Live Video Streaming Technology: a measurement study on Twitch.tv

CSC 466 Group Project

Chenkai Hao,

Zhaoxuan Xu,

Saige Liu

1.Introduction

Empowered by advanced devices that support high quality audio/video and improved Internet bandwidth for both WIFI and cellular technology, the demand of multimedia entertainment has been rapidly increased over the last few years, especially live video streaming applications. Such commercial systems as Twitch.tv and YouTube Live allows general users to broadcast their daily life and topics of interest to massive number of viewers.

This paper aims to present an overview of the technology evolution of live video streaming. To illustrate current technology for live video streaming, this paper presents a measurement study on Twitch.tv, which is one of the most popular live streaming video platform in the world.

From multiple aspects of user experience, including video quality, start-up latency, skipping frames, fps-frames per seconds), networking performance (packet drop rate, overhead ratio over data and packets level, retransmission rate, and data rate), to measure the overall online streaming performance. Considering multiple controlled variables is involving in the experimental, group control methodology is introduced and utilized during the whole measurement. For each comparison group, only one variable is measured and analyzed during the experiment period. The experimental data chart is represented in Section 4.2.

Based on result data, some networking mechanism in Twitch promotes the online streaming performance. Twitch uses scalable encoding technique, H.264 encoding, to encode high quality video streams into small chunks, which provides high video quality without sacrificing bit rate. Also, it can support multiple platforms, network environment and bitstreams. Additionally, Section 5.3 introduces a technique used in Twitch, Dynamic Adaptive Streaming over HTTP (DASH). The key sector of DASH, rate adaptation agent, is capable of selecting the resolution wisely by estimating previous bandwidth according to average throughput, and in order to decrease the streaming chunk loading latency while switching up/down resolution. It largely increases the Quality of Experience.

2. Research Methodology

In the early stage of this research project, assigned papers on the course reading list are reviewed to help the understanding of basic concepts of video streaming technology. After the topic was defined, UVic library online sources were used to search for academic journals and published articles from multiple verified databases like NCBI, Google Scholar, IEEE Xplore, etc. Keywords were used including “video streaming”, “live video streaming”, “measurement study”, “Twitch”, etc. Resulting articles were then filtered by years to obtain a list of up-to-date references.

3. Experimental Methodology

In this section, the basic experimental methodology is discussed along with the tools used in the test. A scientific control group is an essential part of many research designs, allowing researchers to minimize the effect of all variables except the independent variable. The control group, receiving no intervention, is used as a baseline to compare groups and assess the effect of that intervention. In our measurement design, with the concept of control experiments, we control that each trial only has one different controlled variable with another trial. The following chart shows the variables involved in our experiment. With limited amount of variable changes in our experiment, we minimize the effect of other factors, and focus on analyzing one variable at each time. Table 3.1 shows the comparable variables in each trial of our experiment.

Trial #	Controlled Group
1 vs. 2	Channel popularity (120K vs. 1K active viewers)
1 vs. 3	Video resolution (360p vs. 1080p)
3 vs. 5	Network scenario (campus network vs. residential network)
4 vs. 5	Operating system (Mac OS vs. Windows 10)
4 vs. 6	Connection (wired vs. wireless)
7 vs. 8	Video quality version (Auto vs. manually adjust)

Table 3.1 Experiment trials and controlled groups

During our experiment, Wireshark was used to monitor Internet traffics and IP locator was used to distinguish the traffic between the clients and Twitch server. Timer and calculator were also used to get estimated values for monitored data in comparison.

4. Experimental Details

With the measurement by using Wireshark, which gives us both direct and indirect data for analyzing. There are 9 groups of data, calculated or monitored for comparing results. In order to obtain the cleaning data which only involves the source and destination between our device's IP and twitch server's IP. But the filter formula also varies based on different demand on monitored data, the impurity data will be filter out in any way. It minimizes the result affected by another network traffic.

