# OPTIMIZING FAIRNESS OF HTTP ADAPTIVE STREAMING IN CABLE NETWORKS

Michael Adams
Chris Phillips
Solution Area Media
Ericsson

*Abstract*

*This paper describes a novel approach to traffic management for HTTP adaptive streaming that optimizes fairness across multiple clients and increases network throughput. Readers of the paper will gain an understanding of the network impacts of implementing HTTP adaptive streaming, and how network management techniques may be applied to optimize fair bandwidth allocation between competing streams.*

*Benefits for the network operator include:*

- *enforcing fairness in the network (without resorting to techniques such as deep packet inspection),*
- *managing and ensuring consumers' overall quality of experience, and*
- *preventing network instability that can be caused by competing clients in a shared access network.*

*Benefits for the consumer include:*

- *a more consistent overall quality of viewing experience, and*
- *the ability to simultaneously use multiple devices within the home.*

*The concepts described in this paper have been prototyped to show improvements in fairness and overall network throughput without placing special constraints on the client implementation (which is typically outside of the operator's control). The results are being published here for the first time.*

## BACKGROUND

There is a great deal of interest in HTTP adaptive streaming because it can greatly improve the user experience for video delivery over unmanaged networks. Adaptive streaming operates by dynamically adjusting the play-out rate to stay within the actual network throughput to a given endpoint, without the need for "rebuffering". So, if the network throughput suddenly drops, the picture may degrade but the end user still sees a picture.

Although adaptive streaming was originally developed for "over-the-top" video applications over unmanaged networks, it also brings significant advantages to managed networks applications. For example, during periods of network congestion, operators can set session management polices to permit a predefined level of network over-subscription rather than blocking all new sessions. This flexibility will become more and more important as subscribers start to demand higher quality feeds (1080p and 4K).

HTTP adaptive streaming is the generic term for various implementations:

- Apple HTTP Live Streaming (HLS) [1]
- Microsoft IIS Smooth Streaming [2]
- Adobe HTTP Dynamic Streaming (HDS) [3]

Although each of the above is different, they have a set of common properties (see Figure 1):

- Source content is transcoded in parallel at multiple bit rates (multi-rate transcoding). Each bit rate is called a profile or representation.
- Encoded content is divided into fixed duration segments (or chunks), which are typically between two and 10 seconds in duration. (Shorter segments reduce coding efficiency while larger segments impact speed to adapt to changes in network throughput).
- A manifest file is created, and updated as necessary, to describe the encoding rates and URL pointers to segments.
- The client uses HTTP to fetch segments from the network, buffer them and then decode and play them.
- The client algorithm is designed to select the optimum profile so as to maximize quality without risking buffer underflow and stalling (rebuffering) of the play-out. Each time the client fetches a segment, it chooses the profile based on the measured time to download the previous segment.

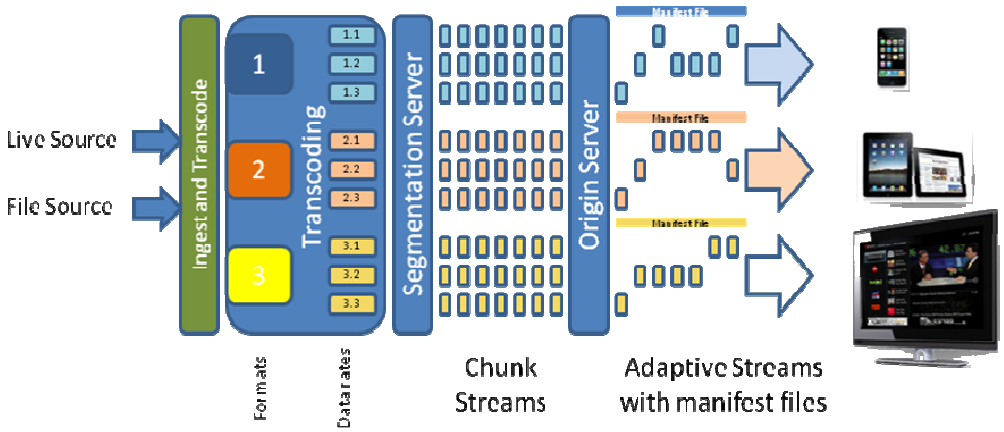



Figure 1: Ingest, transcoding, segmentation and adaptive streaming.

