CS 655 - Computer Networks
Analyzing Transmission Policies for Adaptive Video

Name: Yuhang Song Email: yuhangs at at bu.edu BU ID: U********

GitHub Link: https://github.com/Souukou/bu_cs655_adaptive_video_analyzing

Demo Video: https://drive.google.com/drive/folders/10r44PrRKV3Nk8eDwrmBaaH_GHhUrcVId

If the video link expired/not work, you should be able to find a new one in GitHub repo. Instructions and scripts to reproduce the experiment are also in GitHub repo.

Part 1 Brief Description

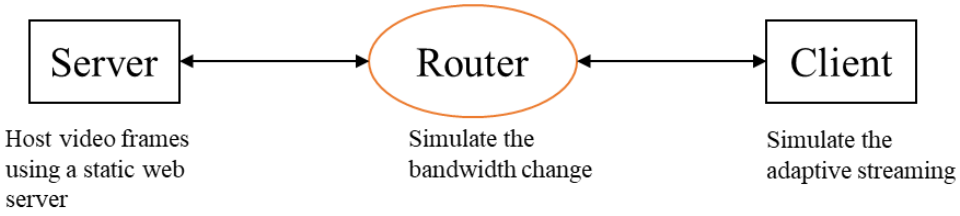
Introduction & Problem Definition

Adaptive video streaming is a technique to change the video quality dynamically to utilize the users' bandwidth. There are many algorithms to adjust the bitrate when streaming. In this project, we will explore three different streaming algorithms and compare their performance in different network situations.

Part 2 Experimental Methodology

Architecture

We have three nodes in this experiment, shown in the graph.



The server nodes run Apache service and host the video frames. We use tc command in the router server to control the bandwidth, simulate network outages, or simulate the mobile phone bandwidth change using a pre-recorded bandwidth record. And the client is responsible for simulating the adaptive streaming process. It will try to retrieve a high bitrate video when it finds the buffering speed is higher than playback speed. And if the buffering speed is lower than playback, it will try to retrieve lower bitrate frames.

We use the log file produced by the client to investigate the streaming quality. We analyzed the video quality using the metrics of average bitrate, interruptions, and variability of bitrate. We use the NYU Metropolitan Mobile Bandwidth Trace file to simulate the mobile bandwidth change. We assume that when users move around with a mobile, the main change of their network is the bandwidth.

Environment

All the experiments are done in the GENI lab, slice yuhang-final. GENI lab environment: 1 Core, 1G Memory, OS: Ubuntu 20.04 LTS.

To set up the experiment environment, we need to download and install some software or scripts on each node. To simplify the setup process, I wrote a shell script to automatically set up the environment.

I adapted the commands provided by the tutorial, but also made necessary change. The scripts can be found in my GitHub repo script/ directory. Also, I keep a copy of mobile bandwidth trace files in my repo, as well as the GENI description file, to avoid original file failure.

Part 3 Result

I wrote shell scripts for every experiment. The scripts can be found in my GitHub repo.

Constant Bitrate

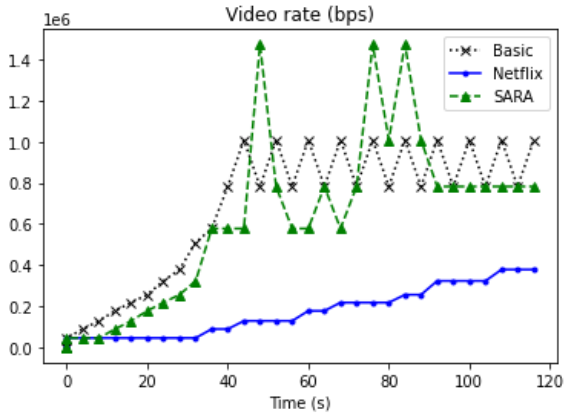
Router

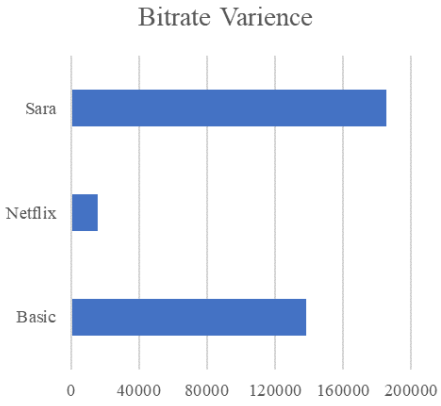
```
bash rate-set.sh 1000Kbit
```

Client

```
timeout 160 python2 ~/AStream/dist/client/dash_client.py -m
http://server/media/BigBuckBunny/4sec/BigBuckBunny_4s.mpd -p 'basic' -d
```

Bitrate Graph and Analysis





Analysis

This is the most ideal situation, where the bandwidth is constantly stable. We can observe that the basic streaming algorithm does best here. It quickly reached the top bitrate and continued keeping at a high bitrate with lower variance. The Netflix streaming algorithm does worst in this case. It is too conservative that the bitrate increased so slowly. And the SARA streaming algorithm in the middle. Though it reached the highest bitrate very soon, it cannot keep stable. The variance of the bitrate is very large. There are no interruptions in constant bitrate.