4.1 Monitored Data

4.1.1 Data Rate:

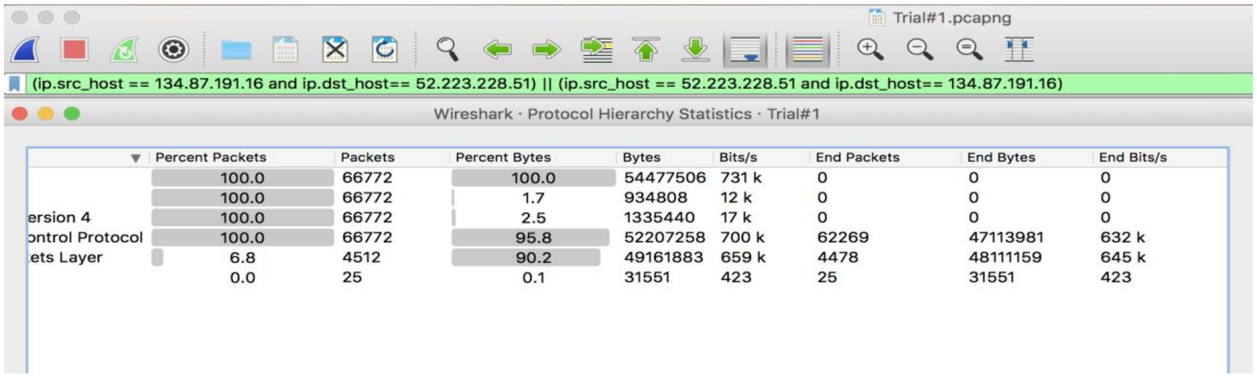
Direction: Bidirectional. The data throughput from our device IP to twitch is negligible, and to be consistent with other data measurement.

Calculation Method: average bit per second during whole test period, data is provided by Wireshark (See figure 1 below). Data rate gives us an insight of how data flow looks like in this trial.

Filter: (ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16).

```
(ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16)
```

Unit: bit/sec



```
(ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16)
```

	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
	100.0	66772	100.0	54477506	731 k	0	0	0
	100.0	66772	1.7	934808	12 k	0	0	0
ersion 4	100.0	66772	2.5	1335440	17 k	0	0	0
ontrol Protocol	100.0	66772	95.8	52207258	700 k	62269	47113981	632 k
ets Layer	6.8	4512	90.2	49161883	659 k	4478	48111159	645 k
	0.0	25	0.1	31551	423	25	31551	423

Figure 1: Example of Wireshark provided data

4.1.2 TCP Transmission Data:

Direction: Bidirectional.

Calculation Method: Provided by Wireshark under Statistics - Protocol Hierarchy. TCP Transmission Data provides us a total amount of data transmitted bidirectionally through our device and twitch channel server.

Filter: (ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16).

Unit: Megabytes (Mb)

4.1.3 Packet Drop Rate

Direction: Bidirectional

Calculation Method: # of lost packets divided by the # of total packets transmitted during test period. PDR could provide us the percentage of how often the packets got dropped and it can reflect the streaming performance in some way.

Filter: (ip.dst_host == 134.87.186.153 and ip.src_host == 52.223.227.247)||(ip.dst_host == 52.223.227.247 and ip.src_host == 134.87.186.153) and tcp.analysis.lost_segment

Unit: Percentage

4.1.4 Protocol Overhead Ratio (Packets)

Direction: Bidirectional

Calculation Method: # of non-video data packets / total video data packets. And it is filtered by frame.length greater or equal to 1460, which is the video data packet size commonly seen in the monitoring result.

Incurring excessive additional overheads would have negative impact on video streaming. The larger the segment size the smaller overhead it will have, our goal is to measure the relationship between overhead ratio and packet drop rate and the whole video performance.

Filter: (ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16) and frame.len>=1460.

Unit: Percentage

4.1.5 Protocol Overhead Ratio (Data)

Direction: Bidirectional

Calculation Method: non-video data size / video data size. Same filter is applied as the previous packets overhead ratio.(frame.length = 1460).

Filter: (ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16) and frame.len>=1460.

Unit: Percentage

4.1.6 Startup Latency

Startup latency is one of the most important indicators to the performance of video streaming applications. In most measurement studies we explored for this project, startup latency refers to the time interval from when the channel is selected until the video content actually starts playing on the screen. To obtain an accurate value of startup delay, the calculation requires the sufficient number of video segments that enables video playback. This number is predefined by video encoder and the streaming engine. However, with the monitored data on Wireshark, it is difficult to identify which data packets are those enable the video playback. Thus, a manual timer was chosen to provide an estimation of the startup delay (in seconds) of video playback.

