What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth?

Saamer Akhshabi,
Lakshmi Anantakrishnan,
Constantine Dovrolis,
College of Computing
Georgia Institute of Technology
s.akhshabi, lakshmi3, constantine@gatech.edu

Ali C. Begen
Video and Content Platforms Research and Advanced Development
Cisco Systems
abegen@cisco.com

ABSTRACT

With an increasing demand for high-quality video content over the Internet, it is becoming more likely that two or more adaptive streaming players share the same network bottleneck and compete for available bandwidth. This competition can lead to three performance problems: player instability, unfairness between players, and bandwidth underutilization. However, the dynamics of such competition and the root cause for the previous three problems are not yet well understood. In this paper, we focus on the problem of competing video players and describe how the typical behavior of an adaptive streaming player in its `Steady-State`, which includes periods of activity followed by periods of inactivity (`ON-OFF` periods), is the main root cause behind the problems listed above. We use two adaptive players to experimentally showcase these issues. Then, focusing on the issue of player instability, we test how several factors (the `ON-OFF` durations, the available bandwidth and its relation to available bitrates, and the number of competing players) affect stability.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Performance, Measurement, Algorithms

Keywords

Adaptive streaming, video streaming over HTTP, player competition, available bandwidth competition, stability

1. INTRODUCTION

Adaptive video streaming over HTTP is quickly getting adopted as the technology of choice for the delivery of video over IP networks. At the same time, the popularity of Internet-based high-quality streaming for various end-user devices such as HDTVs, mobile phones, gaming devices, and computers is on a steady rise as well. The bandwidth requirement for such devices is rapidly increasing as the content quality is improving to meet end-user demands. With abundant video content and increasing bandwidth demands, it is becoming likely that two or more adaptive streaming players have to share a network bottleneck and compete for available bandwidth. Such competition can take place, for example, when several people in the same house watch different movies at the same time. In that case the residential broadband access link is probably going to be a shared bottleneck. Another instance of such competition is when many users watch the same live event (such as Super Bowl) online. An edge network link may constitute the shared bottleneck in that case. It has been previously observed that such competition can lead to performance issues [1, 3, 4]. However, the dynamics of such competition and the root cause for the previous performance problems are not yet well understood.

Our goal in this paper is to identify the underlying root cause of the performance problems that take place when multiple adaptive streaming players compete. We group these problems into three categories: The first relates to the stability of the players in terms of requested bitrates and video quality. The second is the unfairness among competing players. The third is the potential bandwidth underutilization when multiple adaptive players compete.

We first give a short overview of how a typical adaptive streaming algorithm works. Then, we show that this typical behavior can lead to periods during which the player stays idle – downloading nothing from the server. The player cannot estimate the available bandwidth in the bottleneck during these idle times. Depending on when these idle times start and end for each competing player, the previous three issues (instability, unfairness, and underutilization) can arise. In this paper we only focus on the performance of the players in the `Steady-State` in which the player has already built its playback buffer. We experimentally evaluate two adaptive streaming players, the Smooth Streaming player [7] and a variant of the AdapTech Streaming player introduced in [2], when two or more players compete with each other. Finally, focusing on the instability issue, we examine how 1) the relative duration of the player idle periods, 2) the "fair share" of each player and its relation to the available bitrates, and 3) the number of competing players, affect the stability of the system.

To the best of our knowledge, there have only been two prior studies focusing on the competition between adaptive streaming players. Cloonan and Allen [3] developed a simulator to explore corner-case scenarios that may occur with competing streaming sessions. They reported requested bitrate oscillations and unfairness when multiple adaptive streaming players compete and they observed that the instability tends to increase with the number of competing players (we disagree with this point in Section 5). Houdaille and Gouache [4] observed instability and unfairness with competing adaptive streaming players and they proposed a traffic shaping method at home gateways to reduce the extent of these problems.

The rest of this paper is organized as follows: In Section 2, we describe the root cause of the previous performance problems when multiple players compete. In Section 3, we describe the experimental methodology and metrics that we use. In Section 4, we showcase how competition in real scenarios can lead to performance degradation. Section 5 focuses on the stability of the competing adaptive streaming players and studies the various factors that can affect it. We conclude the paper in Section 6.

