



# Watching Video over the Web: Adaptive Streaming over HTTP

Ali C. Begen  
abegen@cisco.com

Slides will be posted at <http://ali.begen.net>

# Agenda

- **State of Technology**

  - Overview of video delivery methods and adaptive streaming over HTTP

  - How the current major streaming players work

  - Some experimental and simulation-based results

- **State of Standards**

  - Brief overview of efforts in MPEG DASH and 3GPP SA4

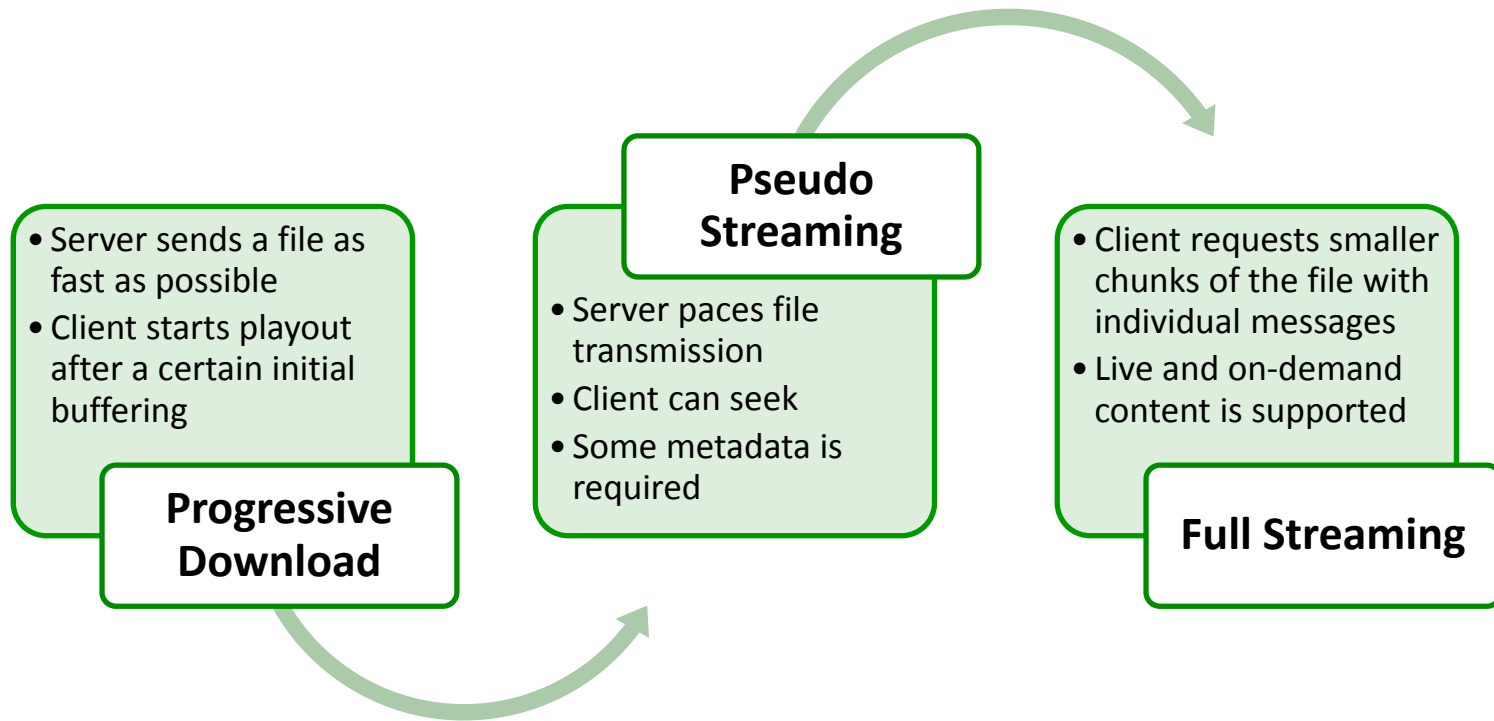
- **Research Directions**

# Push and Pull-Based Video Delivery

	<b>Push-Based Delivery</b>	<b>Pull-Based Delivery</b>
<b>Source</b>	Broadcasters and servers like Windows Media Apple QuickTime, RealNetworks Helix Cisco CDS/DCM	Web/FTP servers such as LAMP Microsoft IIS Adobe Flash RealNetworks Helix Cisco CDS
<b>Protocols</b>	RTSP, RTP, UDP	HTTP, RTMPx, FTP
<b>Video Monitoring and User Tracking</b>	RTCP for RTP transport	(Currently) Proprietary
<b>Multicast Support</b>	Yes	No

# Pull-Based Video Delivery over HTTP

## Progressive Download vs. Pseudo and Full Streaming



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

Client consumption rate may be limited by real-time constraints as opposed to just bandwidth availability

Server transmission rate (loosely or tightly) matches to the consumption rate of the client

# Adaptive Streaming over HTTP

## Overview

- **A Hybrid of Download and Streaming**

  - Imitates streaming via short downloads

    - Clients progressively download the desired portion in small chunks so bandwidth is not wasted

    - Clients stream so we can monitor consumption and track them

- **Adaptation to Dynamic Conditions and Device Capabilities**

  - Adapts to dynamic conditions anywhere on the path through the Internet and/or home network

  - Adapts to display resolution, CPU and memory resources of the client

- **Improved Quality of Experience**

  - Enables faster start-up and seeking (compared to progressive download), and quicker buffer fills

  - Reduces skips, freezes and stutters

- **Use of HTTP**

  - Well-understood/supported naming/addressing approach, and authentication/authorization infrastructure

