SCTE CABLE-TEC EXPO '14
SEPTEMBER 22-25 / DENVER, CO

SCTE Society of Cable Telecommunications Engineers

Challenges with Large-Scale Adaptive Streaming Deployments

A Technical Paper prepared for the Society of Cable Telecommunications Engineers
by

Ali C. Begen, Ph.D.
Lead Engineer
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
abegen@cisco.com

Tim Donahue
Principal Engineer
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA 01719
tdonahue@cisco.com

Joshua Gahm
Principal Engineer
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA 01719
jgahm@cisco.com

Zhi Li, Ph.D.
Software Engineer
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
zhil2@cisco.com

Introduction

HTTP adaptive streaming (HAS) (also known as adaptive bitrate (ABR) streaming) has become the technology of choice for delivering video content over the Internet. As the cable industry moves toward IP video delivery of managed content, HAS will play an increasingly important role in delivering managed video to the primary and secondary screens of cable subscribers.

In HAS, streaming clients are offered multiple representations of the same content. Clients independently choose a segment belonging to a representation to fetch from the content delivery network, a choice that is made at every switching boundary (usually every 2-10 seconds). The choice is based on a number of parameters, including the network (TCP) throughput observed by the client.

A client in the HAS paradigm estimates TCP throughput during segment download, and treats such an estimate as a reliable indicator of sustainable network throughput. The estimate is then used to select a segment from several available encoding rates. When HAS traffic becomes a substantial fraction of the total network traffic, however, TCP throughput achieved during segment download becomes an unreliable indicator of sustainable network throughput.

We show that when multiple HAS clients compete for bandwidth at a network bottleneck, they develop optimistic estimates of fair-share bandwidth and choose higher segment encoding rates than may be reliably delivered through the network. Through analysis and experiments, we demonstrate that this leads to video bitrate oscillation and other undesirable behaviors that negatively impact the video viewing experience.

Compared to OTT content, this presents a problem for cable operators due to an expectation of high video quality among subscribers. As operators move to IP video, a larger fraction of DOCSIS bandwidth will be consumed by IP video traffic, exacerbating the problem.

In this paper we propose solutions, potentially applicable to DOCSIS cable networks, which may be used to address the problem. We also share experimental results from a test network designed to replicate a typical MSO access network.

Background

This section gives a quick overview of HAS technology. The section does not provide a complete introduction to HAS, but instead simply sketches the aspects of the technology that play an important role in the scaling issues described in this paper.

HAS is a technology for delivering live or on-demand video content over an IP network from content servers to viewers. The key idea of HAS that makes it different from earlier video delivery technologies is that the video content is first chopped up into small chunks (physically or virtually), with each chunk representing several seconds of the program. Next, each of these chunks of the program is encoded at several different

encoding bitrates and (usually) also at several different spatial resolutions. Each chunk of the program encoded at a particular bitrate and resolution is called a segment. Finally, the encoding is performed in such a way that an HAS player can readily switch from one encoding and/or resolution to another each time it displays the next chunk of program content. A player may do this without interrupting the smooth playback of the overall program.

As a simple example, a typical HAS representation of a program might divide the program into 2-second chunks, then encode each chunk at bitrates of 10000 Kbps, 6900 Kbps, 4650 Kbps, 3200 Kbps, 2100 Kbps, 1400 Kbps, 960 Kbps and 660 Kbps. The highest two encoding bitrates could encode the content at 1080p resolution, the next two could encode the content at 720p resolution, and the lower bitrates could encode the content at lower resolutions suitable for mobile devices.

The next big innovation of HAS technology is to treat each segment of an encoded program as ordinary Web content, addressable with a unique URL and retrievable by ordinary HTTP GET operations. In effect, HAS turns streaming video into a set of small files, which may be fetched from a Web server like any other Web content. These files may then be played out back-to-back on the consumer device to produce a continuous program.

Playback of an HAS program is controlled by an HAS client, which is generally a software component running on the receiver system. The HAS client makes all decisions about which chunks of the program to fetch from the Web servers and which encoding to select for each chunk. Clients typically fetch segments 10 to 30 seconds ahead of their current playback point so that they can maintain smooth playback, even if the next segment fetch takes longer than expected. The client stores segments awaiting playback in its segment buffer.

HAS clients select a resolution and encoding bitrate for each chunk of content they request from the Web servers based on the screen size of the display device, the processing power of the decoding and rendering hardware, and the available network bandwidth. To estimate the available network bandwidth, clients usually keep a history of the measured download speed of a few of the most recently fetched segments, then estimate the available network bandwidth based on a moving average of these recent download speeds. When the moving average of recent download speeds exceeds the encoding rate of the most recently downloaded segments, the client may perform a rate upshift, fetching subsequent segments of the program at a higher encoding rate. Similarly, if the moving average of recent downloads suggests that the available network bandwidth is below the encoding rate of recently requested segments, the client may downshift, requesting subsequent segments at a lower encoding rate. This process of performing upshifts and downshifts to match the encoding rate of the requested content to the currently available network bandwidth is called adaptation.

Since HAS technology, in effect, turns all video content into ordinary Web content, one might expect that a large HAS deployment would work well at scale, just as the Web works well at scale. However, even though HAS uses the same underlying technologies

as most Web applications, HAS differs from most other Web applications in one respect: HAS clients react automatically to the behavior of other HAS clients, at the application level. This occurs because the rate selection of each HAS client affects the bandwidth available to other HAS clients sharing a congested link. This means that each time an HAS client on a congested link upshifts to a higher encoding rate, the other clients sharing the same link will receive less bandwidth and may downshift, and vice versa.

Since the rate selection decisions of each HAS client affect the rate selection decisions of all competing clients, a population of competing HAS clients form an *accidental distributed control-feedback system*. Such systems often exhibit surprising and unintended behaviors. We will see in the next section that a variety of such behaviors do in fact occur with the HAS clients in wide use today.

Results from Test-Bed Experiments

This section provides results from lab experiments using several different types of widely deployed HAS clients.

To show a leading example, we present the results of a lab test in which 36 Microsoft Smooth Streaming clients compete with each other for bandwidth at a 100 Mbps bottleneck link. In this test, the bottleneck link is served in the direction from the Web server to the clients by a router with a single, shared queue and a RED drop policy. Each of the 36 clients runs on a separate virtual machine running Windows 7. Content is served from a Microsoft IIS server running on Windows Server 2008[1]. A delay is also inserted in the network path to give an overall round-trip time (RTT) between the clients and the server of 50 ms.

