What happens when HTTP adaptive streaming players compete for Bandwidth

Ziyan Qiu[1], Digvijay Gorle[2]
New York University, Brooklyn, New York, USA

Background

People these days depend on a lot in watching movies from YouTube, Netflix, or other video platforms. The need to provide high quality content offer the users in a long run without effecting the bandwidth and prevent network congestion at the same time is a task that requires a lot of effort and is an ongoing process in many companies. In this paper we are going to discuss some of the results and analysis done by a team to understand the congestion, bandwidth, unfairness, and other parameters. This analysis enables us to understand the network congestion better and helps us to come up with algorithms to deal with such situations.

We are going to replicate the following figure and understand 3 major parameters or metrics. The metrics are as follows.

1) The instability metric which is defined as the fraction of successive chunk that is requested by the player in which the bit rate does not seem to be constant.
2) The unfairness rate which is calculated to see the average bit rate of 2 players when each player requests a chunk from the network.
3) The utilization metric which is defined as the throughput that is aggregate in the experiment. This is divided by the available bandwidth gives us the utilization of the bandwidth.

The graph we are going to reproduce in our experiment is as follows:

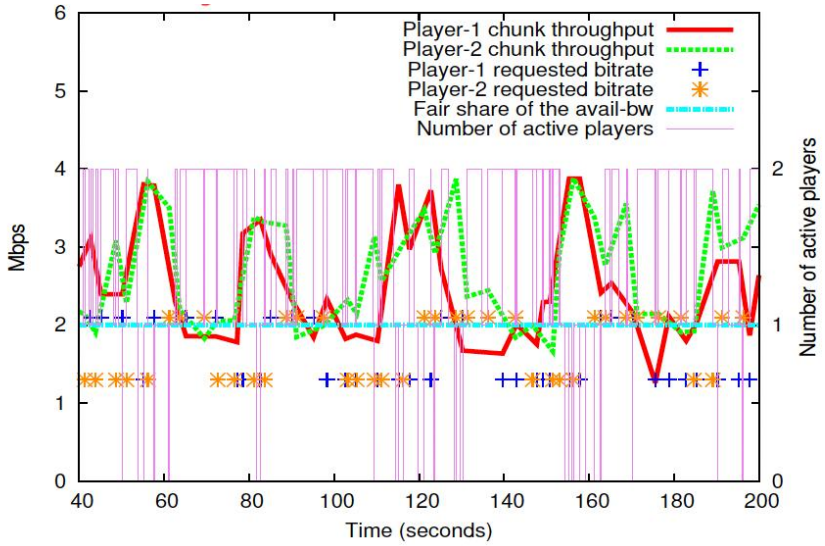

Figure 1: Graph to plot

Akhshabi, Saamer & Anantakrishnan, Lakshmi & Begen, Ali & Dovrolis, Constantine. (2012). What happens when HTTP adaptive streaming players compete for bandwidth?. 10.1145/2229087.2229092.

The above graph gives a lot of information about many things and the useful information can be used to understand how the bandwidth is being utilized between the 2 players. We can see that the data we receive is a graph that shows the activity of the 2 players. The 2 players show different bitrate, throughput based on the state of each player. This allows us to come up with different algorithms to deal with the network congestion at different times.

Reference paper methodology

In the reference paper the host ran a Wireshark as a packet sniffer and used a network emulator DummyNet in this instance. DummyNet allows us to make changes to the available downstream bandwidth that each of the players can receive. The host is connected to the Georgia Tech campus.
In this paper 2 different players are being used for the analysis of the network congestion when 2 players request and fight for the available bandwidth. The first player that is used is the AdaptiveTech streaming player and the second player that is used is the Smooth Streaming Player. First, they set the smooth streaming player for their testbed. 2 main factors are being considered when they are using this setup, the duration of On-Off periods and the fair share of each player in relation to the available bandwidths. Since there were CPU and GPU constraints, they paper did not explicitly use the Smooth streaming Player to get the results. Instead, a simpler player was made to mimic the Smooth Streaming Player. From this they were able to get the desired graphs and make an analysis out of it.
The Microsoft Smooth Streaming Player that is provided by the IIS website was used in the experiment to get the desired graph. There are 8 different bit rates that are ranging from 350 Kbps to 2.75 Mbps. From the graph already displayed above, we can see that the players are sharing a bottleneck bandwidth of 1.6 Mbps. We can also see various information such as the bitrates requested, chunk throughput, number of players that are active, the fair share of each player. From the graph we can see that only the time interval between 80 seconds and 280 seconds was taken into the consideration.
We can deduce that, when the On states of the 2 players are not aligned together, we can see that the chunk throughputs are much larger than the fair share. From their analysis they calculated that instability rate is 12%, the utilization is 94% and the unfairness is 0.085 Mbps.
After this a simpler player was made and they were able to get the following image based in some mathematical equations and Bandwidth probing Algorithm.