2. MULTIPLE COMPETING PLAYERS

In this section we describe qualitatively three performance issues that can take place when two or more adaptive streaming players share a network bottleneck and compete for available bandwidth. An important point is that the issues we focus on in this paper, are *not* due to TCP dynamics, as is often reported, but they mainly arise from the rate adaptation algorithms at the application layer.

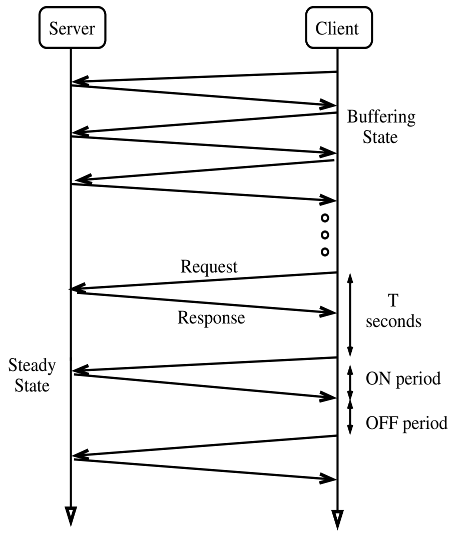

Figure 1: The request-response timing between client and server in the Buffering and Steady states.

An adaptive streaming player typically starts a streaming session in the `Buffering-State` [1]. At this phase, the goal of the player is to build up its playback buffer as quickly as possible and to reach a maximum buffer size. To do so the player requests a new chunk (also known as "fragment" or "segment") as soon as the previous chunk is downloaded.

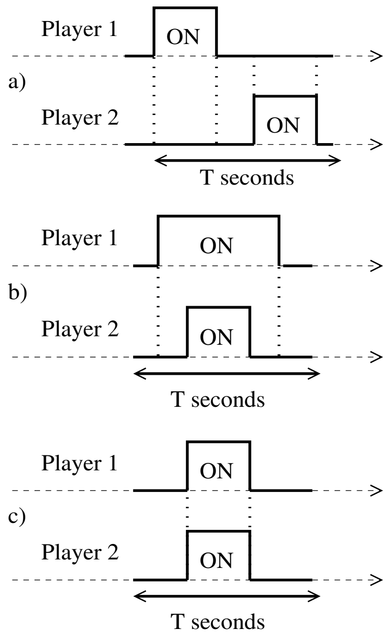

Figure 2: Three instances of the ON-OFF periods of two competing players during one chunk download period.

Once the playback buffer size reaches a certain target (e.g., 30 seconds), the player switches to the `Steady-State` during which it aims to maintain a constant playback buffer size. Assuming for simplicity that each chunk corresponds to T seconds of content, the player requests one chunk every T seconds (if the download duration is less than T) or as soon as the previous chunk is received (otherwise). This can lead to an activity pattern in which the player is either `ON`, downloading a chunk, or it is `OFF`, staying idle. This pattern is illustrated in Figure 1.

In parallel, the player estimates its fair share in the underlying network path by measuring the per-chunk TCP throughput, and computing a running average of those measurements over time [1]. The player then uses that running average to select the bitrate for the next requested chunk.

Depending on the temporal overlap of the `ON-OFF` periods among competing players, they may not estimate their fair share correctly. This can cause the following three performance problems: instability, unfairness, and underutilization. Specifically, consider a simple model with two adaptive players sharing a bottleneck of capacity C. Suppose that both players have already reached the `Steady-State` requesting a new chunk every T seconds. Also, let us ignore for now the well-known TCP shortcomings, and assume that a single active connection gets the whole capacity C, while two active connections share the capacity fairly, receiving $\frac{C}{2}$ each. The fair share for each player, denoted by f, in this model is $\frac{C}{2}$. We denote by f_1 and f_2 the throughput received by player-1 and player-2, respectively, during a chunk download period. Ideally, it should be that $f_1 = f_2 = f$.

Figure 2-a shows the case where the `ON` periods of the two players do not overlap during a chunk download period. Both players measure a per-chunk throughput of C, and so they estimate that $f_1 = f_2 > f$. In other words, both players overestimate their fair share by a factor of two. When both players overestimate their fair share, and depending on the video bitrates, they may request a profile with higher bitrate than f, causing congestion. When that happens, the players will measure that their TCP throughput is less than their previous fair share estimate, and so they will switch back to a lower video bitrate. This oscillatory scenario can repeat, causing instability.