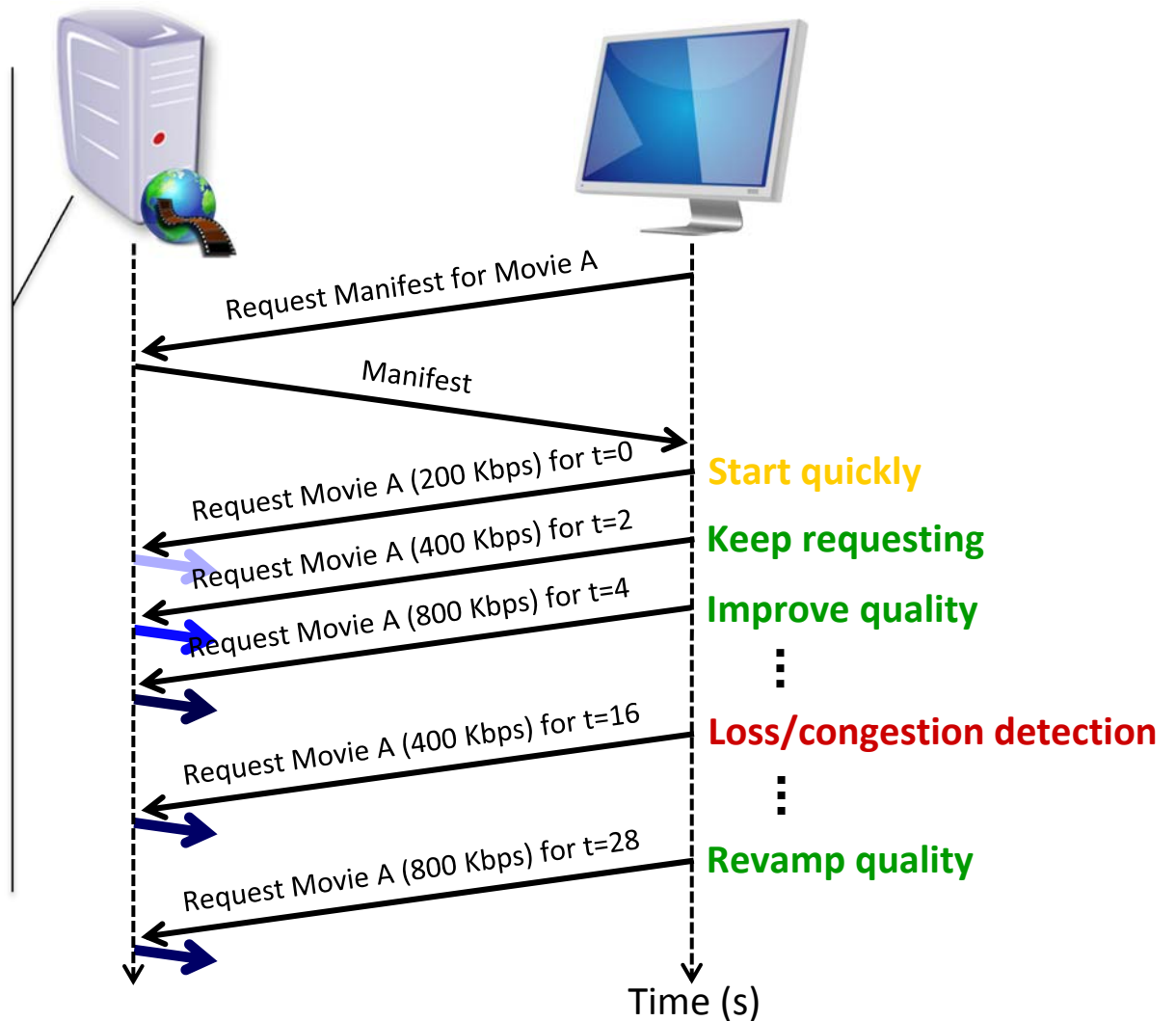
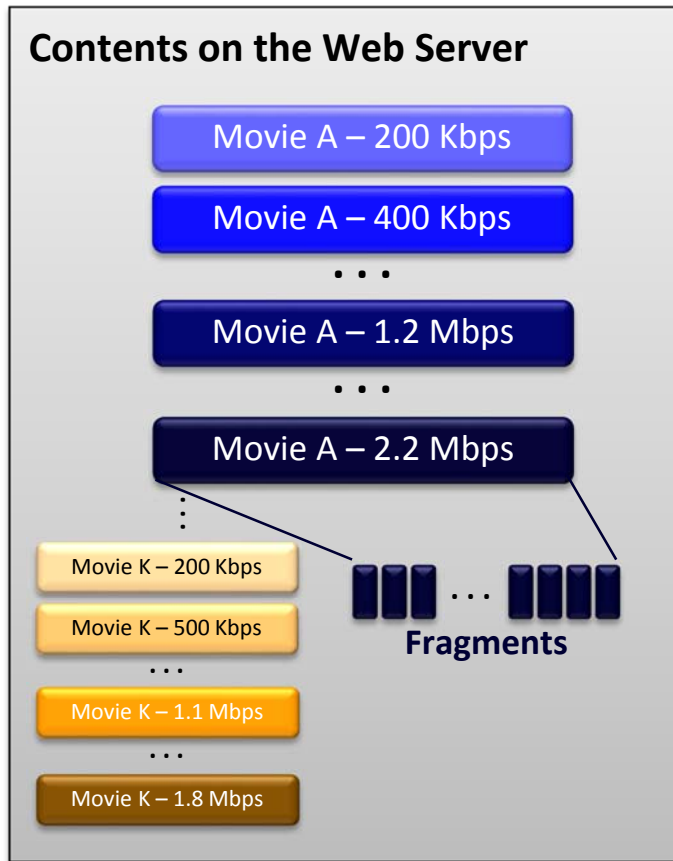
  - Provides easy traversal for all kinds of middleboxes (e.g., NATs, firewalls)

  - Leverages existing HTTP caching infrastructure (Cheaper CDN costs)