Figure 1 shows the average requested bitrate by all of the clients together as a function of time in this experiment. The x-axis represents time in seconds since the start of the experiment. The y-axis represents the average requested encoding rate of the segments requested by all of the clients in a 2-second bucket of request times.

[1] Separate benchmark testing is performed to ensure that the clients running on separate virtual machines on a single host system do not interfere with each other.

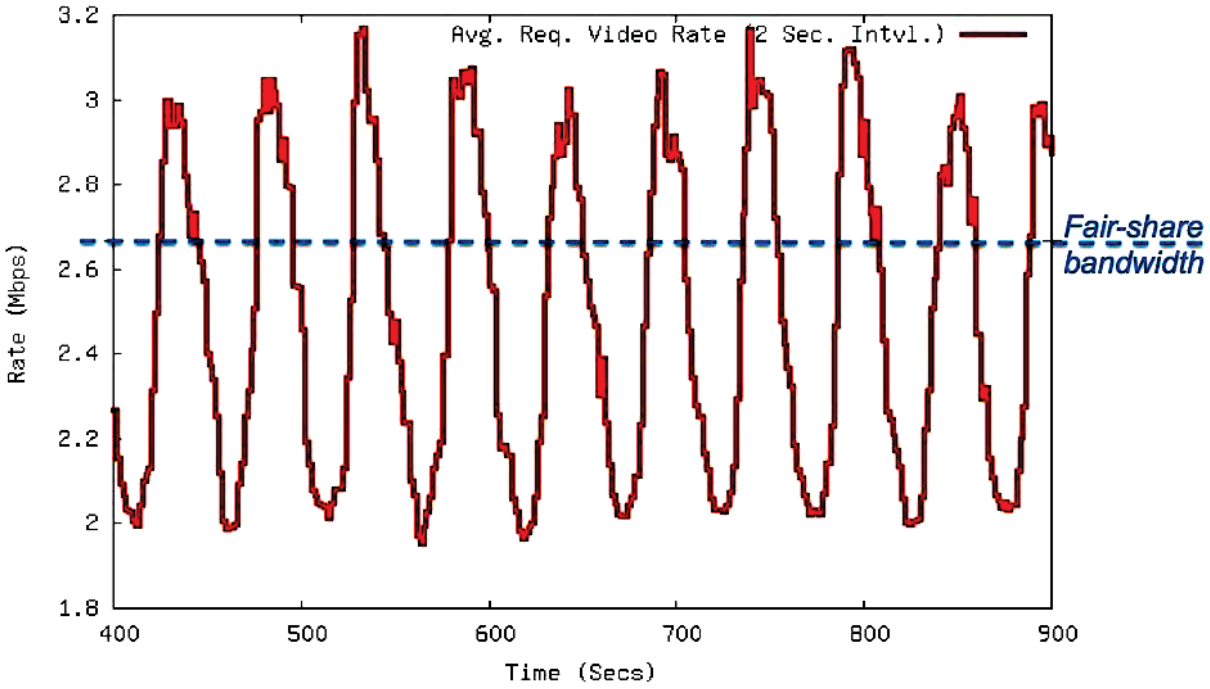

Figure 1: Average requested bitrate vs. time for 36 Microsoft Smooth Streaming clients on a 100 Mbps link.

As shown, the encoding rates requested by the HAS clients exhibit a very regular pattern of oscillation. The oscillation has an approximate 50-second period. Variants of this experiment with differing numbers of clients and network configurations suggest that the result is insensitive to the number of clients, the network RTT, the router's WRED configuration, the use of per-flow queuing (such as stochastic weighted fair queuing, or WFQ) rather than a single queue, the presence of moderate Web-like cross-traffic, the start times of the clients, and the link speed.

This excessive rate shifting is likely to have a negative effect on the subjective quality of experience (QoE) of consumers viewing the content. Figure 2, for example, shows the requested encoding rate as a function of time for four individual clients from the population of 36 clients in the experiment, with each of the four clients plotted in a different color. Each client spends most of its time shifting between two adjacent encoding rates (2.1 Mbps and 3.2 Mbps). However, each client also has periods when it upshifts to a higher rate for a short time and, more troubling, downshifts to a lower rate for a short time. Also, the clients in this experiment shift rates approximately every 19 seconds on average (roughly consistent with the idea that all the clients together upshift and downshift in unison about every 50 seconds).