Figure 2-b shows the situation where the ON period of one player falls within the ON period of the other player. This can happen if one player is requesting a chunk with lower bitrate than the other player. In this case, the former observes a throughput of $\frac{C}{2}$ and the latter observes a throughput that is more than $\frac{C}{2}$. So the two players estimate that $f_1 > f_2 = f$, which means that one player overestimates its fair share. When only one player overestimates its fair share, it can be that the two players converge to a stable but unfair equilibrium in which the player with the larger fair share estimate requests a higher bitrate video.

Figure 2-c shows the situation where the ON period of the two players are perfectly aligned. Both players observe a throughput of $\frac{C}{2}$ and so $f_1 = f_2 = f$. In this case the two players estimate their fair share correctly. Note however that even in this case we can have bandwidth underutilization. To illustrate, suppose that the video has two available profiles with bitrates b_1 and b_2, respectively. Then, both examples in Figure 2-b and 2-c can be stable if $b_1 < \frac{C}{2}$, $b_1 + b_2 < C$, and $b_2 > \frac{C}{2}$. However, the case shown in Figure 2-c, where both players request the b_1 profile, causes underutilization, even though it is stable and fair.

Instability can also cause underutilization. Suppose that $b_1 << b_2$ and $b_1 + b_2 \approx C$. Consider the case that one player requests b_1 and the other requests b_2, which is stable and the capacity of the bottleneck is completely utilized. On the other hand, consider the case that the players oscillate between b_1 and b_2, given that it is not possible that they both receive the b_2 profile. This oscillatory scenario will lead to significant underutilization, when both players request b_1.

The previous examples illustrate some competition scenarios for two adaptive streaming players. In reality, several other factors can play an important role in the appearance and extent of instability, unfairness and underutilization, such as the exact player adaptation algorithm, TCP dynamics, bandwidth fluctuations, and the variability of the video encoding rate. In the rest of this paper, we use actual adaptive streaming players to demonstrate that the previous issues can still arise in reality.

3. METHODOLOGY AND METRICS

In this section, we give an overview of the experimental methodology and define the metrics we focus on. The experimental set up is similar to that in [1]. The host that runs the video players also runs a packet sniffer (Wireshark [5]) and a network emulator (DummyNet [8]). Wireshark allows us to capture and analyze offline the traffic from and to the HTTP server. DummyNet allows us to control the *downstream available bandwidth* (also referred to as *avail-bw*) that the players can receive. That host is connected to the Georgia Tech campus network through a Fast Ethernet interface. The TCP connections that transfer video and audio streams cannot exceed (collectively) the avail-bw at any time.

We use two different players in the following experiments. The first is a variant of the AdapTech-Streaming player introduced in [2], which is based on the Adobe OSMF player [6]. We have instrumented the player to log its internal parameters such as playback buffer size, requested bitrate, chunk download time, and chunk throughput over time. The second is the commercial Smooth Streaming player [7]. We infer the previous parameters for that player by using packet captures, as described in [1]. Due to space constraints, we do not repeat the details in this paper.

We represent by P_r an encoding bitrate of r Mbps that is available for a given video stream at the server (e.g., $P_{2.75}$). We test competing players under constant avail-bw. We do not, however, control the servers and the content ourselves and so we select the avail-bw according to the bitrates available at the servers. An experiment ends when at least one of the players finishes receiving the whole video.

We define the following three performance metrics:
1. The *instability* metric, denoted by θ, is the fraction of successive chunk requests by a player in which the requested bitrate does not remain constant.
2. The *unfairness* metric (for two players) is the average of the absolute bitrate differences between the corresponding chunks requested by each player.
3. The *utilization* metric is defined as the aggregate throughput during an experiment (measured from the Wireshark captures) divided by the avail-bw in that experiment.