Then I merged the frames of basic streaming and reassembled them to a full video to check the real viewing.

```
cat BigBuckBunny_4s_init.mp4 $(ls -vx BigBuckBunny_*.m4s) > BigBuckBunny_tmp.mp4
ffmpeg -i BigBuckBunny_tmp.mp4 -c copy BigBuckBunny.mp4
```

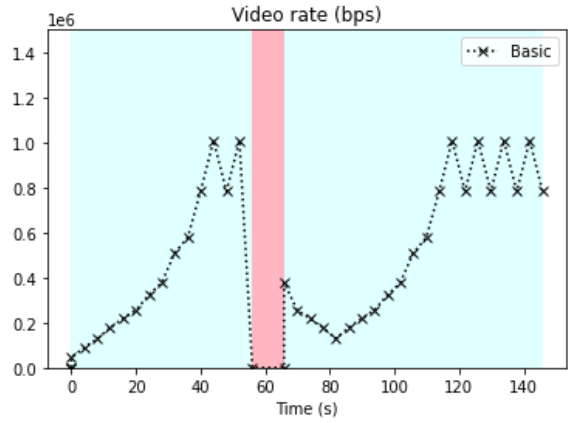
We can observe that in the first few seconds (shown in the left picture), the video is blurred. But it gradually gets clear (shown in the middle picture), as the client is trying to fetch high bitrate video. After it gets stable, although the bitrate varies, the view of the video does not have significant differences. (shown in the right picture)

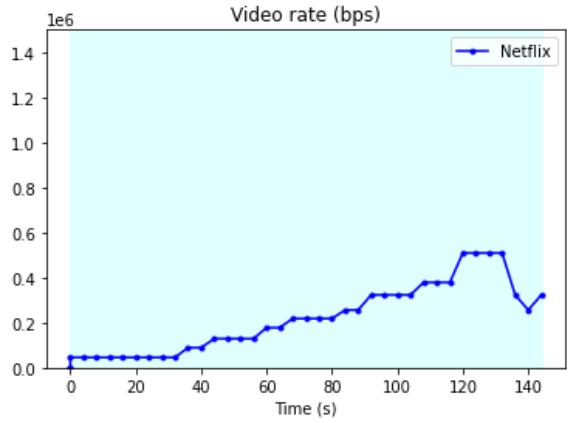
Constant Bitrate with Interruption

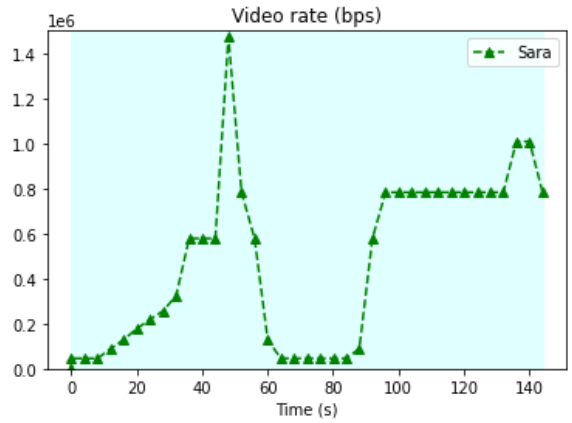
I change the router bandwidth to 50Kbps for 30 seconds. And I got the following graph.

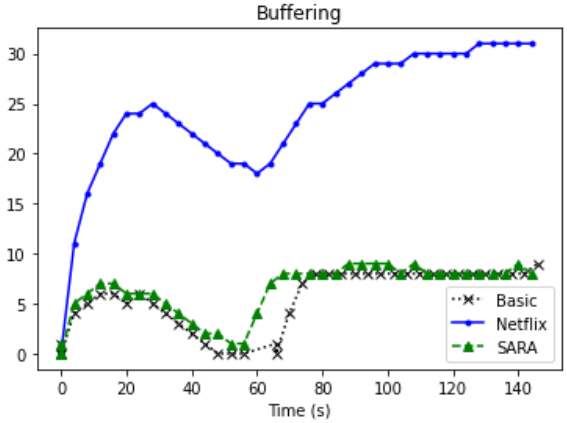
Bitrate Graph

	bitrate	variance	initial_beffering	interruption
Basic	445.7	121662.0	0.0054	9.8
Netflix	205.0	24611.0	0.0053	0.0
Sara	439.7	148940.1	0.0061	0.0