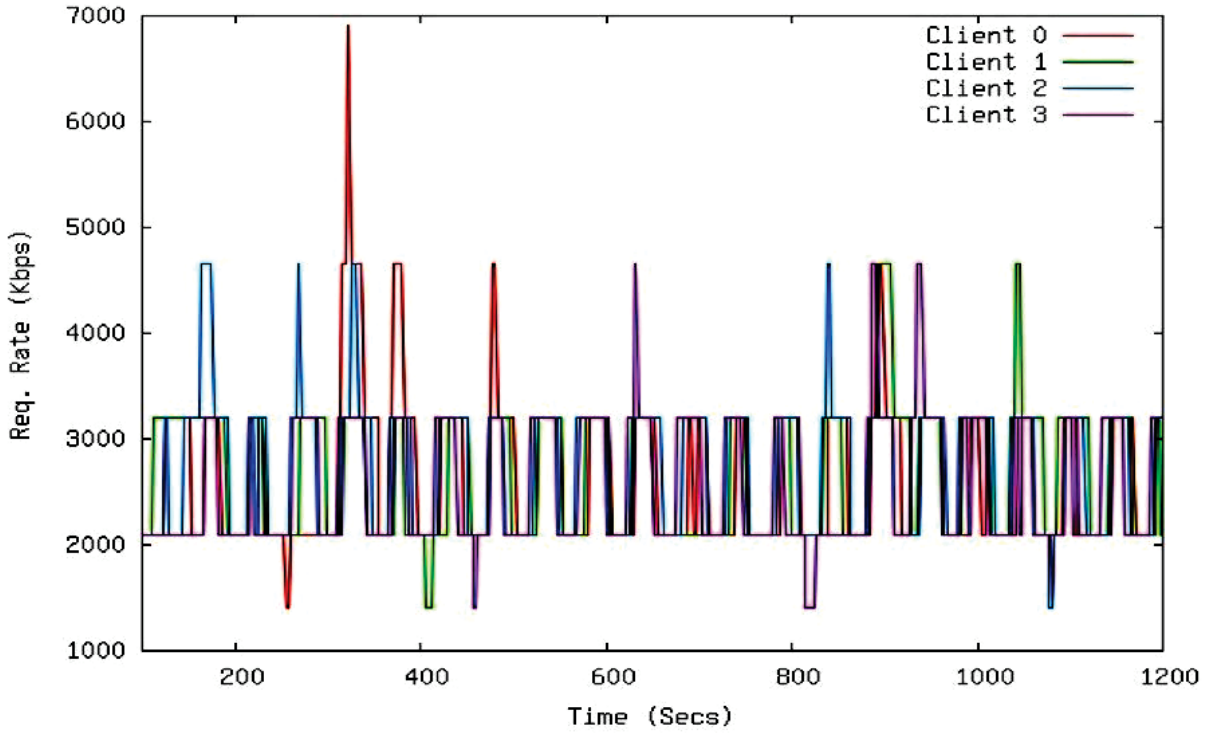

Figure 2: Requested bitrate vs. time for four selected clients from a population of 36 competing Microsoft Smooth Streaming clients.

Other HAS clients also make poor rate selections when multiple instances compete for bandwidth at a bottleneck link. Figure 3 illustrates the rate selection decisions by 22 Apple HLS clients competing for bandwidth on a 100 Mbps link. In this experiment, the clients were able to select from encoding rates 459, 693, 937, 1270, 1745, 2536, 3758, 5379, and 7861 Kbps[2]. The RTT was 20 ms. Several of the clients in this test stalled during playback. These stalls are reflected in the figure as traces that end early.

As may be seen in Figure 3, the clients in this test did not show a regular oscillation pattern, as was seen in the test with Microsoft Smooth Streaming clients. Instead, the Apple OS X Mavericks clients seem to begin with a series of rapid rate shifts near the start of the test, then gradually settle into a more stable set of rate selections, with most of the clients seemingly locked to a particular rate for the last 250-300 seconds of the test. Although we do not know the algorithm behind these rate selection decisions, it appears that some sort of heuristics or increasing dampening over time may be involved, with each client attempting to hunt for a sustainable rate early in the test and

[2] The rate selection trace for each client shown in Figure 3 has been offset from the actual encoding rate by a small but different amount for each client so that even when two clients select the same encoding at the same time their traces do not quite overlap in the graph and each trace can therefore be seen separately.

then eventually locking to a rate it finds sustainable.

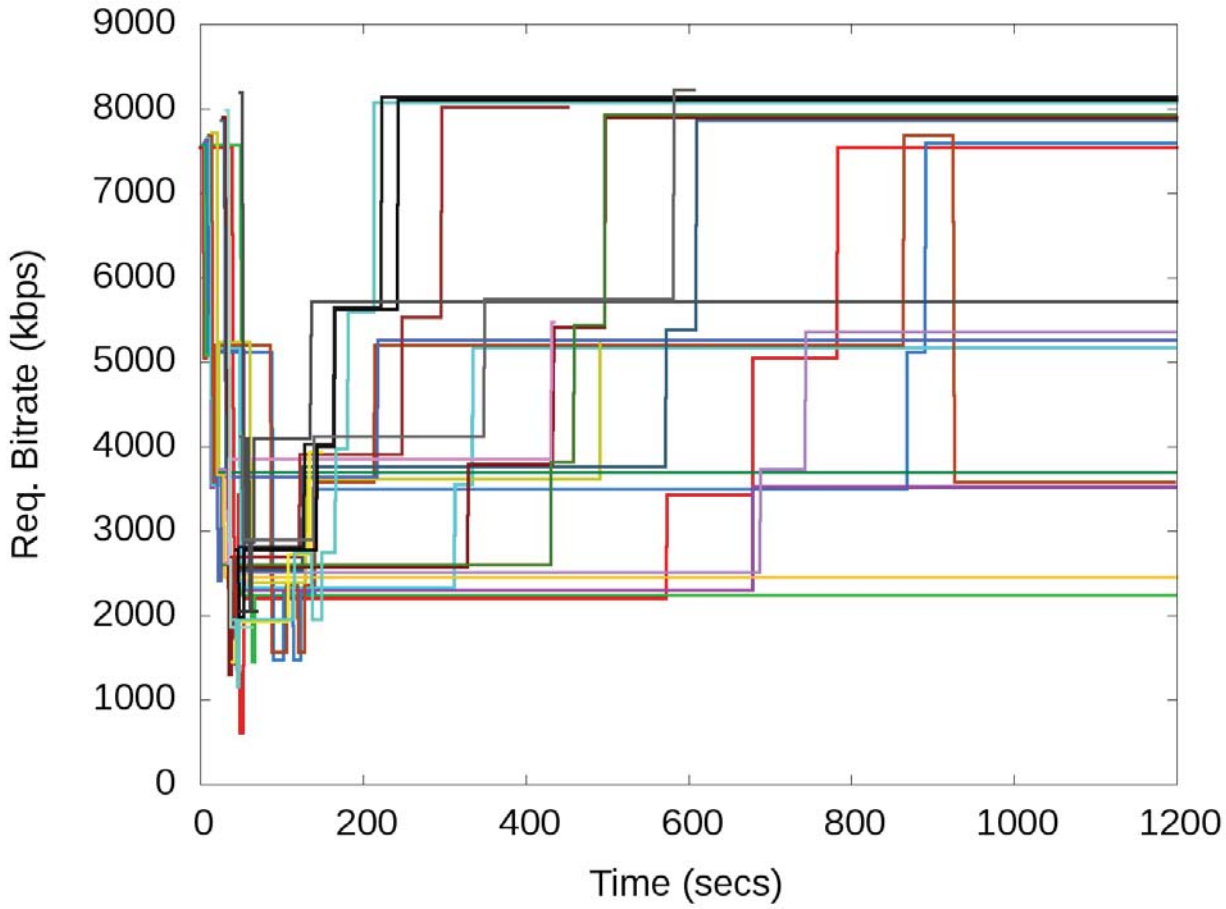

Figure 3: Rate selections of 22 OS X Mavericks clients sharing a 100 Mbps link.

Although the Apple OS X Mavericks clients exhibit fewer rate shifts overall than the Microsoft Smooth Streaming clients shown in the earlier experiment, the apparent hunting-and-locking strategy of the Apple clients results in very unfair outcomes by the end of the test. Several of the clients appear to have settled permanently or at least for a long period, on a relatively low encoding rate. Other clients have settled at the maximum available rate. A much better outcome would have been if all clients had settled at the midrange of the available rates, perhaps distributing themselves between the 3758 Kbps and 5379 Kbps rates, for example.