We first test the Smooth Streaming player when two players compete under constant avail-bw. We also use this player to study the effect of two factors: the duration of the ON-OFF periods, and the fair share of each player relative to the available bitrates. Due to resource constraints, for example CPU and GPU power, on the machine hosting the players, we cannot run multiple instances of the Smooth Streaming player to investigate the effect of the number of competing players. Therefore, we have developed a *simpler player* that mimics the behavior of Smooth Streaming at a qualitative level, without actually decoding and displaying the video streams, to study the effect of the number of players (see Section 4.2).

All experiments are performed on a Windows 7 Professional host with an Intel(R) Core(TM)i5 CPU M480 2.67GHz processor, 4.00 GB physical memory, and an Intel(R) HD Graphics processor with 1307 MB total memory.

4. BASIC EXPERIMENTS

We performed many experiments with the Smooth Streaming player and observed that the performance issues described in Section 2 also take place in practice. Due to space constraints, we show results here for only one of those experiments. We also show results from a single experiment with a simpler player, which is based on the AdapTech Streaming player introduced in [2].

4.1 Smooth Streaming Player

We use Microsoft Silverlight Version 5.0.61118.0 provided by Microsoft at the IIS Web site.[1] The manifest file for the video that is provided there declares eight video encoding bitrates ranging from 350Kbps to 2.75 Mbps. Figure 3 shows

[1]http://www.iis.net/media/experiencesmoothstreaming

the requested bitrates, the chunk throughputs as well as the number of active (i.e., ON) players for an experiment with two Smooth Streaming players sharing a bottleneck with 1.6 Mbps avail-bw. Note that we only focus on the time interval between $t = 80$ s and $t = 280$ s, when both players are in `Steady-State`. When the ON periods of the two players do not overlap (i.e., the number of active players is less than two, e.g., between $t = 140$ s and $t = 160$ s), the chunk throughputs are much larger than the fair share (0.8 Mbps). This happens when the players are requesting lower bitrates ($P_{0.47}$ and $P_{0.63}$). We do not know exactly how the Smooth Streaming player uses the chunk throughput measurements to estimate the avail-bw but we can observe that the players decide then to switch to a higher bitrate (e.g., at $t = 157$ s to $P_{0.845}$) that is larger than the fair share, which is unsustainable. The durations of the ON periods increase then, the number of active players becomes two, and the measured throughput by each player decreases. Consequently, the players switch back to the lower bitrates at around $t = 182$ s. This oscillatory pattern continues throughout the streaming session. Overall, this experiment's instability is 12%, the unfairness is 0.085 Mbps, and the utilization is 94%.

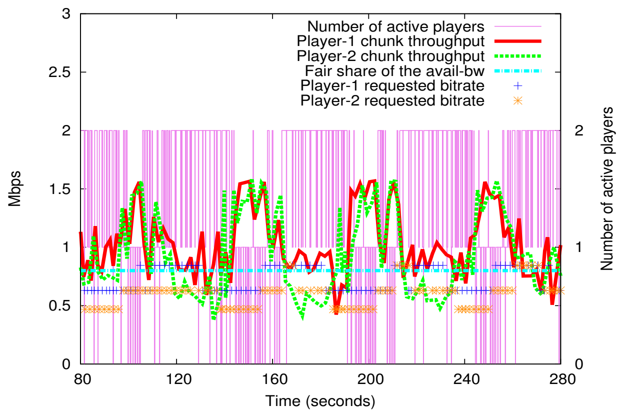

Figure 3: Requested bitrates and chunk throughputs for two competing Smooth Streaming players.

4.2 Simpler Player

Our goal here is to design a simpler player that behaves similar to Smooth Streaming, at least qualitatively, while allowing us to run a large number of players at the same host. To do so, we use a variant of the AdapTech Streaming player introduced in [2]. We simplify the player and choose the parameters of the adaptation algorithm based on our experiments in [2] to match the Smooth Streaming player. This player also has two states: `Buffering` and `Steady-State`. The maximum buffer size is set to 30 seconds. The player maintains two throughput-related metrics, the throughput of the latest downloaded chunk, denoted by A, and a running average of A, denoted by $\hat{A}$. If $A(i)$ is the throughput of the i'th chunk, the running average $\hat{A}$ is:

$$\hat{A} = \begin{cases} \delta\hat{A}(i-1) + (1-\delta)A(i) & i > 0 \\ 0 & i = 0 \end{cases}$$

with $\delta = 0.8$.