    - E.g., Akamai, Level 3, Limelight

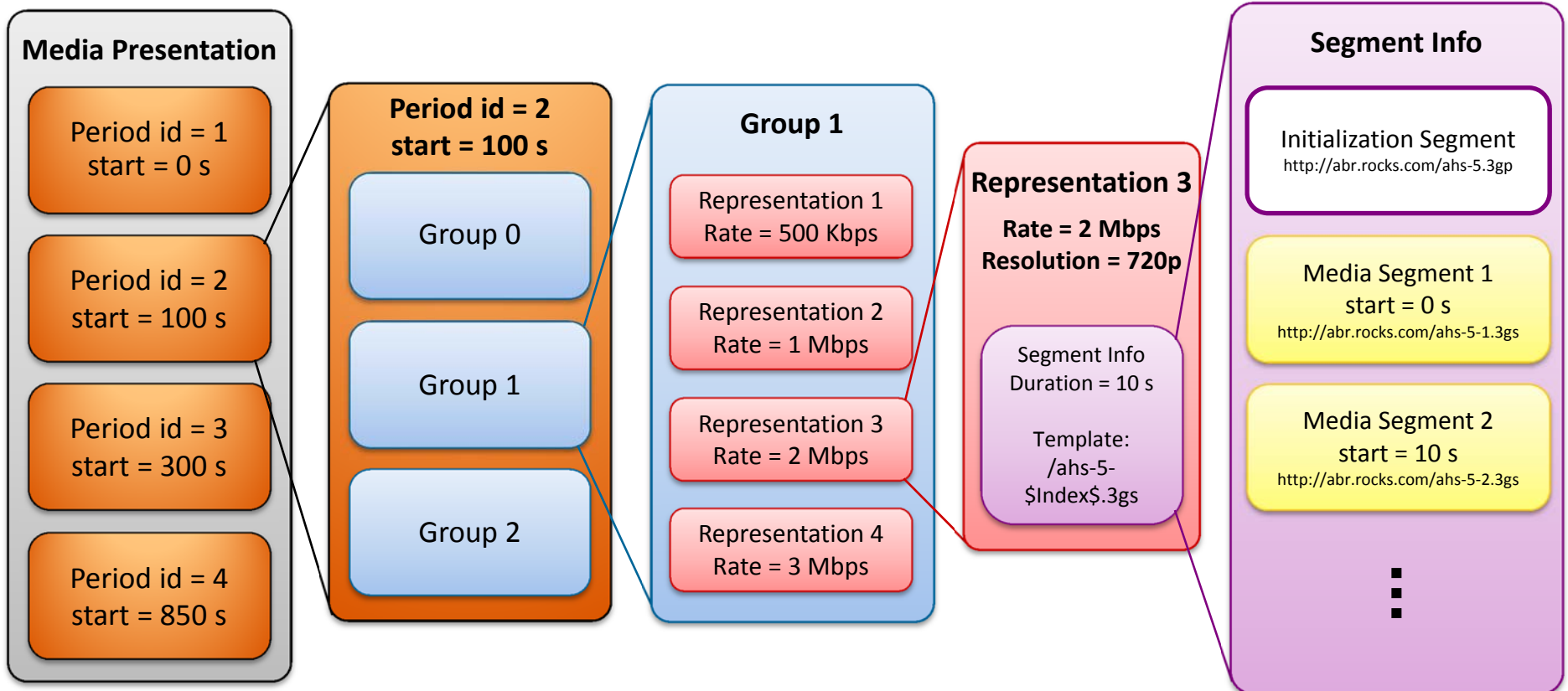
# Adaptive Streaming over HTTP

## Multi-Bitrate Encoding and Other Concepts



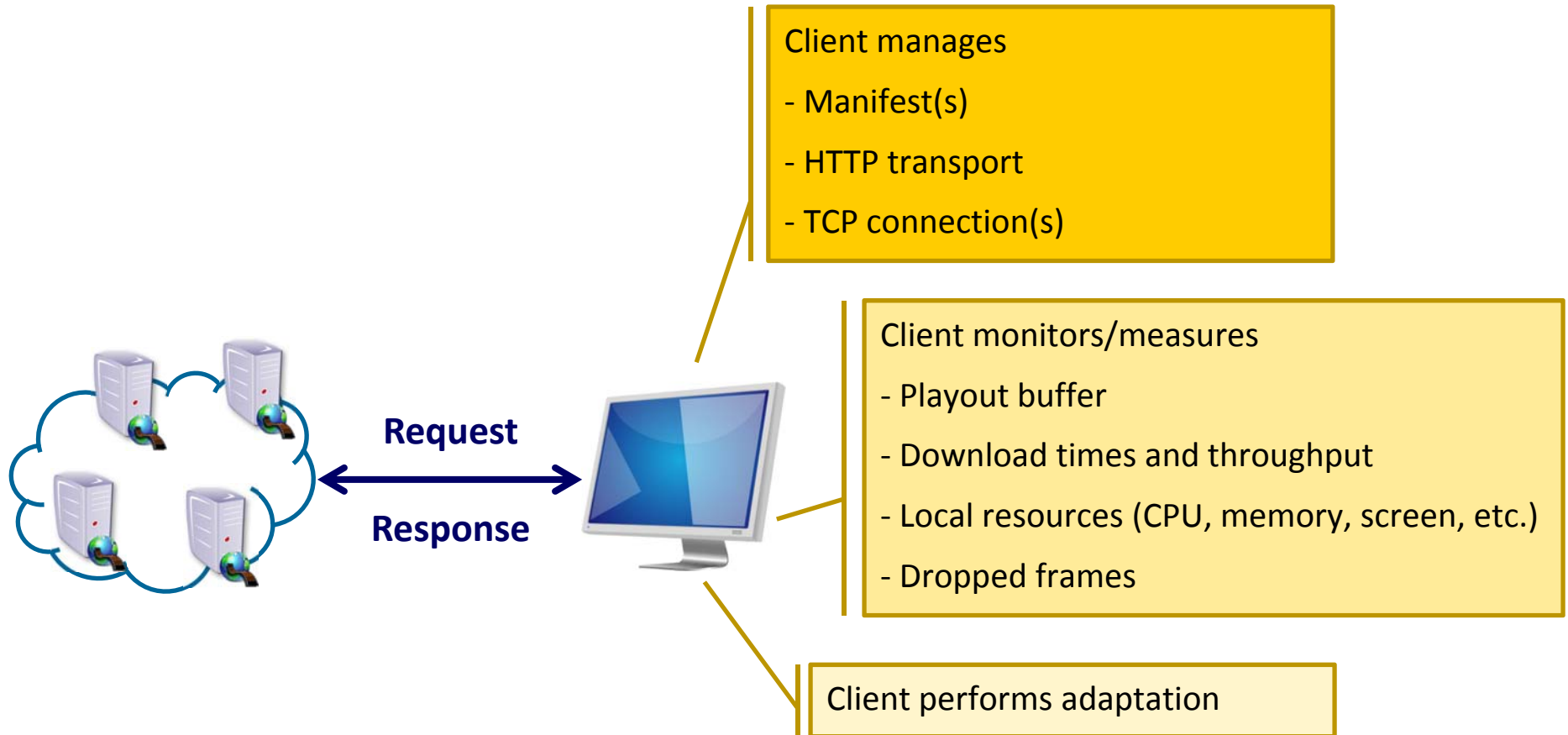
# Manifest – Media Presentation Description (MPD)

## List of Accessible Segments and Their Timings



# Adaptive Streaming over HTTP

## Smart Clients



# Major Players in the Market

- **Microsoft Smooth Streaming**

<http://www.iis.net/expand/SmoothStreaming>

- **Netflix**

<http://www.netflix.com/NetflixReadyDevices>

- **Adobe HTTP Dynamic Streaming**

<http://www.adobe.com/products/httpdynamicstreaming/>

- **Move Adaptive Stream (Acquired by Echostar)**

<http://www.movenetworks.com>

- **Apple HTTP Live Streaming**

<http://tools.ietf.org/html/draft-pantos-http-live-streaming>

- **Others**

Octoshape Infinite Edge

Widevine Adaptive Streaming (Acquired by Google)