Although both the Apple HLS and Microsoft Smooth Streaming clients exhibit quite different behaviors in tests where multiple instances compete for bandwidth at a bottleneck link, both types of client have trouble settling on a stable, fair, optimal rate selections in these tests. Although space does not permit providing examples from other test scenarios, we have performed many similar tests using many different HAS

technologies, client types, and test conditions. In these tests, we have found that most, if not all, widely deployed HAS clients exhibit problems with stability, fairness and optimal rate selection in scenarios where clients must compete for bandwidth at a bottleneck link. Moreover, these results tend to be relatively unaffected by a variety of testing parameters, including the number of clients in the test, type of queuing in effect at the bottleneck link, relative start times of the players, presence of moderate Web-like background traffic, and particular choices of encoding rates available in the manifest.

Understanding the Root Cause

In the previous section we saw examples of how widely deployed HAS clients have trouble making stable and optimal rate selections when multiple HAS clients compete for bandwidth at a bottleneck link. In this section we provide some insight into the root cause of the difficulties that HAS clients have in estimating available bandwidth in a reliable manner. In summary, the root cause of these problems is that when HAS clients compete with each other for bandwidth, each client has a tendency to grossly overestimate the available network bandwidth. Overestimation of available bandwidth is particularly acute when the bottleneck link is slightly underutilized for a brief period.

Earlier published work has shown how HAS clients can overestimate the available bandwidth when they use segment download rates to estimate available bandwidth. Akshabi et al., first described this phenomenon in [1], showing how the bandwidth overestimation could result from the on/off pattern of transfers requested by an HAS client in its steady-state mode of operation. Li et al. also described the phenomenon in [2] and provided a theoretical analysis in Appendix A of [3].

In this paper, we therefore do not provide a complete explanation of the phenomenon, but instead show that the phenomenon exists in the types of experiments we have shown above. We then provide an explanation for the behavior we observed in those experiments.

In Figure 4, we show the measured download rate, on a segment-by-segment basis, for three Apple HLS (OS X Lion) clients. In this experiment 30 clients were competing for bandwidth on a 100 Mbps bottleneck link. Allowing for network overhead, the fair-share bandwidth available to each client in this experiment is approximately 3.2 Mbps. This fair-share bandwidth is shown by the black line in Figure 4. Note how the actual measured download rates seen by the three representative clients are not only very noisy, but also strongly upward biased when compared with the fair-share bandwidth. Many of the measurements are too high by a factor of two; a significant number of the measurements are too high by a factor of four.

Since HAS clients typically use the measured segment download rate as an estimate of the available fair-share bandwidth, it is not surprising that the clients in this experiment have trouble settling on a sustainable encoding rate selection. Note also that because the measurements are almost all high, averaging these measurements over a number of downloads will still not yield a reliable estimate of the available bandwidth.

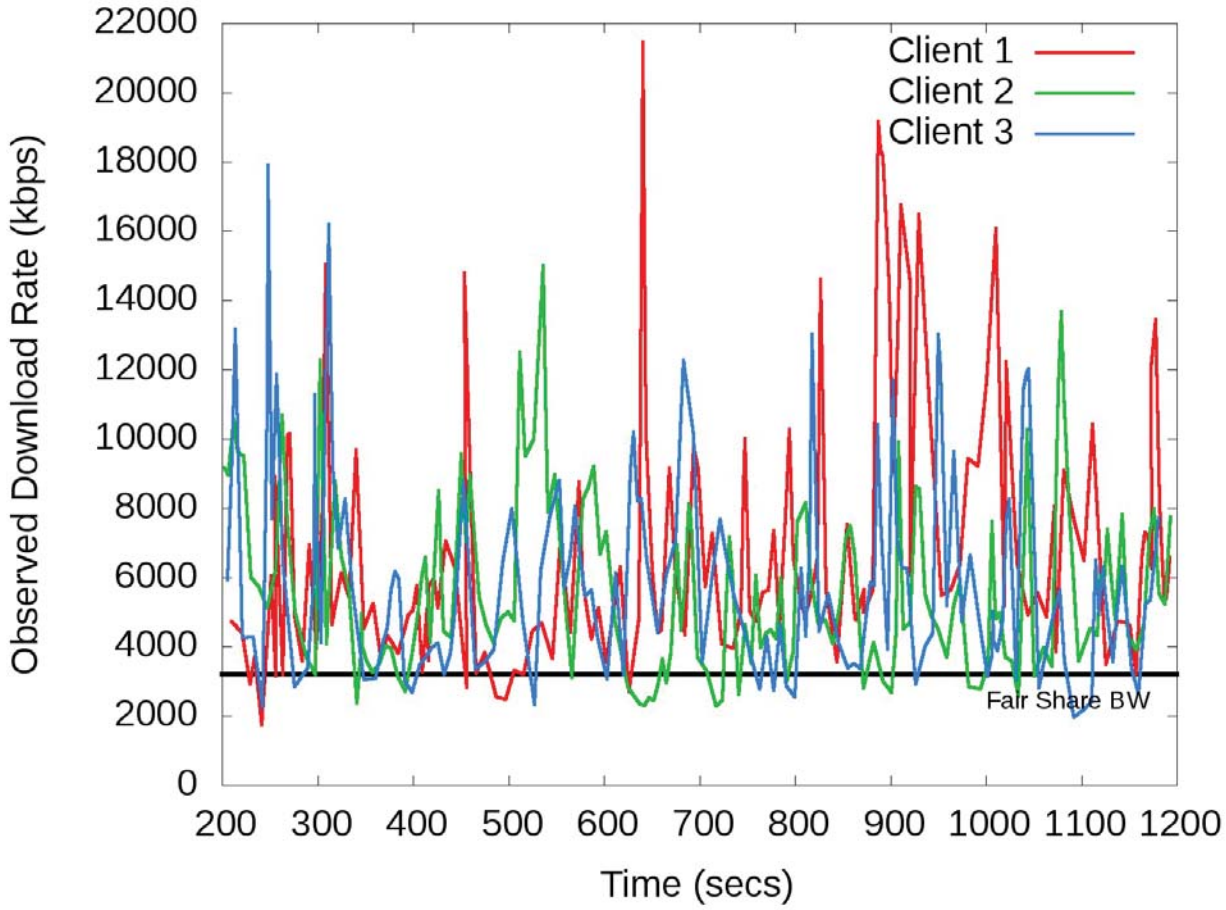

Figure 4: Observed download rates of three clients from a population of 30 OS X Lion HLS clients.