The player starts a streaming session requesting the lowest available profile. It selects the next profile based on $\hat{A}$, as follows. Assume that the bitrate of the ith profile is b_i and the rank of the current profile is ϕ_{cur}. Then the index of the next candidate profile ϕ is given by

$$\phi = \max\{i : b_i < c \times \hat{A}\}$$

where c is a slack parameter ($0 < c < 1$) that is necessary due to the variability of the video encoding rate and the temporal bandwidth fluctuations – we use c=0.8. The requested profile for the next chunk is determined as follows

```
if $\phi > \phi_{cur}$ then
    Increase the requested bitrate by one profile.
else if $\phi < \phi_{cur}$ then
    Decrease the requested bitrate by one profile.
else
    Stay with the current requested bitrate
end if
```

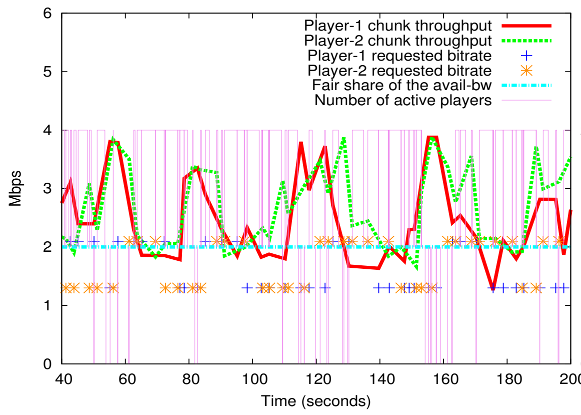

Figure 4: Requested bitrates and chunk throughputs for two competing "simpler" players.

In the following experiment, we use a movie trailer ("Freeway") provided by Akamai's HD-video Web site for Adobe HTTP Dynamic Streaming.[2] We use four different encoding bitrates between 0.9 Mbps and 2.5 Mbps. Figure 4 shows the requested bitrates, the chunk throughputs, and the number of active players from $t = 40$ s to $t = 200$ s for two competing players sharing a bottleneck with 4 Mbps avail-bw. The players show an instability pattern similar to that with the Smooth Streaming players. The instability metric for this experiment is around 16%, the unfairness metric is 0.27 Mbps, while the utilization is 92%.

The experiments with Smooth Streaming as well as with our simpler player show that these clients do a reasonable job regarding fairness and utilization. However, the players fail to address the root cause of the instability problem described in Section 2. In the next section, we focus on the instability issue and show how certain key factors affect the stability of the adaptive streaming players.

[2]http://zeridemo-f.akamaihd.net/content/inoutedit-mbr/inoutedit_h264_3000.f4m

5. STABILITY

In this section we focus on three factors that affect the instability of competing adaptive players. The first factor is the relative duration of the `OFF` periods in the activity pattern of each player. The second factor is the fair share of each player relative to the bitrates of the available video profiles. The third factor is the number of competing players. For the first two factors we use the Smooth Streaming player, while for the third factor we use our simpler player. We summarize the metrics and parameters used in this section in Table 1.

Metric	Summary
f	Bandwidth fair share for each player
θ	Instability metric
γ	Fraction of time: exactly one player is idle
μ	Fraction of time: both players overestimate f
λ	Overlap factor
N	Number of competing players

Table 1: List of metrics and parameters in Section 5

5.1 Duration of ON-OFF Periods

How does the duration of the competing players' `ON-OFF` periods affect their stability? We denote the fair share of each player, i.e., the total avail-bw divided by the number of competing players, by f. We use two competing Smooth Streaming players, starting with $f = 400$Kbps and increasing the fair-share in steps of 100Kbps until f=3 Mbps. We repeat each experiment four times for each value of f.

We measure the instability metric θ for each streaming experiment. Additionally, we denote by γ the fraction of the streaming session's duration in which exactly one player is idle, i.e., in the `OFF` state. $\gamma = 1$ if at any point in time exactly one player is in the `OFF` state while the other player in the `ON` state. $\gamma = 0$ when the players' `ON-OFF` periods are perfectly aligned in time (as in Figure 2-c). We then match each player-1 request to the player-2 request that is closest in time. We denote by μ the fraction of time where both players estimate a throughput that is larger than f, based on these time-matched requests.