MPEG DASH

MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is certain to become a significant force in the marketplace [4]. While HLS uses the MPEG-2 transport stream format (which is widely deployed in most conventional digital TV services), Smooth Streaming and MPEG-DASH use an MPEG-4 Part 14 (ISO/IEC 14496-12) transport format known as fMP4 or ISO MP4FF.

Smooth Streaming and MPEG-DASH include full support for subtitling and trick modes, whereas HLS is limited in this regard. MPEG-DASH enables common encryption, which simplifies the secure delivery of content from multiple rights holders and to multiple devices.

Another key difference is the way in which MPEG-DASH and Smooth Streaming play-out is controlled when transmission path conditions change. HLS uses manifest files that are effectively a playlist identifying the different segments so that, for instance, when path impairment occurs, the selection of the URL from the manifest file adapts so that lower bit-rate segments are selected. In Smooth Streaming the client uses time stamps to identify the segments needed and thus certain efficiencies are gained. Both HLS and Smooth Streaming handle files in

subtly different ways, each claiming some efficiency advantage over the other. Both use HTTP, which has the ability to traverse firewalls and network address translation, giving it a clear advantage over RTSP, RTMP and MMS.

Adaptive Streaming Standardization

There are a number of initiatives aimed at large parts of the overall solution for streaming video. A document called *MPEG Modern Transport over Networks* was approved at the 83rd MPEG meeting in January 2008, which proposed a client that was media aware with optimization between the transport and content layers to enable video to traverse networks in an adaptive manner. However, at that time, its focus was on the widespread adoption of a variant of AVC/MVC called SVC (Scalable Video Coding) that would allow the client to generate acceptable video from a subset of the total aggregated transport stream.

Subsequently, at the 93rd meeting, the focus was changed to HTTP streaming of MPEG Media called *Dynamic Adaptive Streaming over HTTP* (DASH) using 3GPP's Adaptation HTTP Streaming (AHS) as the starting point. MPEG has standardized a Manifest File (MF), a Delivery Format (DF), and means for easy conversion from/to existing File Formats (FF) and Transport Streams (TS) as illustrated in Figure 2.

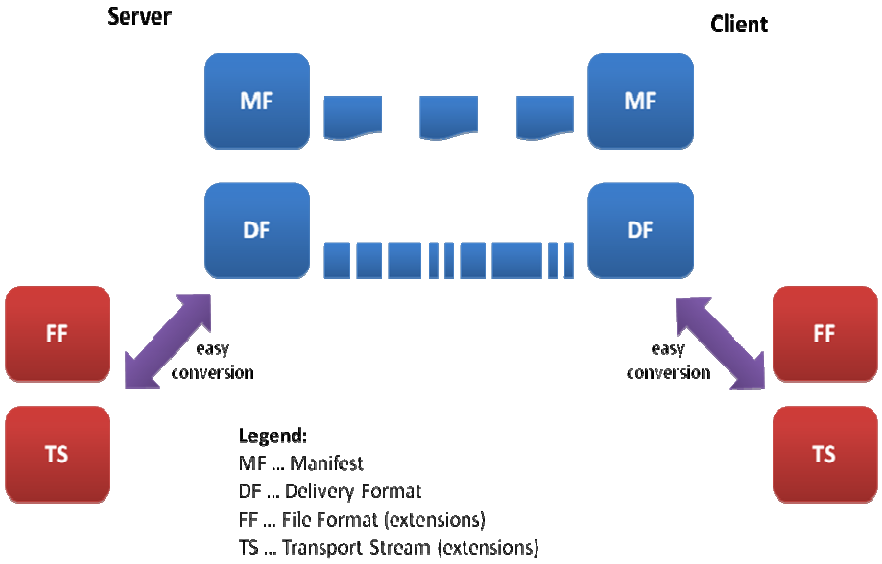



Figure 2: Dynamic Adaptive Streaming over HTTP (DASH).
*Courtesy: Christian Timmerer, Assistant Professor at Klagenfurt University (UNIKLU)*

The MPEG-DASH standard got Final Draft International Standard status in December 2011. MPEG-DASH has the potential to simplify and converge the delivery of IP video, provide a rich and enjoyable user experience, help drive down costs and ultimately enable a better content catalog to be offered to consumers, because more revenues can be re-invested in content, rather than paying for operating overheads. It will help streamline and simplify workflows and enable operators and content providers to build sustainable business models to continue to deliver the services that consumers demand.