Vidiator Dynamic Bitrate Adaptation

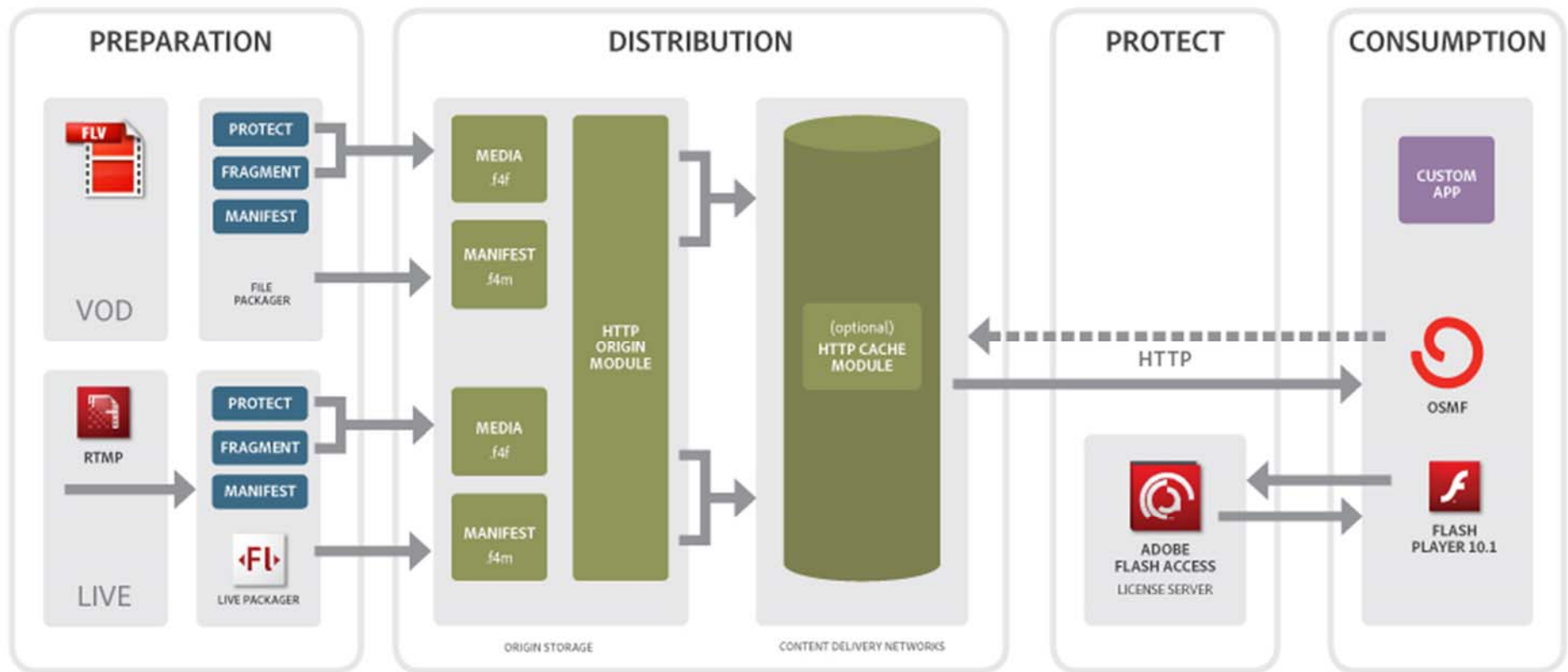


# Example (Microsoft Smooth) Player Showing Adaptation



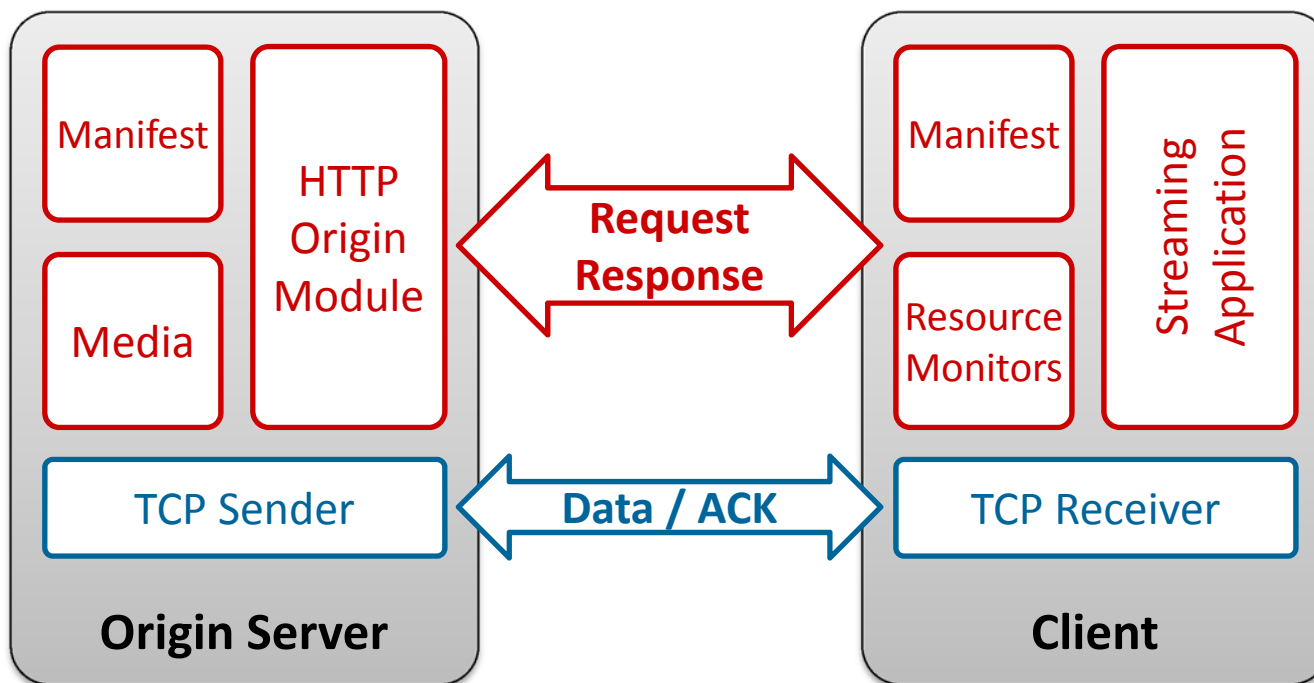
# Adaptive Streaming over HTTP

## Example Workflow from Adobe



Source: <http://www.adobe.com/products/httpdynamicstreaming/>