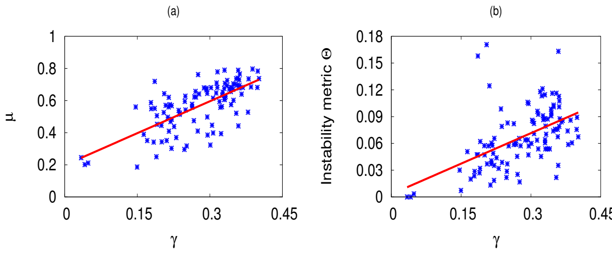

Figure 5: a) μ, the fraction of time where both players estimate a throughput that is larger than their fair share f, and b) the instability metric (θ) as a function of the fraction of time in which exactly one player is idle (γ).

Figure 5-a shows a scatter plot of μ and γ for all experiments. The correlation between the two metrics is 0.70. When only one player is `OFF`, the `ON` player observes the entire capacity and so it overestimates the fair-share f. Recall that the overestimation of f, as described in Section 2 and showcased in Section 4, can trick the player to switch to a bitrate that is higher than f, which is unsustainable. This in turn leads to unnecessary bitrate changes and instability. Figure 5-b shows a scatter plot of the instability metric θ and the fraction γ. Again, we see a clear positive correlation between the fraction of time in which one player is idle (γ) and the instability of the requested bitrates. The correlation coefficient is 0.52.

5.2 Fair Share and Available Profiles

Does the fair share of each player (f) affect the stability of the system? Suppose that there are only two available profiles at the video server. We perform the following set of experiments: start with a fair share that is equal to the lower profile bitrate, and increase f in steps of 50Kbps until it exceeds the higher profile bitrate. For each value of f, two Smooth Streaming players compete for bandwidth. We repeat the experiment eight times for each value of f.

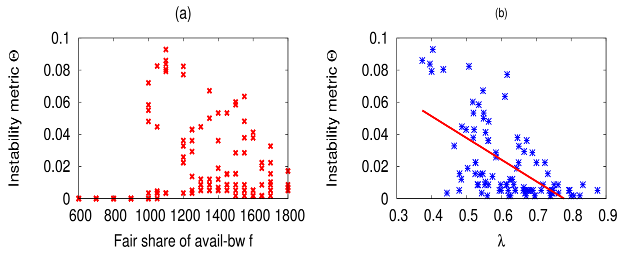

Figure 6: a) The instability metric θ as a function of the fair share f, and b) θ as a function of the player overlap factor λ.

Figure 6-a is a scatter plot showing the instability metric θ as function of the fair-share f. The two profile bitrates are 470Kbps and 1.52 Mbps. Note that instability is always 0 when $f < 1$ Mbps. This is because the avail-bw in that range is not sufficient for even one player to switch to the higher profile (that would require at least 2.1 Mbps of avail-bw, including audio). At the other extreme, if $f \geq 1.8$ Mbps the avail-bw is sufficient for both players to switch to the higher profile and so the instability is again close to 0.

The more interesting case is the range $1 < f < 1.8$ Mbps, in which there is high variability in the stability of the players across different experiments. In this range of f at least one of the players can switch to the higher profile, triggering instability.

However, there are also several experiments in that range of f in which the instability is close to 0. Further analysis of those experiments revealed that they correspond to the case of Figure 2-b, where the `ON` period of one player almost falls within the `ON` period of the other. The former is requesting the lower profile, while the latter is requesting the higher profile. To demonstrate this point, we calculate the fraction of the `ON` duration of one player in which the other player is also `ON`. We then take the maximum of that value among the two players. We denote this maximum by λ. $\lambda = 1$ if the `ON` period of one player always falls within the `ON` period of the other.

Figure 6-b shows a scatter plot with the instability metric θ and the "overlap" factor λ. There is a clear negative correlation between the two metrics (correlation coefficient -0.61). The more *overlapping* the `ON` periods of the two competing players are, the more stable they become, as expected based on Figure 2-b.