FAIRNESS IN ADAPTIVE STREAMING

HTTP adaptive streaming clients implement a “greedy” algorithm, in which

they will always seek to achieve the maximum bit rate available. This can lead to instability, oscillation and unfairness, where some clients will win and others will lose in times of network congestion [5], [6].

## Reference Architecture

Figure 3 illustrates a typical arrangement where HTTP adaptive streaming is used to deliver video and audio programming to a device in a subscriber's home. Note that a CDN is typically used to replicate segments within the core network and this is assumed to have infinite bandwidth. At the edge of the network, the bandwidth is constrained by:

1. The downstream path over DOCSIS.
2. The wireless network path to a Wi-Fi connected device.

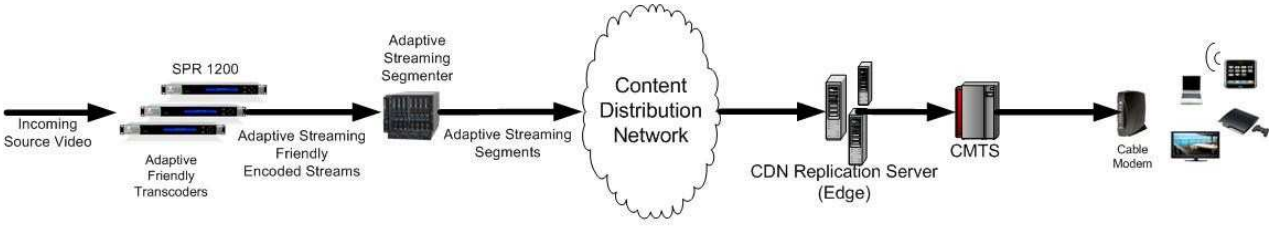



Figure 3: Reference architecture

## Prototype System Description

Figure 4 shows the prototype system that was developed to analyze the behavior of standard HLS adaptive streaming clients on a shared access network.

Packet scheduling is determined by the bandwidth monitor/allocator. It also creates a virtual pipe and constrains all packet delivery within it. The virtual pipe can be dynamically resized while the system is running. Two scheduling algorithms can been implemented within the virtual pipe; best-effort and weighted fair queuing. Best-effort implements first-come, first-served packet delivery. The weighted fair queuing algorithm schedules transmission according to the virtual pipe size and compares the amount of data transmitted through various classes to ensure that each class is allocated its fair share.

The bandwidth monitor/allocator also logs bandwidth utilization data for use by the real-time statistics monitor and the graphing module. This data is used to visualize the behavior of the system and to understand the behavior of the adaptive streaming clients.