Looking at Figure 5, we can also see how bandwidth overestimation drives the oscillation of the system in the experiment with 36 Microsoft Smooth Streaming clients competing for 100 Mbps of bandwidth. The red line in Figure 5 corresponds to the plot in Figure 1, again showing the average requested bitrate as a function of time in the experiment with 36 competing Microsoft Smooth Streaming clients. In Figure 5, however, separate plots show how the oscillation in requested encoding rates corresponds to oscillations in the average measured download rate of segments (green plot) and in the overall link utilization (blue plot, read against the right-hand axis).

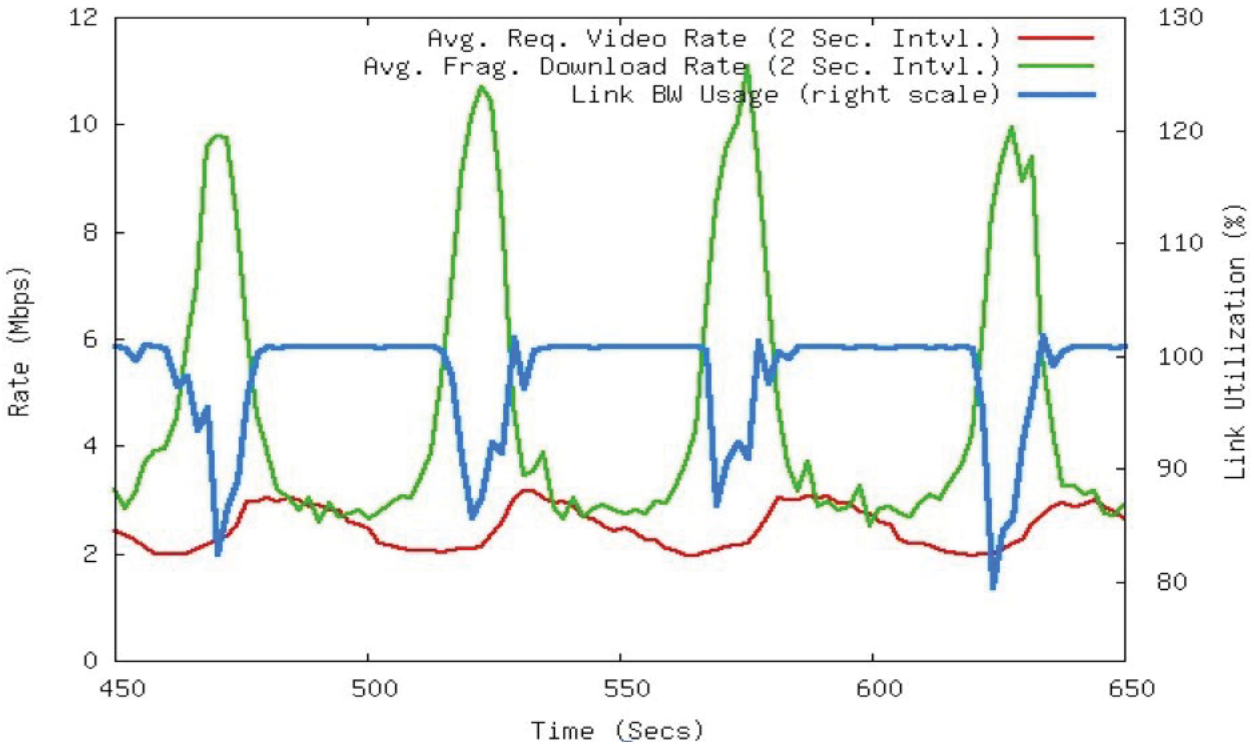

Figure 5: Requested bitrate, download rate and link utilization for 36 Microsoft Smooth Streaming clients on a 100 Mbps link.

Referring first to the green trace, the peaks in average requested encoding rate are preceded each time by a sharp peak in the measured download rate of segments. Since the HAS clients decide when to upshift and downshift based on a moving average of recent segment download rates, it is not surprising that if the clients suddenly complete some downloads at an unexpectedly high rate (e.g., 10 Mbps in this experiment) many of the clients decide to upshift shortly thereafter. The question then is why the clients would regularly manage to complete segment downloads as fast as 10 Mbps when, in this example, the actual fair-share bandwidth on the link is around 2.6 Mbps.

Since most, if not all, existing HAS clients use measured segment download rate to estimate available bandwidth on a link, it is worth inquiring further about the circumstances in which overestimation of available bandwidth can occur and how severe the problem can be. Although space does not permit a full discussion of this phenomenon (see [2] and [3] for more details), a few points are worth noting.

First, overestimation of available bandwidth seems to occur mostly, if not entirely, when the bottleneck link is undersubscribed to some extent. The relationship between link subscription and segment download rate can be seen from the results of an experiment using real TCP connections and a test program designed to behave like an HAS client in steady-state mode. In more detail, the test program in this experiment is constructed so that it requests a fixed size segment every two seconds, with the size of the segment configurable as a parameter of the test. If the requested segment download completes

within two seconds, the test program waits until the start of the next two-second interval before requesting another download. If the requested segment download completes in more than two seconds, the test program requests the next download immediately. This behavior is very similar to the behavior of an HAS client in steady state-mode, but without any rate shifting.

Using this test program, it is possible to emulate the behavior of different size populations of clients (effectively a separate copy of the test program for each client) and, by varying the segment size, different levels of link subscription. In this context, link subscription means the total size of the data that would be requested by all of the clients together in a two-second interval if infinite bandwidth were available on the link, divided by the actual size of the data that can be downloaded in a two-second interval with the actual link bandwidth. For example, consider a 100 Mbps link with 100 instances of the test program all set to try to download 2.02 Mbits every 2 seconds. The link subscription rate in this case is 101%.