5.3 Number of Competing Players

Assuming that the capacity of the bottleneck link is fixed, is there a relation between the stability of the system and the number of competing players? We examine this question with the following experiments using our simpler player.

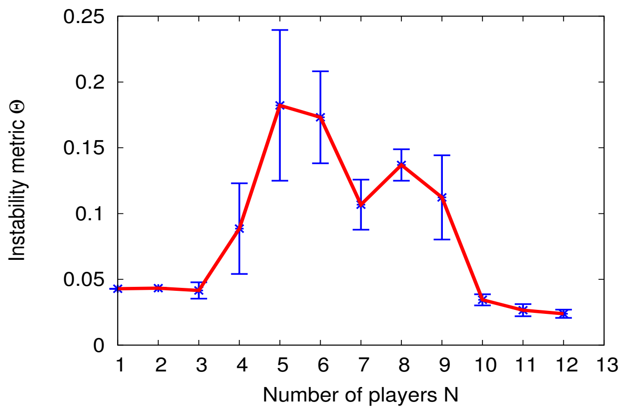

Figure 7: The instability metric θ as a function of the number of competing players N.

Figure 7 shows the instability metric θ as a function of the number of competing players, denoted by N. The players share an 11 Mbps bottleneck. Note that the lowest and highest available profiles are $P_{0.9}$ and $P_{2.5}$ with two other profiles in between. Each point is obtained by repeating the experiment four times and averaging the instability values across all experiments. The corresponding 95% confidence intervals are also shown.

The players start streaming about five seconds apart. When $N < 4$, the fair share of each player f is sufficient for all players to switch to the highest profile $P_{2.5}$. So, the instability factor is then small (only a few initial rate changes before stabilizing). On the other extreme, when $N > 10$, the fair share f is less than 1.1 Mbps and so it is barely sufficient for the players to receive the lowest profile (including the audio stream, which requires 64 Kbps). The players would then stay in the `Buffering-State` and rarely switch to a higher profile, meaning that the instability is close to 0.

As N increases from 4 to 10, however, f falls within the lowest and the highest available bitrates. Then, the players are mostly in the `Steady-State`, and they go through the previously discussed ON-OFF activity pattern. The instability increases until it peaks at $N = 5$ or 6. This corresponds to approximately a fair share value that is half way between the lowest and highest profiles. As we increase N further, f decreases, all players are forced to request lower bitrates, and instability decreases.

6. CONCLUSIONS

We described how the competition for available bandwidth between multiple adaptive streaming players can lead to instability, unfairness, and bandwidth underutilization. We identified the root cause of the problem as the behavior of adaptive players in the `Steady-State` phase; that phase includes periods of activity (`ON` periods) followed by periods of inactivity (`OFF` periods). A player cannot estimate the available bandwidth during `OFF` periods because it does not transfer any data then. We conducted experiments with real adaptive streaming players and showed that the previous issues can arise in practice. Finally, we showed how certain factors, namely the duration of `ON-OFF` periods, the fair share relative to the available profile bitrates, and the number of competing players, can affect the stability of the system.

In future work, we plan to expand the previous study with analytical and computational models for adaptive streaming systems. Another goal is to propose a solution that takes into account the several dimensions of this problem including the complexity of the player's adaptation logic and its interaction with the underlying transport protocol and congestion control mechanism.

7. REFERENCES

[1] S. Akhshabi, A. C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. *ACM MMSys*, 2011.

[2] S. Akhshabi, S. Narayanaswamy, A.C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptive video players over http. *Signal Processing: Image Communication*, 27:271–287, 2012.

[3] T. Cloonan and J. Allen. Competitive analysis of adaptive video streaming implementations. *SCTE Cable-Tec Expo Technical Workshop*, 2011.

[4] R. Houdaille and S. Gouache. Shaping http adaptive streams for a better user experience. *ACM MMSys*, 2012.

[5] A. Orebaugh, G. Ramirez, J. Burke, and J. Beale. *Wireshark and Ethereal network protocol analyzer toolkit*. Syngress Media Inc, 2007.

[6] OSMF Player. `http://www.osmf.org`.

[7] Smooth Streaming Player. `http://www.iis.net/download/SmoothClient`.

[8] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM CCR*, 27(1):31–41, 1997.