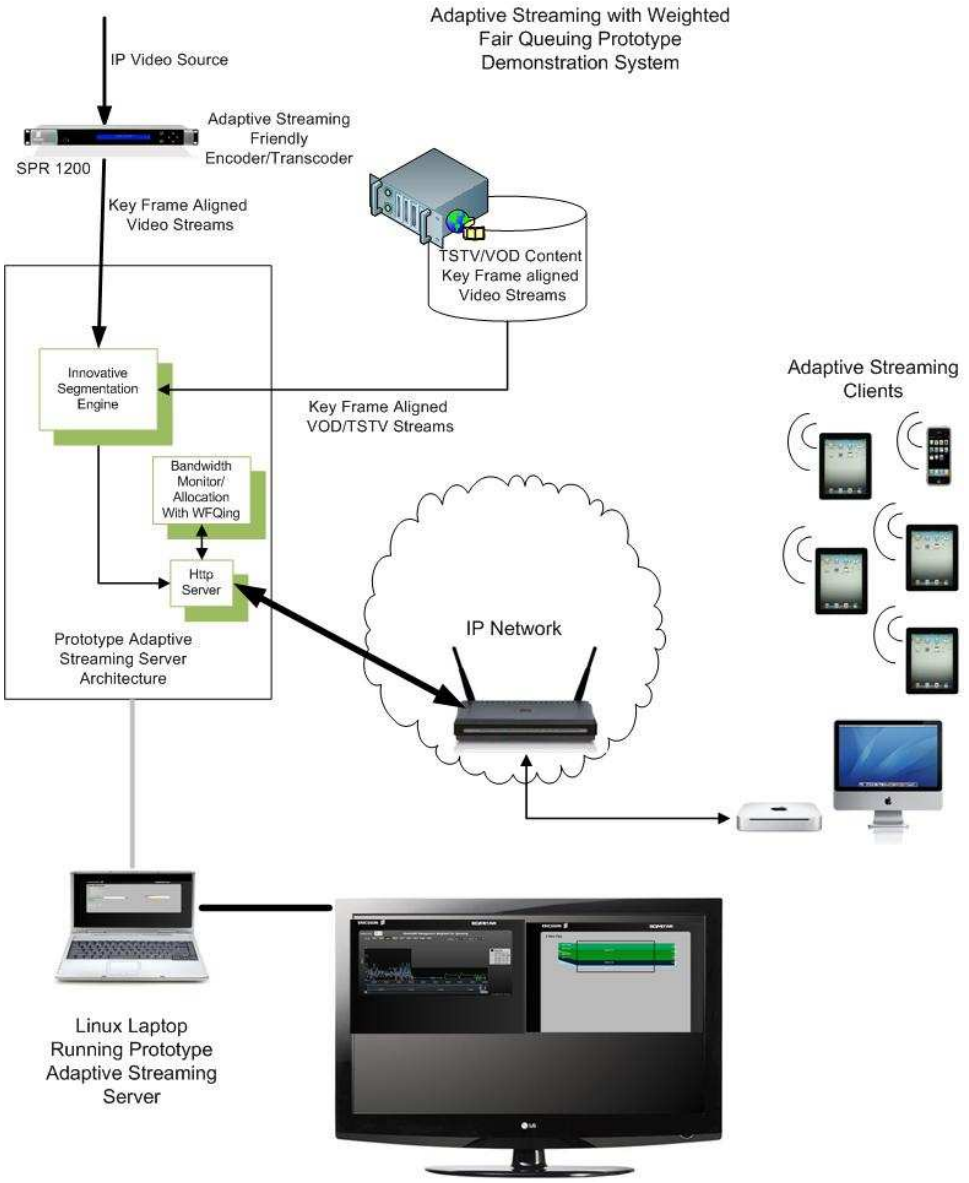



Figure 4: Prototype system

## EXPERIMENTAL RESULTS

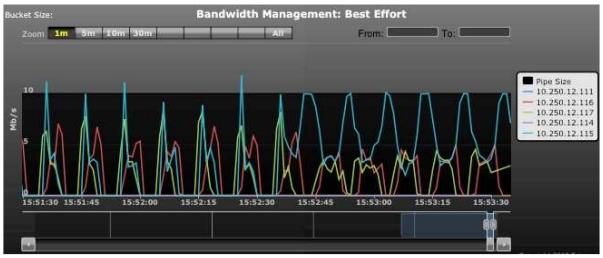
The experimental approach followed was to compare results of the best-effort (no traffic management) behavior with that of weighted fair queuing. The first best-effort run was repeated with identical parameters except for enabling the weighted fair queuing algorithm in the second and third runs. A fourth and fifth run were done with a different set of encoding profiles to investigate the effect that this would have on the behavior of the adaptive streaming clients.

### Run 1: Best Effort

| | |
|---|---|
| Start Time | 15:30:00 GMT |
| Pipe | 10 Mbps |
| Content | How to Train your Dragon |
| Profiles | 560, 660, 760, 860, 1000, 2000, 4000 Kbps |
| Clients | Mac Mini (115), iPad (112), iPad (114), iPad (111), iPhone (117) |
| Server | Best Effort |

1. Each client started in sequence; all clients settled to 2 Mbps profile (Graph 1).
2. After 10 minutes iPad (114) goes to 1 Mbps profile; 9 Mbps pipe utilization.
3. Stopped iPad (114); 8 Mbps or 100% utilization (Graph 2).
4. Stopped iPad (111); 6 Mbps or 60% utilization (Graph 2).
5. After approximately 10 minutes, Mac Mini (115) jumped to 4 Mbps profile; still at only 80% pipe utilization.
6. Eventually, after approximately 30 minutes, system achieved 10 Mbps or 100% utilization (Graph 3).


Graph 1: Five clients started in sequence.




Graph 2: From four to three clients, 60% utilization.