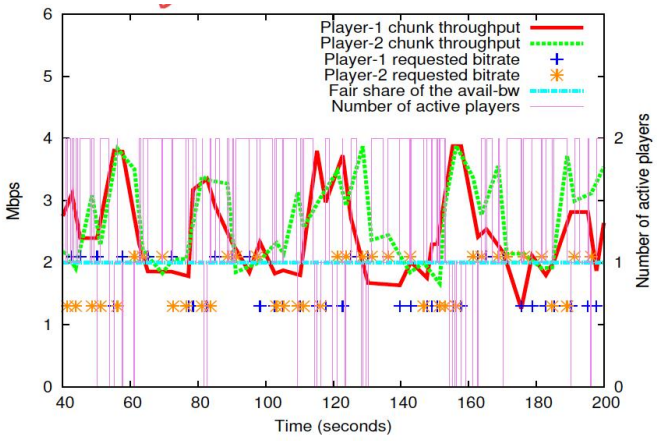

Figure 2- Simpler Player Output

Our Experimental Design

The goal of our experiment is to replicate the same result that is obtained in the reference paper and make a deduction based on the difference in the 2. The system used in our case is the TCP/IP network of 2 clients hosts, each hosting an adaptive streaming player connected to the same server host through the router, with a bandwidth of the bottleneck of the server to the router being 1.6Mbps.

The metrics that are to be obtained in our case is the same as the reference paper being the following:

- 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.
- The unfairness metric (for two players) is the average of the absolute bitrate differences between the corresponding chunks requested by each player.
- The utilization metric is defined as the aggregate throughput during an experiment divided by the available bandwidth in that experiment.

The algorithm that is used by the adaptive stream players is the same and the steady state buffer is set to 10 seconds in the experiment rather than the 8 seconds to speed up the experiment. The running average parameter is set to 0.8 on itself.

The workload used is a closed loop, where each player will send video packets whose resolution will be determined by the algorithm used in the player. All the players will always have packets to send. We also probed a little bit on the bandwidth algorithm to get our output.

There are no factor levels or factors in our case.

We used the Geni portals testbed to run the experiment. The following diagram shows our network architecture for the experiment.

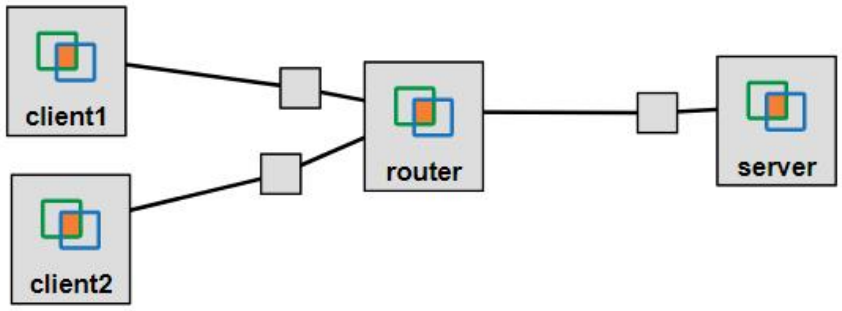

Figure 3: Network Design

Due to the change in the parameters, we were unable to get the exact picture as the one that is derived from the reference paper. we used another paper as the reference to run our experiment and it is cited as below,

- *P. Juluri, V. Tamarapalli and D. Medhi, "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," 2015 IEEE International Conference on Communication Workshop (ICCW), 2015, pp. 1765-1770, doi: 10.1109/ICCW.2015.7247436.*

We used the Dash Algorithm to get our output. The detailed steps of our experiment will be further explained in the ongoing pages.

Our results

The following figure is the figure that is our experiment objective.

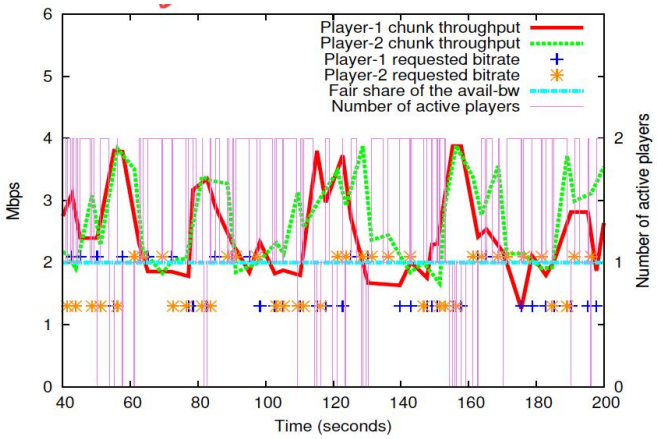

Figure 4: Required Graph

Despite the many difficulties faced during the entire project, we were able to obtain a figure that was like the above in general. The differences can be seen in some respects such as the oscillation time periods, irregularities and as such. We used the GENI testbed to set the router and the network topology.