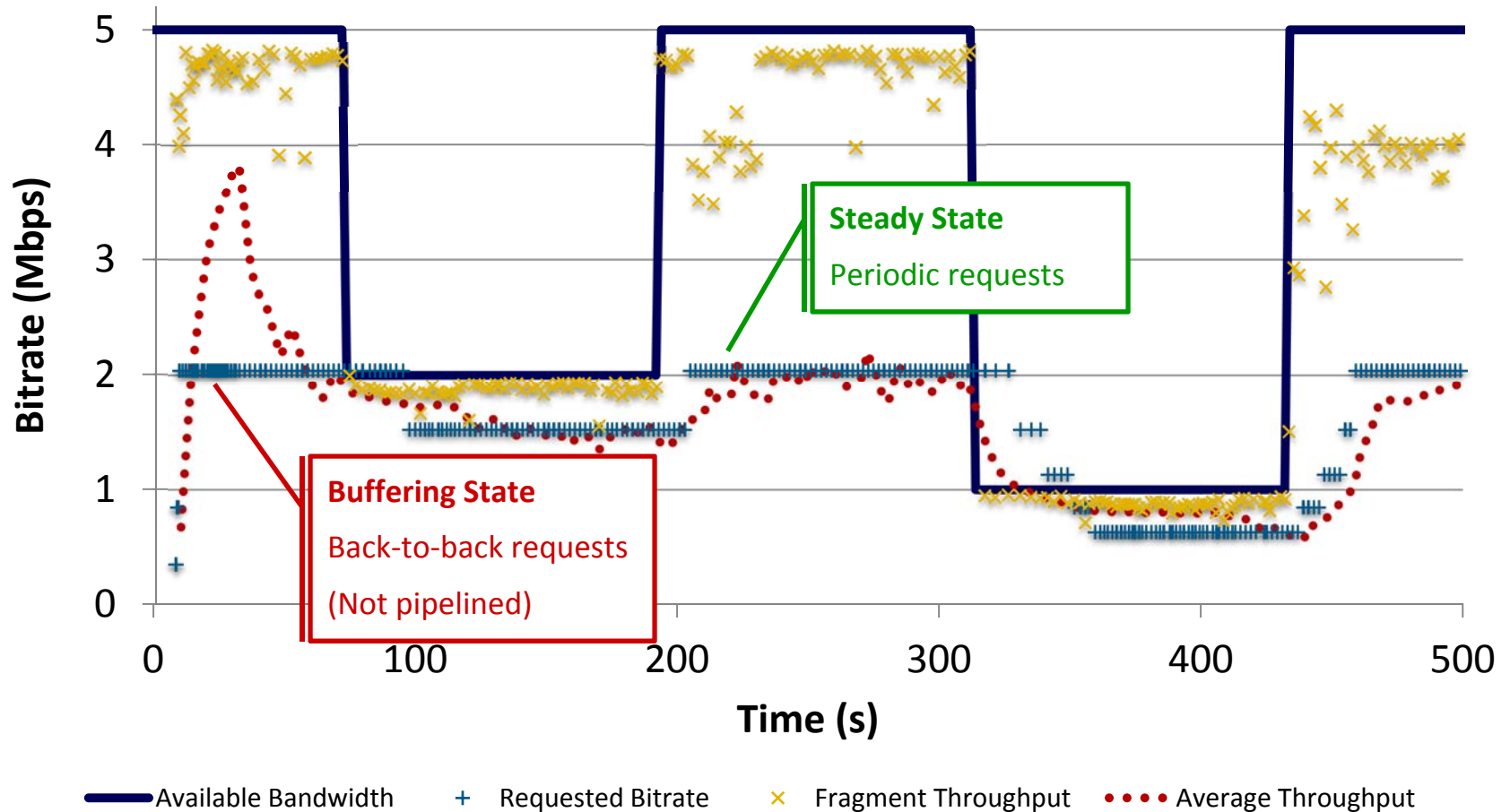
# Inner and Outer Control Loops



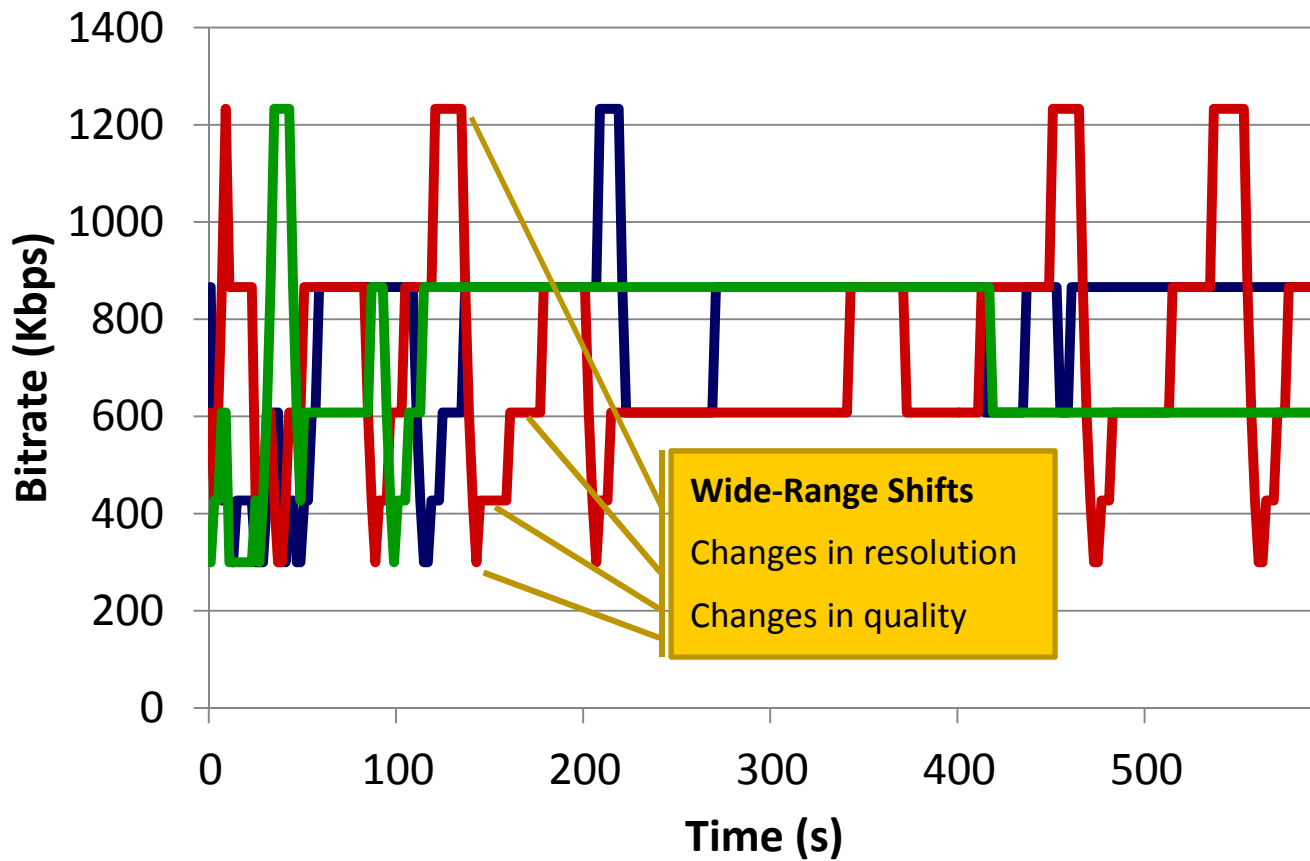
There could be multiple TCPs destined to potentially different servers  
(Cache servers are similar except that they do not have the origin modules)