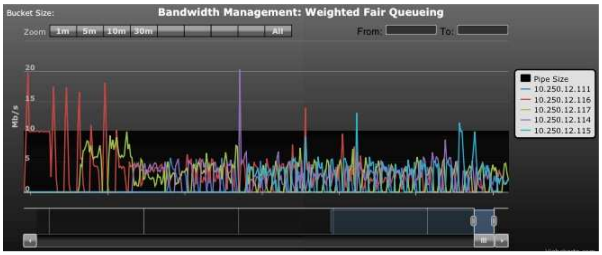

Graph 3: Final state achieved: 100% utilization.

This result was unexpected. It appears that the three clients (Graph 2) became locked in a synchronous pattern and as a result measured a lower segment-download rate than expected. As a result, none of the clients moved to a higher-rate profile, even though adequate bandwidth (4 Mbps) remained unused. Eventually (Graph 3) this pattern corrected itself, but not until approximately 30 minutes had elapsed.

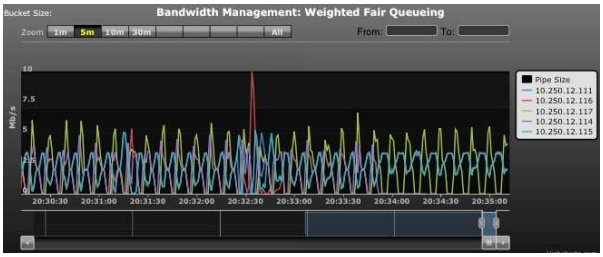
Run 2: Weighted Fair Queuing

| | |
|---|---|
| Start Time | 15:27:00 GMT |
| Pipe | 10 Mbps |
| Content | How to Train your Dragon |
| Profiles | 560, 660, 760, 860, 1000, 2000, 4000 Kbps |
| Clients | Mac Mini (115), iPad (112), iPad (114), iPad (111), iPhone (117) |
| Server | Weighted Fair Queuing |
| Weighting Factor | All clients set to 1 |

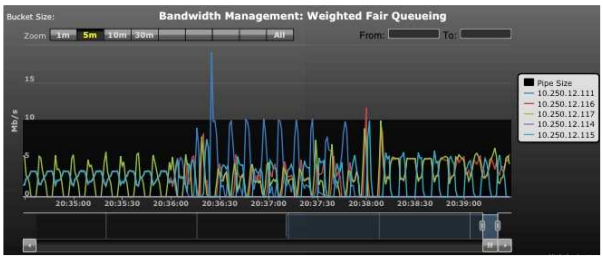
1. Each client was started in sequence (as before); same result as best-effort case (Graph 4).
2. iPad (111), iPad (116), and Mac Mini (115) all occasionally reached the 1 Mbps profile. Pipe stays at very close to 100% utilization.
3. Stopped iPad (114); immediately remaining clients achieved 100% pipe utilization. After 2 min iPad (111) reached the 4 Mbps profile (Graph 5).
4. Stopped iPad (111); pipe at 90% and then increased to 100% utilization (Graph 6).



Graph 4: Five clients, 100% utilization.




Graph 5: Four clients, 100% utilization.



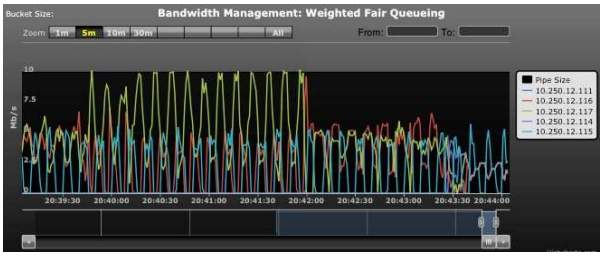

Graph 6: Three clients, 100% utilization.

It is apparent from Graphs 5 and 6 that the throughput of the system is much higher than in the best-effort case. In all cases, the pipe was utilized at, or close to, 100%. This may be understood by considering the scheduling algorithm at the HTTP server sequences through each partial segment download. Hence the clients take turns to achieve a more efficient segment download and therefore request a higher profile than in the best-effort case. The pipe throughput is maximized by the scheduling algorithm.