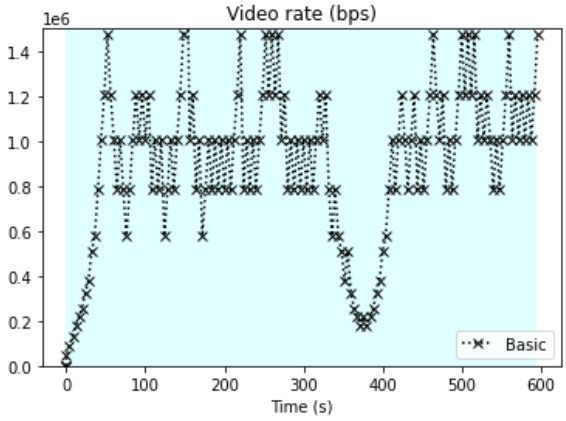


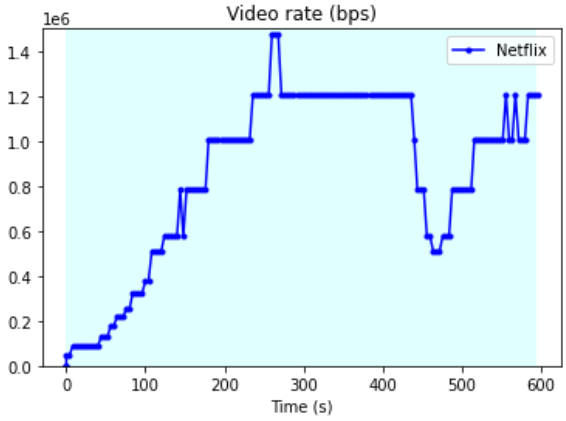
Analysis

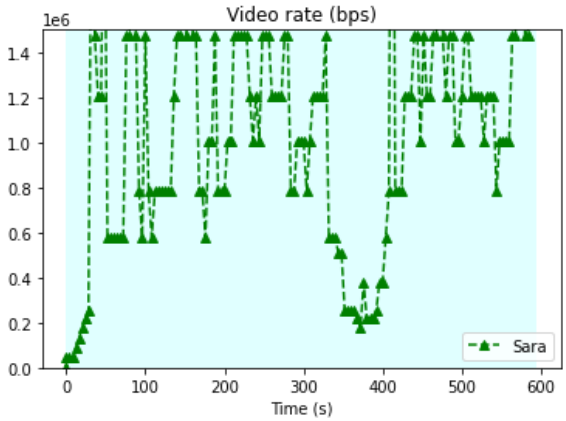
In this experiment, we limited the bandwidth to 50 Kbps for 30 seconds. During this period, all three clients have to use the buffered video frames for playback. And it is easy to observe that only the basic streaming algorithm encountered the interruption. That is because it has not buffered too many video frames. And we can see the Netflix algorithm is stable, and the bitrate of Netflix algorithm slowly increases. When encountered with network congestion, there is no interruption or significant change in the bitrate. This would be friendly to users, as the user won't fell the network congestion and only feel the video is getting clearer and clearer. As for Sara algorithm, it also buffered enough video frames to playback during the congestion, so there is no interruption either. However, there is still a significant bitrate change in the video.

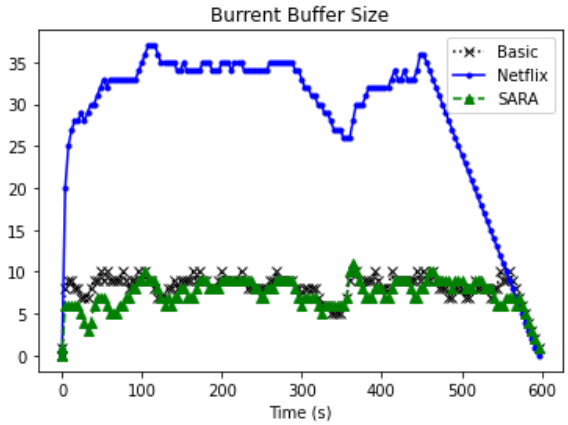
Mobile User with Stable Network (bus_62)

We use the bus_62 mobile bandwidth tracking file to simulate the use of mobile with a relatively stable connection.