# Interaction of Inner and Outer Control Loops

## Microsoft Smooth Streaming Experiments

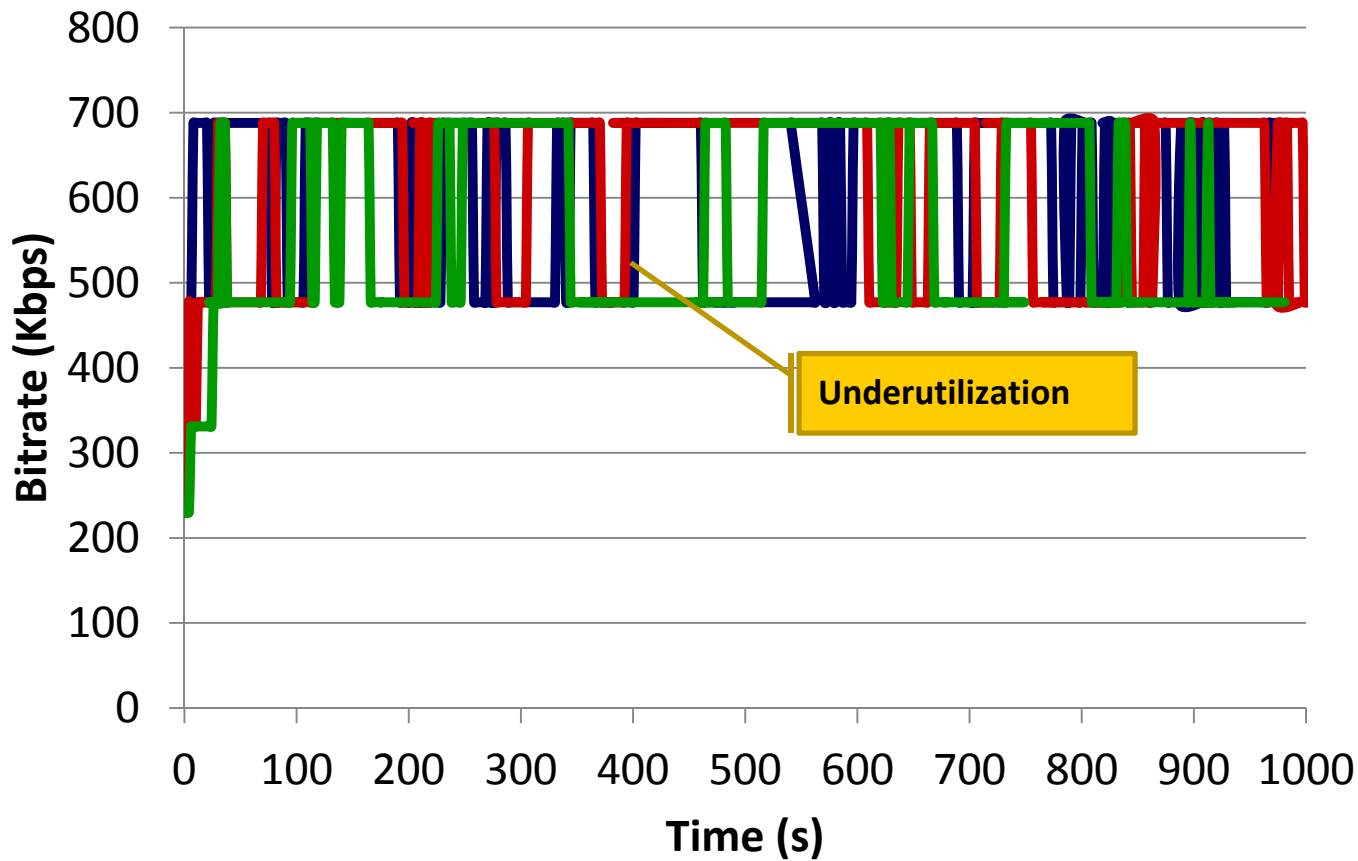


# 10 Microsoft Smooth Clients Sharing 10 Mbps Link Streaming “Big Buck Bunny” (Three Clients are Shown)



Available Profiles: 300, 427, 608, 866, 1233, 1636, and 2436 Kbps

# 10 Microsoft Smooth Clients Sharing 10 Mbps Link Streaming “World in Motion” (Three Clients are Shown)



Available Profiles: 230, 331, 477, 688, 991, 1427, 2056 Kbps

# 100 Simulated Clients Sharing 100 Mbps Link

## Streaming “World in Motion”

- **Aggressive Clients**

Stay at bitrate R provided that recent download speeds are at least  $0.9xR$

- **Conservative Clients**

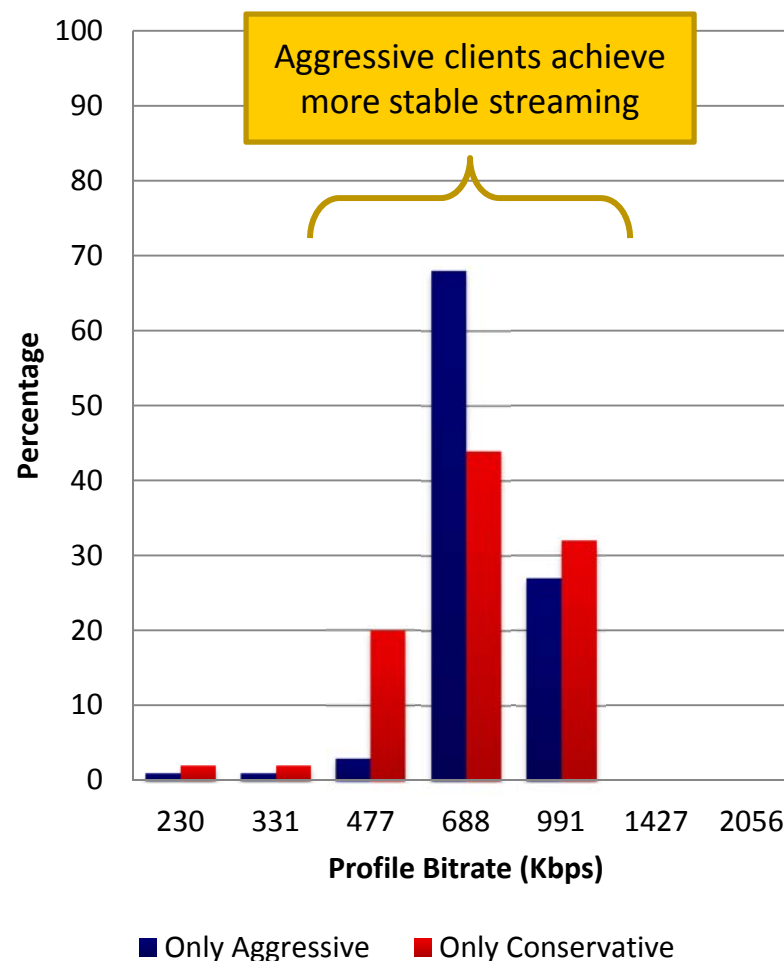
Stay at bitrate R provided that recent download speeds are at least  $1.2xR$

- **Other Parameters**

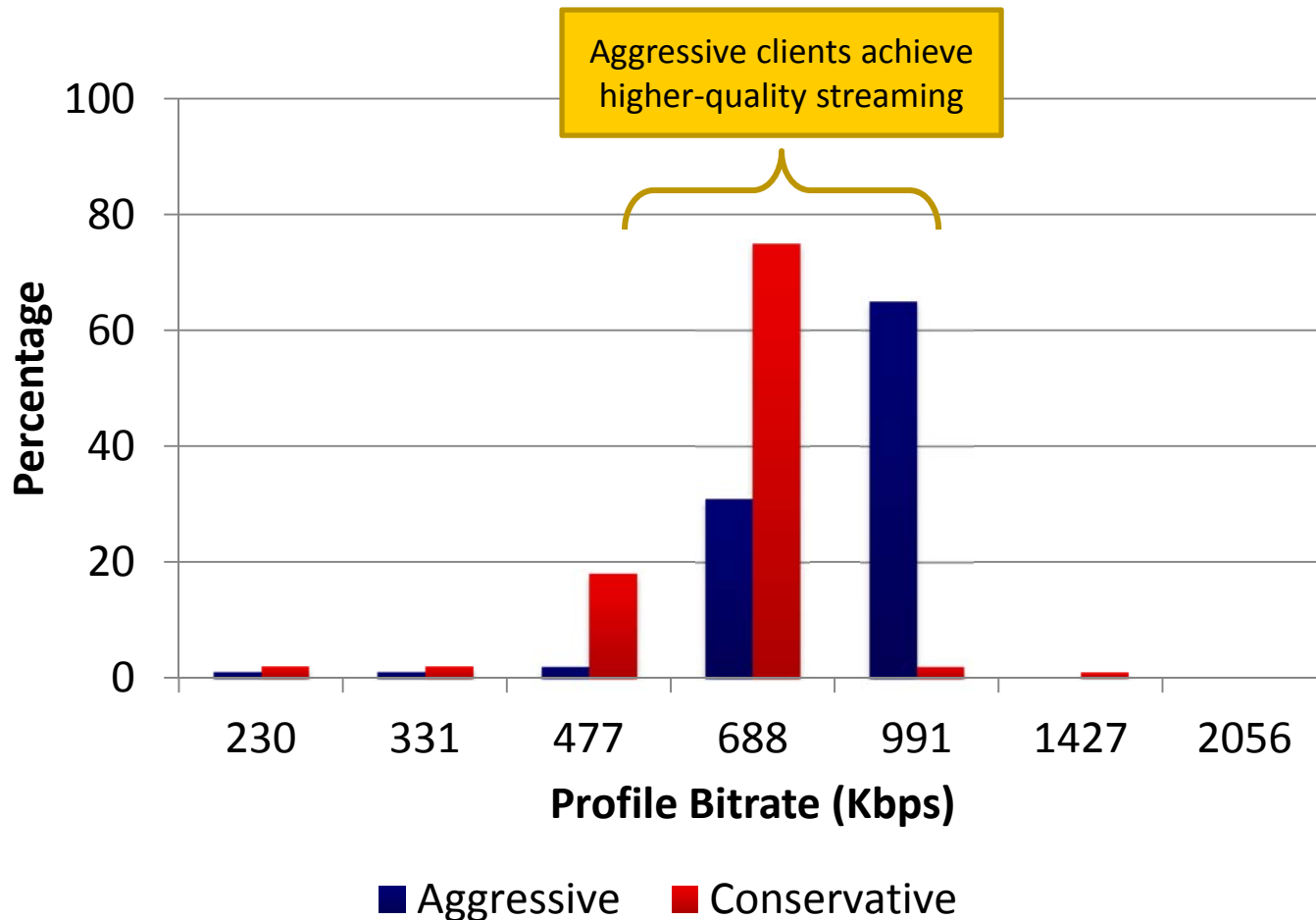
Fragments: 2 s

Minimum Buffer Threshold: 4 s

Maximum Buffer: 45 s



# 50-50 Mixed Simulated Clients Sharing 100 Mbps Link Streaming “World in Motion”





# State of Standards



# MPEG – Dynamic Adaptive Streaming over HTTP (DASH)

## Design Principles

- DASH defines the MPD format and delivery formats using ISO BMFF and MPEG2-TS
- DASH is not a system, protocol or codec specification and does not specify