4.1.7 Retransmission Packets and Data

Direction: Bidirectional

Calculation Method: Retransmitted data / Total data size. It indicates the amount of data being retransmitted and it might affect the synchronization of the video and sound.

Filter: (ip.src_host == 134.87.191.16 and ip.dst_host== 52.223.228.51) || (ip.src_host == 52.223.228.51 and ip.dst_host== 134.87.191.16) and tcp.analysis.retransmission.

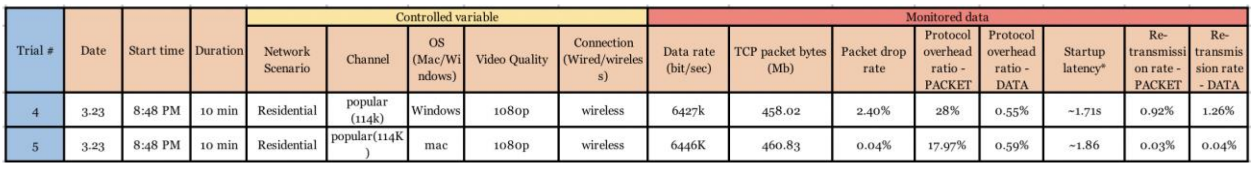
Unit: Percentage

4.2 Experiment results and analysis

This section provides the comparison between different trials described above, and each comparison uses control variable method - only one variable is analyzed during the test.

4.2.1 Operating System Comparison (Mac OS vs. Windows 10)

In our experiment, we use both Mac and Windows machines as shown in Table 4.2.1-1. In the comparison between trial 4 and 5, we monitored same channel with 114k viewers simultaneously. We started from the same time and last for same amount of time (10 min). The Video quality is both set to be 1080P, and the experiment scenario is both on campus. The only variable is Operating system/computer configuration.

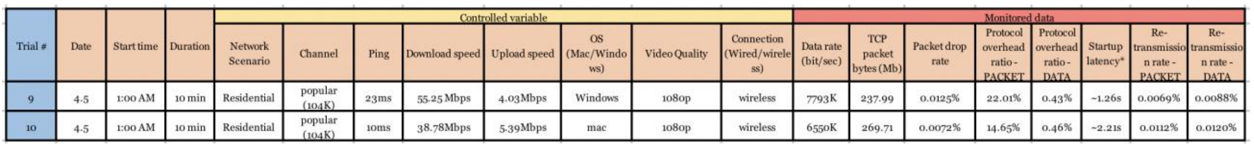


Trial #	Date	Start time	Duration	Controlled variable					Monitored data							
				Network Scenario	Channel	OS (Mac/Windows)	Video Quality	Connection (Wired/wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA	Startup latency*	Re-transmission rate - PACKET	Re-transmission rate - DATA
4	3.23	8:48 PM	10 min	Residential	popular (114k)	Windows	1080p	wireless	6427k	458.02	2.40%	28%	0.55%	~1.71s	0.92%	1.26%
5	3.23	8:48 PM	10 min	Residential	popular(114K)	mac	1080p	wireless	6446K	460.83	0.04%	17.97%	0.59%	~1.86	0.03%	0.04%

Table 4.2.1-1

From the data shown in Table 4.2.1-1, we could see that the TCP packet rate and data rate are almost the same. P4(Windows laptop)'s packet rate is 458.02 megabytes, with a data rate of 6427K bit/sec. P1(Mac laptop)'s packet rate is 460.83 Mb and its data rate is 6446K bit/sec. The differential between TCP packet rate is as few as 2.81 Mb. This result shows that in the same period, the transmission speed and total data transmitted stays the same. Also, the startup latency and Protocol overhead ratio is relatively the same. When looking at the data side, the overhead ratio is 0.55% and 0.59%, P1 is little bit larger than P4. But on the packet side, the windows laptop is having a lager overhead ratio. These data show that the operating system and different configuration will not affect the data transmission speed and the protocol overhead. However, Window pc's packet drop rate and re-transmission rate is much higher than the Mac laptop. The gap between their packet drop ratio is from 2.4% to 0.04% in this case.