Based on the experiment that is conducted we can conclude that the original experiment is robust, and it is very sensitive to the parameters such as the throughput, bandwidth and as such. A slight deviation from any of the stated parameters leads to a major difference in our experiment. Despite the sensitivity, we can still reproduce the result. The result we obtained is as follows:

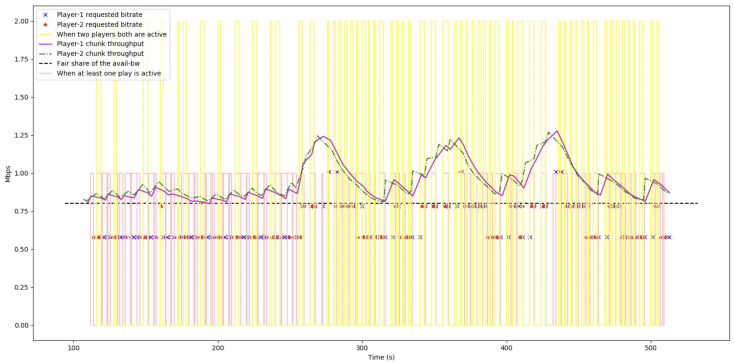

Figure 5: Obtained Graph

From the graph we can see that there is a difference in the bandwidth allocation form the start where it started late in our case but started early on in case of the original experiment. We also calculated and compared the utilization metric which came up to be 88.4% in our case but not in the case of the original paper where the same is 94%. From the original paper and in our case, we can see that in both the cases, we can notice that the as the number of players increase the players throughput also reduces. We can see that if only one player is active at a given moment, i.e., the On-Off of the 2 players are not

aligned, we can see one of the 2 players having a high throughput. However, the moment the other player also comes in play the throughput decreases for both the players.

Run the experiment

To run the testbed experiment, we first need to reverse four hosts in the Geni portal, one server, one router, and two clients.

On the server host:

install the Apache HTTP server:

```
sudo apt update

sudo apt install -y apache2
```

Then, download the video segments and put them in the web server directory:

```
wget https://nyu.box.com/shared/static/d6btpwf5lqmkqh53b52ynhmfthh2qtby.tgz -O media.tgz

sudo tar -v -xzf media.tgz -C /var/www/html/
```

The Big Buck Bunny DASH dataset is from:

Stefan Lederer, Christopher Müller, and Christian Timmerer. 2012. Dynamic adaptive streaming over HTTP dataset. In Proceedings of the 3rd Multimedia Systems Conference (MMSys '12). Association for Computing Machinery, New York, NY, USA, 89–94. DOI:https://doi.org/10.1145/2155555.2155570

Then, download the script to set a constant data rate of the output queue:

```
wget https://gist.githubusercontent.com/ffund/4a2b04f957a5f5bee206563f16717286/raw/7b88ee798f33905cbf912557816cd1deb252493c/rate-set.sh -O ~/rate-set.sh
```

And change the ip address in the script to the address of the interface of the server.

Then run:

```
bash rate-set.sh 1.6Mbit
```

Then install iftop to monitor the traffic:

```
sudo apt install iftop
```

On the client host:

Download the AStream DASH video client on the "client" node:

```
git clone https://github.com/pari685/AStream
```

Then install python2 to run:

```
sudo apt update

sudo apt install -y python2
```

Then modify the stream player script ~/AStream/dist/client/adaptation/basic_dash2.py to match with the rule in the reference paper:

Add global variable and constant:

```
download_rate = 0
DELTA = 0.8
```

Change download rate computation:

```
global download_rate
    download_rate = DELTA * download_rate + (1 - DELTA) * (recent_download_sizes[-1] * 8 / previous_segment_times[-1])
```

And change the bit rate incremental algorithm:

```
next_rate = bitrates[current_index + 1] if(download_rate > bitrates[current_index + 1] * config_dash.BASIC_UPPER_THRESHOLD) else current_bitrate
```

To run the experiment

On two clients host run:

```
python2 ~/AStream/dist/client/dash_client.py -m http://server/media/BigBuckBunny/4sec/BigBuckBunny_4s.mpd -p 'basic' -d
```

On server host run:

sudo iftop -nNbpt >> it.log

Finally, we have all the data needed to reproduce the figure.

The result of our experiment closely aligns the assumption from the paper that the overlapping ON period of the competing stream players causes the oscillation of the requested bit rate. It can be shown that, when only one player is active, it seems that it has the complete bandwidth and obtains a download rate higher than the fair share, then it tends to increase its requested bit rate. However, when it increases the bit rate, the ON period of this player is going to be longer, causing it to be overlapped with other one, then receiving a download rate close to its fair share, then decrease its requested bit rate again. Although it can be the case that the ON periods of the competing players are perfectly aligned thus do not have oscillation in bit rate, like the first hundred seconds in our figure, this situation cannot stay steady in the long term.