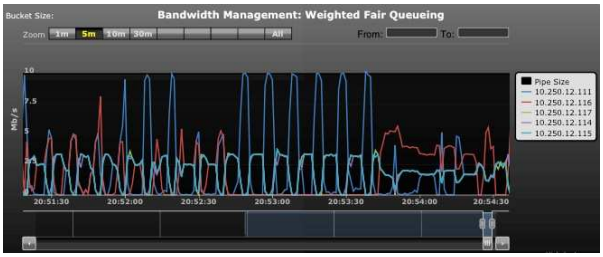
Run 3: Weighted Fair Queuing

| | |
|---|---|
| Start Time | 20:39:00 GMT |
| Pipe | 10 Mbps |
| Content | How to Train your Dragon |
| Profiles | 560, 660, 760, 860, 1000, 2000, 4000 Kbps |
| Clients | Mac Mini (115), iPad (112), iPad (114), iPad (111), iPhone (117) |
| Server | Weighted Fair Queuing |
| Weighting Factor | iPad(116) = 2, 3, 4 |

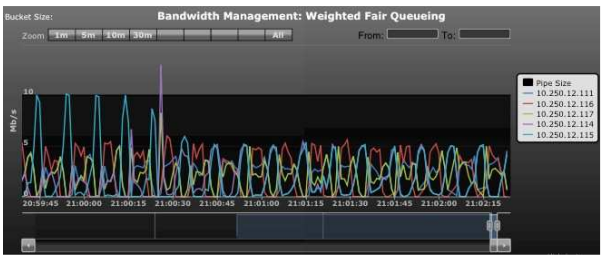
1. All clients started - no premium factor applied (Graph 7).
2. iPad (116) stopped.
3. Premium factor 2 applied to iPad (116) and started - quickly ramped to 2 Mbps (Graph 8).
4. Premium increased to 3. 115 went to 4 Mbps (momentarily).
5. Premium increased to 4. 114 went to 4 Mbps (Graph 9).




Graph 7: Fair network queuing.




Graph 8: Weighted fair queuing: iPad (116) weighting factor = 2.



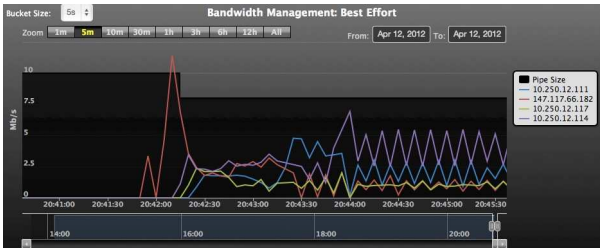

Graph 9: Weighted fair queuing: iPad (116) weighting factor = 4.

The weighting factor allows the premium client to achieve a higher profile than in Run 2, but it did not achieve the highest profile, probably because it was such a large jump in bit rate from 2 Mbps to 4 Mbps. Therefore, it was decided to re-run the test with a closer set of profile rates.

Run 4: Best Effort

| | |
|---|---|
| Start Time | 20:42:00 GMT |
| Pipe | 8 Mbps |
| Content | Promo Reel |
| Profiles | 400, 600, 910, 1200, 1600, 2000 Kbps |
| Clients | iPad (111), Mac Mini (182), iPhone (117), iPad (114), iPad (116) |
| Server | Best Effort |

1. iPad (111), Mac Mini (182), iPhone (117), and iPad (114) started (Graph 10 and Figure 5).
2. 5th client iPad (116) started at 20:46:25 (Graph 11 and Figure 5).




Graph 10: Best effort, four clients.

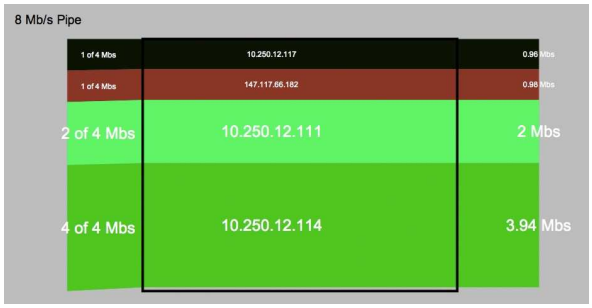
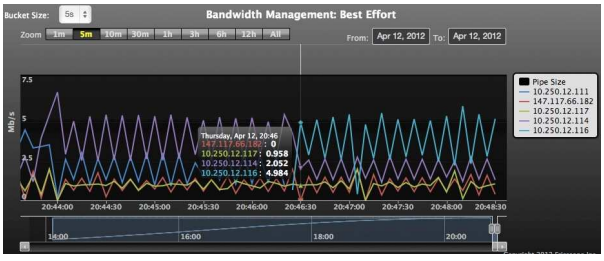



Figure 5: Best effort, four clients




Graph 11: Best effort, fifth client started at t = 20:46:25.