There is no research or evidence show that Mac OS has a better performance than Windows. The high drop rate in the data could be a typical case. After investigating, we find out that the Windows laptop was using a proxy extension on the chrome browser when receiving data from twitch.tv. The proxy server might delay the round trip time of packets, thus the pc will experience high packet drop rate. Thus, another experiment using the same controlled variables is made to verify this assumption.



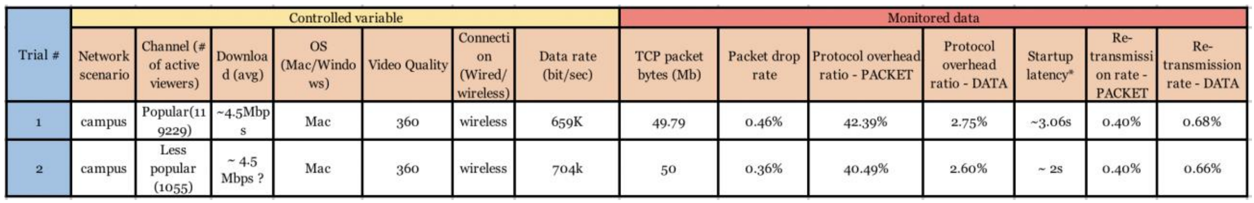
Trial #	Date	Start time	Duration	Controlled variable								Monitored data								
				Network Scenario	Channel	Ping	Download speed	Upload speed	OS (Mac/Windows)	Video Quality	Connection (Wired/wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA	Startup latency*	Re-transmission rate - PACKET	Re-transmission rate - DATA
9	4.5	1:00 AM	10 min	Residential	popular (104K)	23ms	55.25 Mbps	4.03Mbps	Windows	1080p	wireless	7793K	237.99	0.0125%	22.01%	0.43%	~1.26s	0.0069%	0.0088%
10	4.5	1:00 AM	10 min	Residential	popular (104K)	10ms	38.78Mbps	5.39Mbps	mac	1080p	wireless	6550K	269.71	0.0072%	14.65%	0.46%	~2.21s	0.0112%	0.0120%

Table 4.2.1-2

As shown in Table 4.2.1-2, the first line is Windows laptop and the second one is Mac laptop, all the other variables are as same as in Table 4.2.1-1. In this case, the packet drop rates are 0.0125% and 0.0072%, Windows pc is slightly bigger than the other, but the difference is too trivial that can be neglected. For the re-transmission rate, the calculated data can all be rounded to 0.01%. The experiment is done on the midnight, so the error rate is extremely small. The result shows that Operating system and computer configuration will not affect the network performance.

4.2.2 Channel Popularity (popular vs. less popular channel)

In the comparison of popular and less popular channels, our experiment used two Apple MacBook under wireless campus networking with an average downloading speed of 4.5 Mbps. Both popular channel with 120k active viewers and less popular channel with 1k active viewers are monitored at the same period for a 10 minutes traffic. Based on the observation, the protocol overhead of the popular channel is slightly than the less popular one in both packet amount level and total data size level, as shown in Table 4.2.2.



Trial #	Controlled variable							Monitored data						
	Network scenario	Channel (# of active viewers)	Download (avg)	OS (Mac/Windows)	Video Quality	Connection (Wired/ wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA	Startup latency*	Re-transmission rate - PACKET	Re-transmission rate - DATA
1	campus	Popular(119229)	~4.5Mbps	Mac	360	wireless	659K	49.79	0.46%	42.39%	2.75%	~3.06s	0.40%	0.68%
2	campus	Less popular (1055)	~ 4.5 Mbps ?	Mac	360	wireless	704k	50	0.36%	40.49%	2.60%	~ 2s	0.40%	0.66%

Table 4.2.2 Experiment comparison between different channel popularity

High protocol overhead ratio could be an indicator that the query node communicates with more peer nodes to exchange data availability information. From the network architecture perspective, a mesh-pull video streaming architecture could have been introduced into Twitch.tv as explained in the measurement study of a large-scale IPTV application by Hei et al. in 2007 [2]. Although Twitch might have different mechanism of peer recognition, this study explained a gossip-like protocol that provides a clear insight of the communication among peer nodes in both initial and balanced stages. A query node firstly obtains an initial list of peers from the tracker server and then extends the peer list by catching peer information from those initial nodes, as shown in Figure 4.2.2. With much more available active peers in the network, a query node intrinsically maintains a larger peer list with the limitation of its memory storage. Therefore, the communication generates more traffic in packet level to download required video chunks from a potentially larger choices of peer nodes.