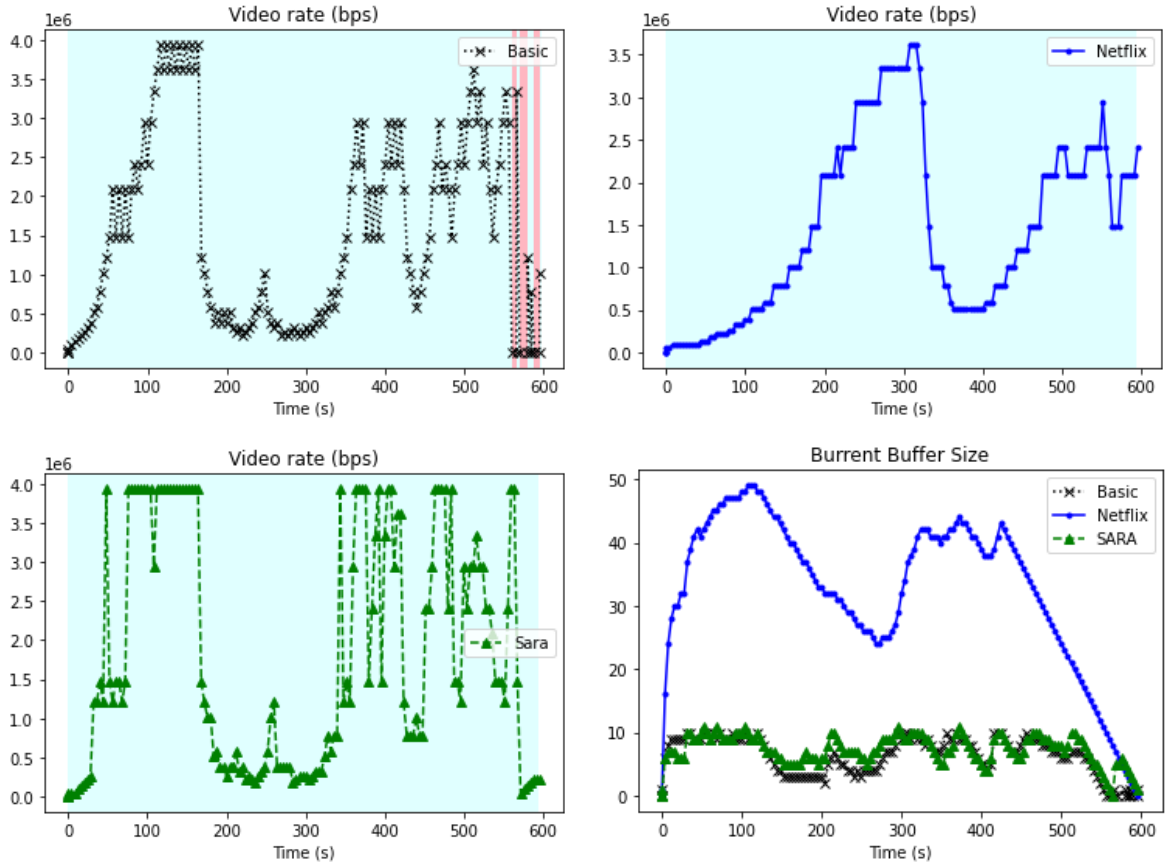
Analysis

We can observe that when mobile users have a stable network, the basic algorithm works well. It is very similar to constant bitrate. There are no interruptions, only some bitrate changes. The Netflix algorithm always prepared a lot of buffering video frames, but that won't be very useful when the network is stable. And the bitrate of Netflix is always lower than other two algorithms. The Sara algorithm always tries to reach the highest bitrate when possible. When those

three algorithms encountered short congestion, the Netflix algorithm is much more stable and has the lowest change in its bitrate.

Mobile User with Unstable Network (car_2)

We can use car_2 mobile bandwidth tracking file to simulate the unstable network conditions. We added a multiplier of 0.2 to make the network connection even worse. The network speed change rapidly and sometimes we get very good connections while sometimes we nearly get no connection. This is closer to the real situation we meet every day.



Analysis

Obviously, the Netflix algorithm is the most stable one in this situation. As it buffers a huge amount of video frame, it will not get interrupted when network congestion. This algorithm is very suitable for these kinds of weak-signal environments. And surely the Basic algorithm did the worst. It has 28 seconds of interruption. As the basic algorithm does not buffer enough frames, it will run out of the buffered frame when network is continuously weak. The Sara algorithm also looks better than Basic algorithm in this situation, as it at least did not get interrupted.

Part 4 Conclusion

From the above experiment, we can conclude that three adaptive streaming algorithms have different trade off between bitrate, interruptions, and variability of rate. Basic algorithm does not buffer many video frames, so it is more likely to get interrupted when the network change. If the network is ideally stable enough, like using fiber network, the basic algorithm brings the user the best experience with the lowest variance and high bitrate. In contrast, the Netflix algorithm buffers a lot of video frames, so it is more robust when encountering network congestion. It keeps a very stable bitrate that makes the user feel less about the network changes. It is more suitable for mobile users which the bandwidth changes rapidly. But the cost is bitrate. The Netflix algorithm always has the lowest bitrate amount of three algorithms. The Sara algorithm always tries to get the highest bitrate while keeping a little more buffer than Basic algorithm. But it has a high bitrate variance because it continues trying to reach a higher bitrate.

Part 5 Division of Labor

I finished the whole project myself.