UHD 4K DASH
  - https://www.ucc.ie/en/misl/research/datasets/ivid_uhd_dataset/
- Open Dataset from ITU-T P.1203 Standardization
  - https://github.com/itu-p1203/open-dataset

Other Datasets

- SWAPUGC
  - https://github.com/emmanouil/SWAPUGC
- A 4G LTE Dataset with Channel and Context Metrics
  - https://www.ucc.ie/en/misl/research/datasets/ivid_4g_lte_dataset/
- A Multi-Carrier Mobile Geo-Communication Dataset
  - https://dl.acm.org/citation.cfm?id=3193572
- ODIs Saliency Maps
  - https://drive.google.com/file/d/1hbPDS2FqzZRqpAbhRurL7L-rT0bjIZeB/view
- Exploring User Behaviors in VR
  - https://wuchlei-thu.github.io/
- 360° Videos Head Movements
  - http://dash.ipv6.enstb.fr/headMovements/
- 360° Video Viewing in Head-Mounted VR
  - https://dl.acm.org/citation.cfm?id=3192927