Figure 6 shows the results of using this test program to determine how measured download rates change as the link subscription rate varies from 90% to 110%, using real TCP connections. This test uses 100 client emulator instances on a 100 Mbps link with a two-second fetch interval and a 50 ms RTT. As can be seen from the plot, the measured download rate is higher than the actual fair-share bandwidth by about a factor of three until the link subscription rate reaches full utilization[3]. When the link reaches full subscription, the measured download rate of the segments abruptly shifts to a fairly accurate estimate of the actual fair-share bandwidth (approximately 1 Mbps). We call this abrupt shift from gross overestimation of available bandwidth in the client population to fairly accurate estimation the “bandwidth estimation cliff”.

[3] Full utilization in this experiment occurs around 96% on the x-axis of the graph because the subscription rate shown on the x-axis does not include correction for L2 and L3 overhead.

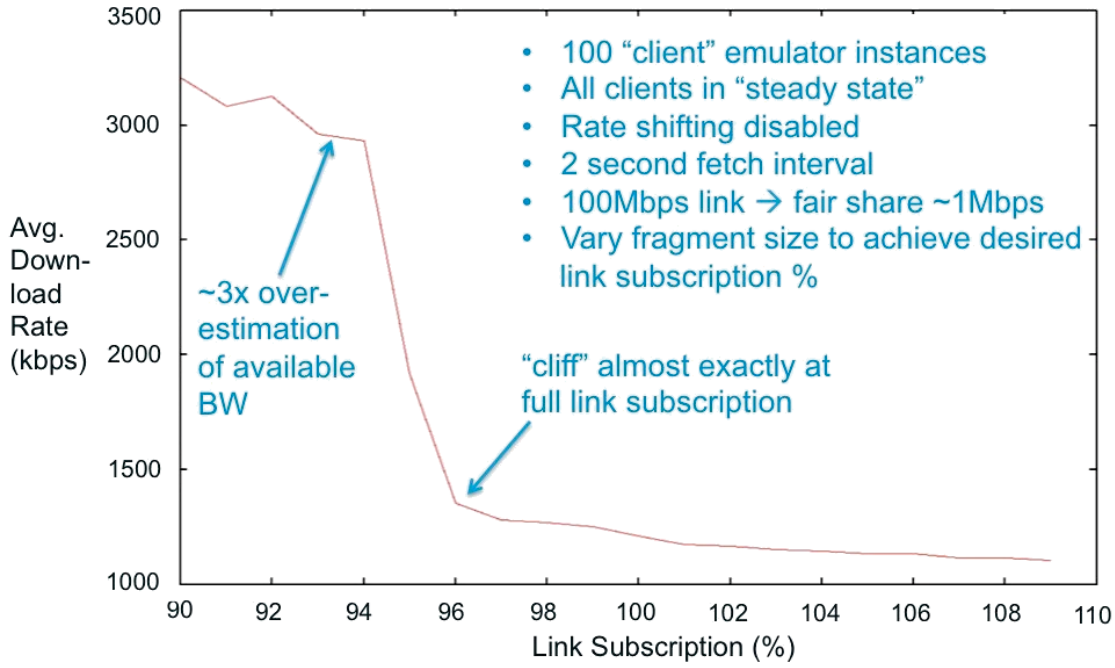

Figure 6: Bandwidth overestimation vs. link subscription produced by the client emulator test program.

With an understanding of the bandwidth cliff effect, it is not difficult to understand how HAS clients have difficulty in estimating available bandwidth when they compete with other HAS clients and use average download rate as an estimate of the available bandwidth. Specifically, since HAS rates are quantized, it is quite unlikely that a population of HAS clients will select a set of encoding rates in such a way as to exactly utilize the available bandwidth on a shared link. In the event that the clients select rates that slightly oversubscribe the link, then the rates cannot be sustained for very long before some clients exhaust their segment buffers and must downshift. In the event that the clients select rates that slightly undersubscribe the link, each client will measure download rates much higher than the actual available bandwidth and will therefore upshift to a higher rate. In effect, HAS clients cannot get an accurate estimate of their available bandwidth until they are already using too much bandwidth.

A Client-Based Solution

To overcome the bandwidth overestimation problem, we have proposed a client-based solution based on a "probe and adapt" principle, which we name PANDA (Probe AND Adapt). PANDA does not require more measurements than a conventional scheme – all it needs are the TCP throughput as calculated and the client buffer size. It is fairly straightforward to implement PANDA based upon modifying an existing scheme.

In PANDA, the TCP downloading throughput is taken as an input only when it is an accurate indicator of the fair-share bandwidth. This usually happens when the network

is oversubscribed (or congested) and the off-intervals are absent. In the presence of off-intervals, the algorithm constantly *probes* the network bandwidth by slightly incrementing its sending rate, and prepares to back off once it experiences congestion. This new mechanism shares the same spirit with TCP's congestion control, but it operates independently at the application layer and at a per- segment rather than a per-RTT time scale.

Probing constitutes fine-tuning the requested network data rate, with continuous variation over a range. By nature, the available video bitrates in HAS can only be discrete. A main challenge in our design is to create a continuous decision space out of the discrete video bitrates. As a solution, we fine-tune the intervals between consecutive segment download requests such that the average data rate sent over the network is a continuous variable. Consequently, instead of directly tuning the video bitrate, we probe the bandwidth based on the average data rate, which in turn determines the selected video bitrate and the fine-granularity inter-request time.

There are various benefits associated with PANDA's probe-and-adapt approach. First, it avoids the pitfall of inaccurate bandwidth estimation. Having a robust bandwidth measurement to begin with gives the subsequent operations improved discriminative power (for example, strong smoothing of the bandwidth measurement is no longer required, leading to better responsiveness). Second, with constant probing via incrementing the rate, the network bandwidth can be more efficiently utilized. Third, it ensures that the bandwidth sharing converges towards fair share (i.e., the same or adjacent video bitrate) among competing clients. Lastly, an innate feature of the probe-and-adapt approach is asymmetry of rate shifting – PANDA is equipped with conservative bitrate level upshift but more responsive downshift. Responsive downshift facilitates fast recovery from sudden bandwidth drops, and thus can effectively mitigate the danger of playout stalls caused by buffer underrun.

The PANDA algorithm has been fully described in [2], and space does not permit a full description here. Figure 7, however, does show the results of an experiment with 36 PANDA clients competing for 100 Mbps of bandwidth (top graph) and also a graph of the rate selections of 36 Microsoft Smooth Streaming clients under similar conditions (bottom graph). As we can see, the PANDA clients exhibit many fewer rate shifts than the Microsoft Smooth Streaming clients, and also confine their rate selections to a much narrower range.