### Content Provisioning

- Size and duration of segments

- Number and bitrates of representations, frequency of RAPs

### Normative Client Behavior

- Download times of segments

- Switching between different representations

### Transport of MPD

- DASH supports

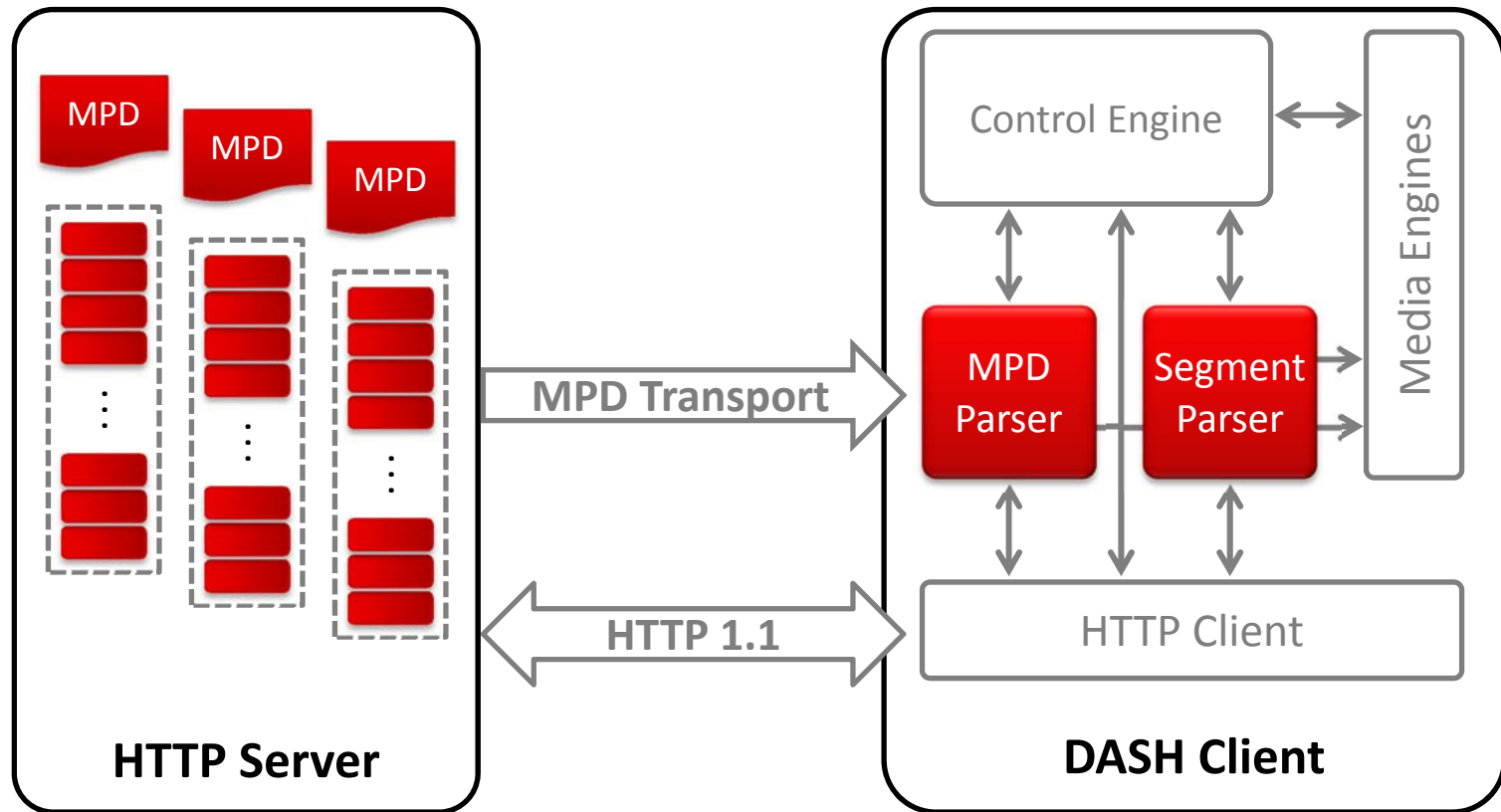
- Live, on-demand and time-shifted content delivery and trick modes