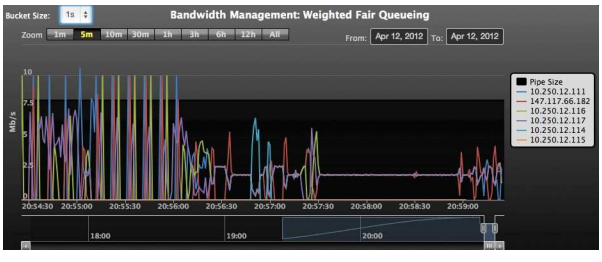

Figure 6: Best effort bit-rate allocation.

In this case, the bandwidth was shared unfairly. We hypothesize that the clients that first achieved the highest profile (2 Mbps) were able to maintain an unfair share of the pipe because of a positive feedback effect.

Run 5: Weighted Fair Queuing

| | |
|---|---|
| Start Time | 20:54:30 GMT |
| Pipe | 8 Mbps |
| Content | Promo Reel |
| Profiles | 400, 600, 910, 1200, 1600, 2000 Kbps |
| Clients | iPad (111), Mac Mini (182), iPhone (117), iPad (116), iPad (114) |
| Server | Weighted Fair Queuing |
| Weighting Factor | iPad(114) = 3 |

1 iPad (111), Mac Mini (182), iPhone (117), and iPad (114) started (Graph 12 and Figure 7).
2 iPad(114) with weighting factor of 3 started, and quickly ramped to 2 Mbps (Graph 13 and Figure 8)




Graph 12: Four clients, fair network queuing.

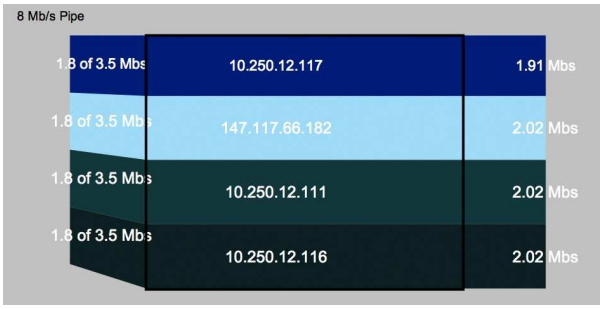
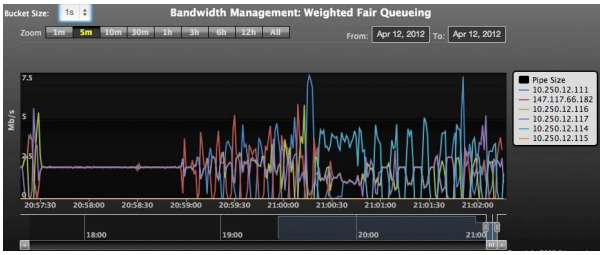



Figure 7: Four clients, fair network queuing.




Graph 13: Five clients, WFQ iPad (114) weighting factor = 3


Figure 8: Five clients, WFQ iPad (114) weighting factor = 3

In this case, bandwidth was allocated equally (Figure 7) with four equally weighted clients. Subsequently, a greater share of bandwidth was allocated to a premium client (Figure 8) with a weighting factor of 3.

## CONCLUSIONS

Implementation of a bit-wise, round-robin scheduler at the HTTP server can be used to effectively enforce fairness amongst HTTP adaptive streaming clients. In addition, a weighting factor may be established for a "premium" client to ensure that it experiences greater throughput during periods of access network congestion.

It appears that the weighted fair queuing mechanism [7] implemented in the prototype system is effective because it operates at the HTTP server layer, which is at a higher layer in the network stack than TCP congestion control alone [8], [9], and because it operates over a significantly longer time frame. If a client tries to "cheat" the system by requesting a higher profile than can be sustained by the network, it will only impact its own performance and not that of other clients.

## IMPLEMENTATION OPTIONS

In order for the weighted fair queuing algorithm to be effective it must be implemented at the point in which traffic converges on a shared link in the access network. In the case of a DOCSIS access network, each downstream service group must be treated separately. The algorithm is implemented at the HTTP server (at the edge cache) and a virtual pipe must be created to each downstream service group (as illustrated in Figure 9).