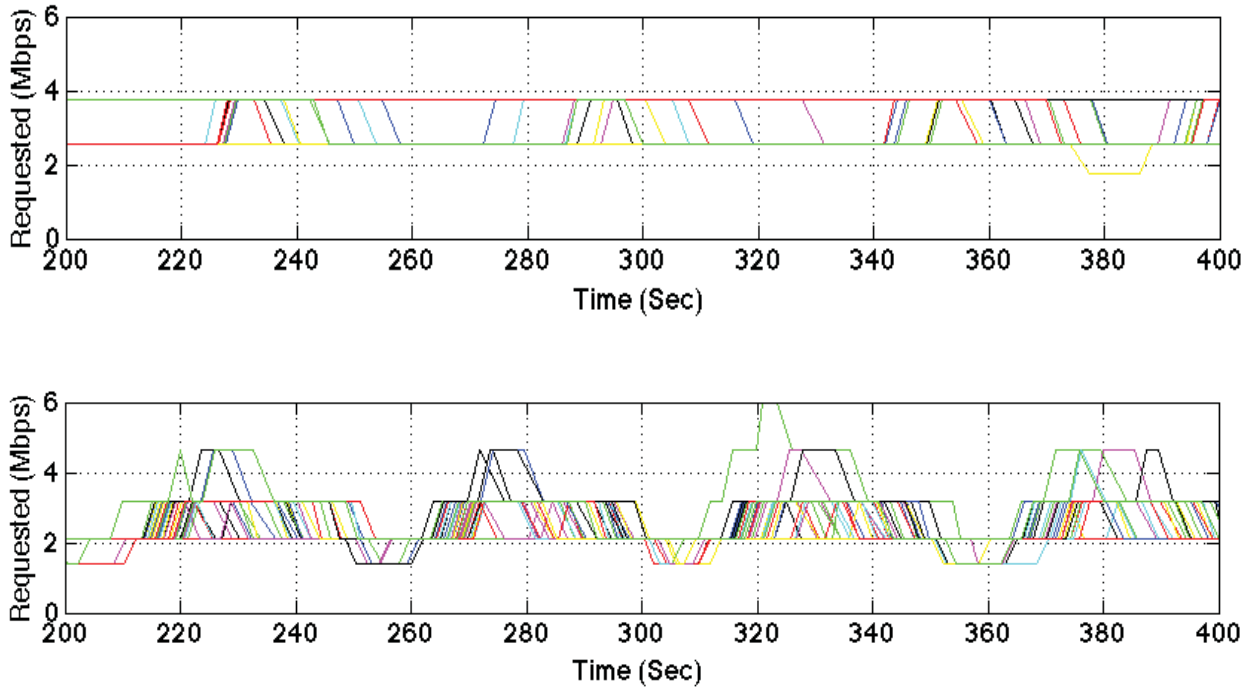

Figure 7: Bitrate selections of 36 PANDA clients (top) compared with bitrate selections of 36 Microsoft Smooth Streaming clients (bottom) on a 100 Mbps link.

Network-Based Solutions

It may be possible to deliver improved video playback QoE by implementing improved encoding rate selection algorithms in HAS clients, then deploying updated player software to consumers. However, there are many different HAS clients in use, deployed across many millions of devices. A more practical strategy may be to attempt to improve HAS client rate selection and playback performance through application of network-based mechanisms. If successful, we may eliminate the need for customers and managed service operators to update large numbers of HAS clients.

We have considered two basic types of network-based improvements to HAS adaptation. The first approach would apply network QoS mechanisms to HAS packet flows, in ways which would cause the existing rate selection algorithms in deployed HAS clients to initially choose an optimal encoding rate and reduce the frequency and magnitude of rate shifts during content playback. The second approach would make use of, in a DOCSIS transport network, DOCSIS signaling between HAS endpoints, transport network elements and, potentially, controllers running on third-party servers to optimize rate selection decisions made by clients.

Modern L2/L3 network transport equipment typically includes the ability to:

- Classify packets into traffic classes by matching values in packet header fields, including source and destination address, source and destination port number,

transport protocol, Type of Service bits, Differentiated Services Code Point (DSCP) value, and packet length;
- Guarantee a minimum available transport network bandwidth to the packets in a traffic class;
- Shape the packets in a traffic class to a specific rate;
- Limit the packet transmission rate to a maximum value, while providing some tolerance for bursts above that maximum value;
- Limit the packet rate to a peak value, dropping packets which would otherwise cause the traffic rate (averaged over some short interval) to exceed the peak rate;
- Prioritize transmission of packets in one class above or below transmission of packets in other traffic classes;
- Change the value of certain packet header fields depending on whether the packets conform to or exceed specified packet rates, so that downstream equipment may classify and process such packets in different ways.

It seems possible that application of one or more of the available network QoS mechanisms could be used to influence and improve HAS rate selection decisions. We are investigating several approaches here as part of our work.

Control Plane Approaches

Control of video streaming systems is mostly motivated by the desire to maintain consistently good viewer experience, taking into account constraints in the delivery and the execution environment. The comprehensive definition of consistent good viewer experience is a complex aspect for itself, but it is well known that at least the following aspects influence a viewer's experience: visual quality (frame rate, spatial resolution, distortion, etc.), audio quality (mono vs. stereo vs. multi-channel audio), data loss, rebuffering and stall durations and frequency, startup latency, reactiveness to viewer actions, and for some cases, the glass-to-glass delay.

Earlier on-demand video delivery over the Internet was based on stateful protocols, e.g., the Real-time Streaming Protocol (RTSP) that used Real-time Transport Protocol (RTP) and UDP. RTSP-based systems enabled a significant amount of server/network control as the server made decisions on the delivered data based on measurements and feedback from the client. If reliable delivery could be guaranteed, the service provider had a control over the audiovisual quality presented for each client. However, such systems have proven to be expensive and not highly scalable in terms of delivery and server infrastructure. In HAS, the clients perform the adaptation task without direct feedback or instructions from the server, network or the service provider. Yet, one of the essential questions is the following: Is this autonomous behavior sufficient for HAS clients or could HAS clients use external hints to make more appropriate choices? The external hints could be any information to be provided by a streaming server, network elements such as routers and CMTS, or the service provider. These hints could aim to achieve any of the following goals: (i) higher server and network resource utilization, (ii)

better client performance, and (iii) better fairness among the HAS clients.

More relevant questions are:

- What is the scope of being controlled for the HAS clients?
- Is the control happening in the client or is the control on lower layers?
- Which are the entities that control the HAS client?
- Is this an enforced control or is it only an operational assistance?