- Splicing and ad insertion

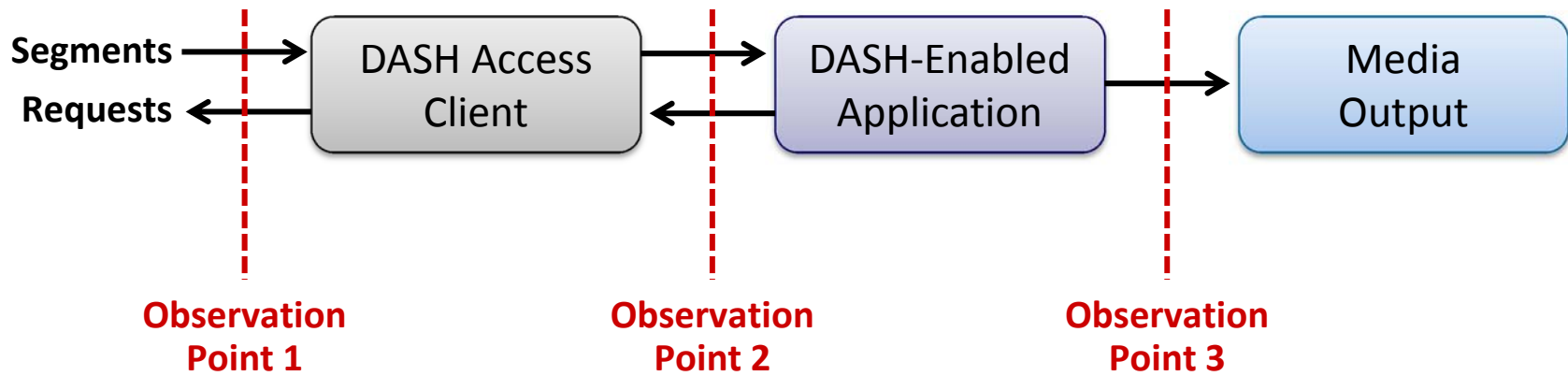
- Byte-range requests

- Content descriptors for protection, accessibility and rating

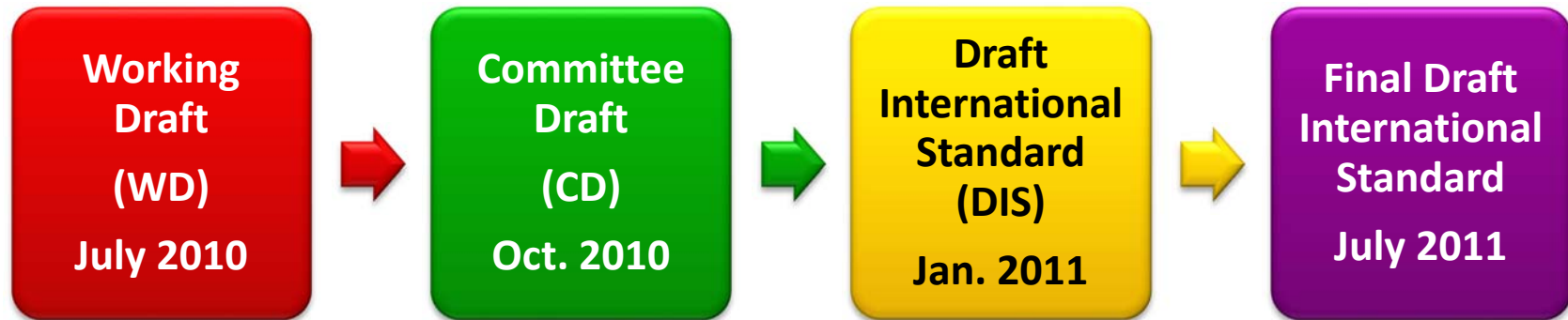
# Scope of MPEG DASH



# Collection of Quality Metrics



# Timeline in MPEG



## Timeline in 3GPP SA4

- **SA4 started its activity mid 2009 and completed specification work early March 2010 for Release 9**

Integrated in Transparent End-to-end Packet-switched Streaming Service (PSS)

TS 26.234 (PSS Codecs and Protocols)

TS 26.244 (3GPP File Format – 3GP)

TS 26.247 (PD and Dynamic Adaptive Streaming over HTTP – 3GP-DASH)

Describes (XML-based) media presentation description (MPD) file

Defines the segment format and the delivery protocol

Provides informative description for an adaptive streaming client

- **SA4 has been doing bug fixing and maintenance since April 2010**

Editorial corrections and clarifications

Alignment sought with OIPF

Work started towards Release 10 (to be published mid 2011)

# Links for Organizations and Specs

- **3GPP**

<http://ftp.3gpp.org/specs/html-info/26247.htm>

- **MPEG DASH**

ISO/IEC 23001-6 and ISO/IEC 14496-12:2008/DAM 3 available at

[http://mpeg.chiariglione.org/working\\_documents.htm](http://mpeg.chiariglione.org/working_documents.htm)

Mailing List: <http://lists.uni-klu.ac.at/mailman/listinfo/dash>

- **W3C Web and TV Interest Group**

<http://www.w3.org/2011/webtv/>

- **DECE UltraViolet**

<http://www.uvvu.com/>

- **IETF httpstreaming Discussion List**

<https://www.ietf.org/mailman/listinfo/httpstreaming>

- **OIPF Volume 2a - HTTP Adaptive Streaming**

<http://www.openiptvforum.org/specifications.html>



# Research Directions



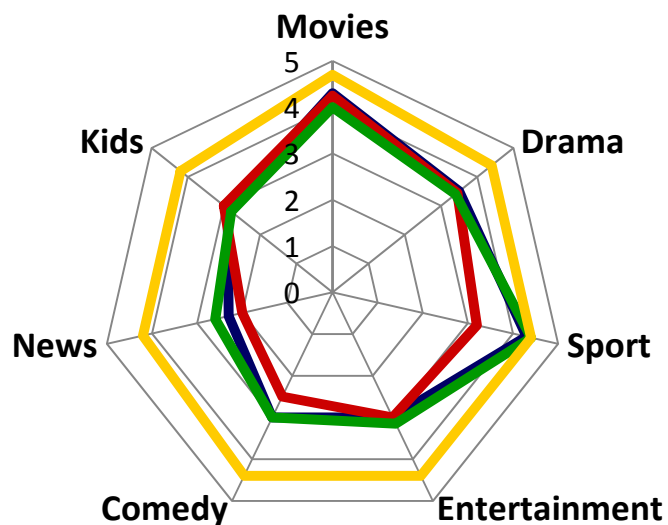
# Content Encoding, Packaging and Segmentation

- Preparing the content for adaptive streaming is an art

How to choose the target bitrates/resolutions to make switching as seamless as possible

How to determine the fragment durations

How to encode the content so that the perceived quality is stable and good even in the case of frequent up/downshifts



— Resolution — Color Gamut — Frame Rate — Interruptions

Source: Screen Digest

# Implications of Adaptive Streaming on the Networks

- **Client's behavior impacts its and others' streaming performance**

## Buffering State

Clients send back-to-back HTTP requests

Goal is to get to Steady State as soon as possible

## Steady State

Clients send periodic HTTP requests, thus do not use a TCP connection all the time

If the connection idles, e.g., because the buffer is full enough or the viewer hits Pause, TCP connection may enter Slow-start Restart (RFC 5681)

Goal is to keep the buffer level within min-max limits

**Attempting to strictly control variety of clients could be counter-productive**

- **Adaptive streaming causes large-volume new traffic**

It is all unicast (at least for now)

It is not well-enough studied to know exactly how to deal with it

Shared networks like HFC or DSL backhaul are likely to face bandwidth issues

Selfish clients melted an HFC network down during a college game

# Streaming with Multiple TCP Connections

## Similar to Driving on a Multi-Lane Highway



- Using multiple concurrent TCPs
  - Is not necessarily for greedily getting a larger share of the bandwidth
  - Helps mitigate head-of-line blocking
  - Allows fetching multiple (sub)-fragments in parallel
  - Allows to quickly abandon a non-working connection without having to slow-start a new one
- Studies showed that performance deteriorates if many clients adopt this approach and they do not limit their aggregated bandwidth consumption

# TCP Fairness ≠ Fair Streaming

- **Clients typically operate in unfettered greedy mode**

They consume all the bandwidth TCP gives them unless additional mechanisms exist

We do not want fair streaming necessarily anyway

The content/service provider could offer multiple tiers (e.g., premium vs. regular)

- **Can or should we provide controlled unfairness on the server side or in the network?**

Would better caching/replication/pre-positioning content avoid the overload?

Is there a better transport than TCP, maybe MPTCP, DCCP or SCTP?

Can or should the network somehow make the clients slow down differently than they do with normal TCP?

- **Should we consider IP multicast to help reduce bandwidth usage?**

Maybe when lots of viewers are watching in a narrow time window, or to spread very popular content to the edge to deal with flash crowds

## Further Reading

- **Special sessions on adaptive streaming in ACM MMSys 2011**

Technical Program and slides: at <http://www.mmsys.org/?q=node/43>

VoD of the sessions will be available in ACM Digital Library

- **2<sup>nd</sup> W3C Web and TV Workshop**

<http://www.w3.org/2010/11/web-and-tv/>

- **A two-part article from Cisco to appear in IEEE Internet Computing**

First part on streaming protocols (March/April 2011)

Second part on applications, standardization and open issues (May/June 2011)

**Thank you**