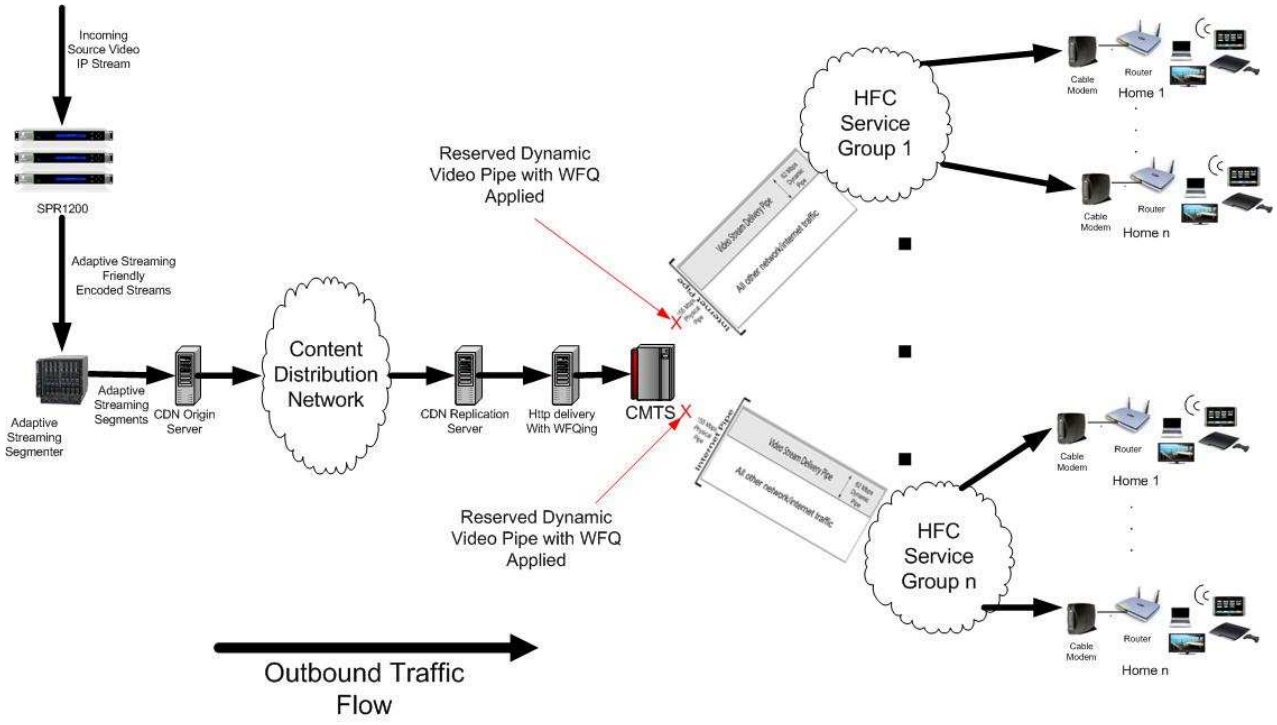



Figure 9: Implementation in DOCSIS Network

References


1. R. Pantos and W. May. HTTP Live Streaming. IETF Draft, June 2010.
2. A. Zambelli. IIS smooth streaming technical overview. Microsoft Corporation, 2009.
3. Adobe HTTP Dynamic Streaming. http://www.adobe.com/products/hds-dynamic-streaming.html.
4. ISO/IEC 23009-1:2012 – Information Technology – Dynamic Adaptive Streaming over HTTP.
5. Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. Mac MiniSys'11, February 23–25, 2011, San Jose, California, USA.
6. Bing Wang, Jim Kurose, Prashant Shenoy, and Don Towsley. Multimedia streaming via TCP: An Analytic Performance Study. Department of Computer Science, University of Massachusetts.
7. Martin J. Fischer, Denise M. Bevilacqua Masi, and John F. Shortle. Simulating The Performance of a Class-based Weighted Fair Queuing System. Proceedings of the 2008 Winter Simulation Conference.
8. Robert Kuschnig, Ingo Kofler, and Hermann Hellwagner. An Evaluation of TCP-based Rate-Control Algorithms for adaptive Internet streaming of H.264/SVC. Institute of Information Technology (ITEC) Klagenfurt University, Austria.
9. Luca De Cicco and Saverio Mascolo. An Experimental Investigation of the Akamai Adaptive Video Streaming. Dipatimento di Elettrotecnica ed Elettronica, Politecnico di Bari.