These questions are part of one of the core experiments that is currently being run at the DASH working group at MPEG. The core experiment is called server and network assisted DASH (SAND). In SAND, there are messages sent by the clients to the media origin, DASH-aware network elements (DANE) and third-party (analytics) servers. These messages are called metrics. There are also messages sent by the third-party servers and DANEs back to the clients that are called SAND parameters. Finally, there are messages sent by the media origin to the DANEs that are parameters for enhancing delivery by the DANEs; these are called PED parameters. Figure 8 shows these message flows.

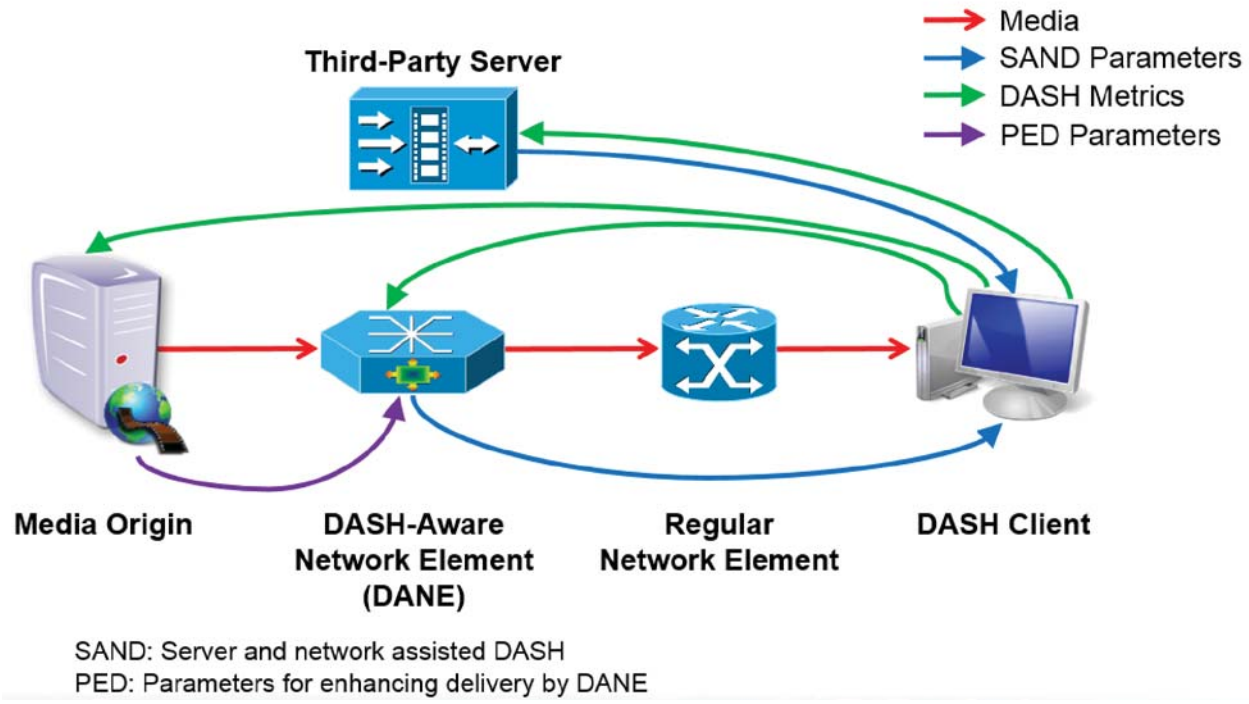

Figure 8: The message flows that are being considered as part of the SAND core experiment at MPEG.

As the core experiment is still in progress[4], the standardization of the SAND architecture

[4] The details regarding the current status of the core experiment can be found in [4].

is subject to discussion. Many vendors involved in the discussion agree that the SAND architecture will address several needs of the service providers and mobile operators, who currently do not have a standard way of managing the streaming clients deployed in their networks. An immediate use of the SAND architecture would be to provide assistance to the HAS clients so that they can avoid bitrate oscillations. Overall, we expect a rapid progress in this core experiment.

Conclusions

This paper has shown that when most of the traffic on a network is ABR, behaviors begin to emerge that are different than one would expect if ABR traffic were just like ordinary Web traffic. Behaviors that have been observed through experiments on test networks include, unnecessary client rate-shifting, synchronized patterns of upshifting and downshifting among competing clients, and overestimation of available bandwidth combined with link underutilization.

As ABR deployment advances, ABR will increasingly become the dominant traffic load on many networks most of the time. At periods of peak load, this ABR traffic will drive shared links into congestion. When this happens, it is expected that some of the phenomena described in this paper will begin to noticeably affect user experience and will need to be addressed. In DOCSIS deployments, we expect that when ABR is deployed for delivery of primary, on-network video content, we will begin to see bonding groups driven into congestion at peak prime-time viewing hours, possibly triggering the types of pathologies we have documented during the most popular viewing periods.

While current ABR technologies have been extremely successful in the over-the-top applications for which they were originally designed, the technology will need to evolve as ABR moves to full scale and onto managed networks. This paper has outlined possible client-based and network based approaches that may be used to address these problems as they begin to emerge.

Bibliography

[1] Saamer Akhshabi, Lakshmi Anantakrishnan, Constantine Dovrolis and Ali C. Begen, "What happens when HTTP adaptive streaming players compete for bandwidth?," in *Proc. ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Toronto, Canada, June 2012.

[2] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen and David Oran, "Probe and adapt: rate adaptation for HTTP video streaming at scale," *IEEE J. Select. Areas Commun., Special Issue on Adaptive Media Streaming*, vol. 32/4, pp. 719-733, Apr. 2014.

[3] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen and David Oran, "Probe and adapt: rate adaptation for HTTP video streaming at scale", *arXiv:1305.0510v2, http://arxiv.org/abs/1305.0510v2.*

[4] "Descriptions of core experiments on DASH amendment," (w14347), MPEG 108, Valencia, Spain, Mar. 2014

Abbreviations & Acronyms

ABR	Adaptive bitrate streaming
DASH	Dynamic adaptive streaming over HTTP
HAS	HTTP adaptive streaming
HLS	HTTP live streaming
PANDA	Probe-and-adapt
QoE	Quality of experience
QoS	Quality of service
RED	Random early drop
RTT	Round-trip time
SAND	Server and network assisted DASH