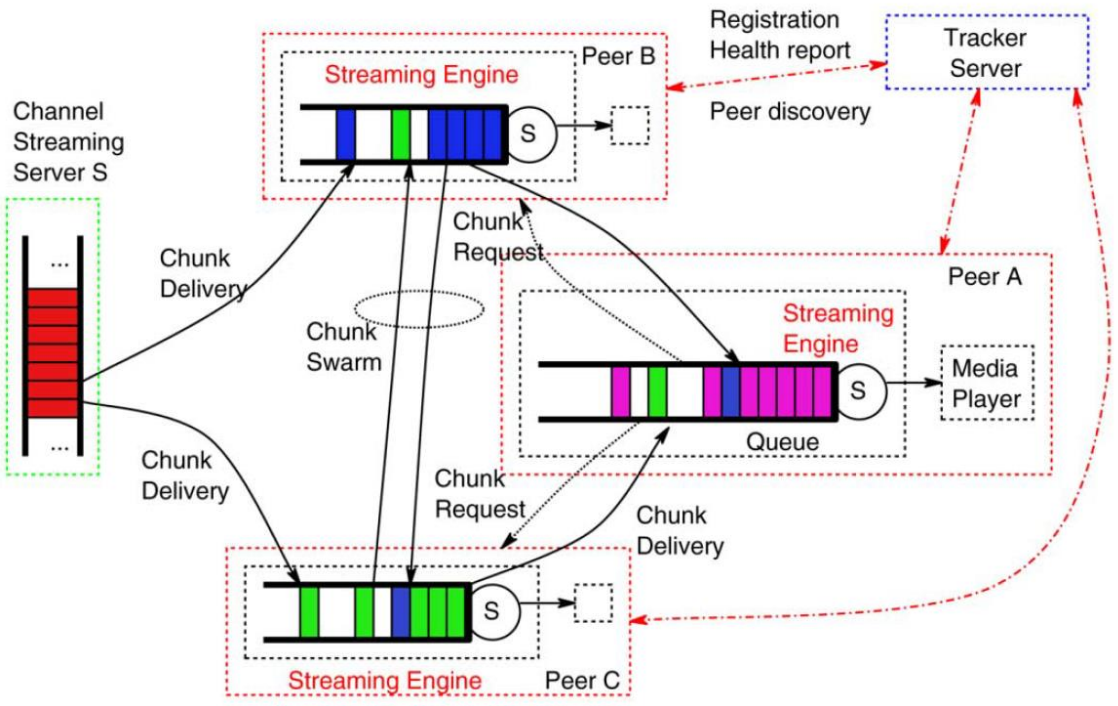


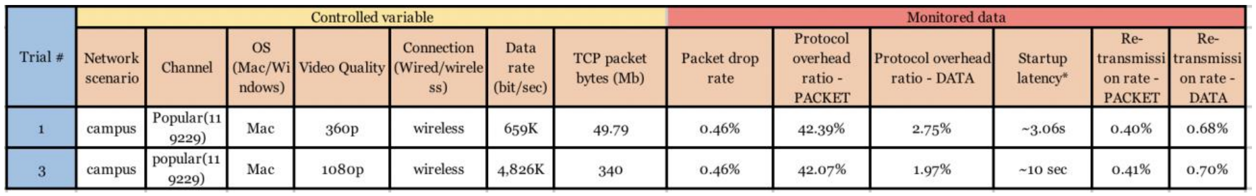
Figure 4.2.2 Peer-assisted video streaming model

It is also observed that the startup latency of the popular channel is longer than it is of the less popular one. One of the major reasons could be that the query node asks for larger amount of peer nodes to obtain the peer list in the communication establishment phase. We calculate the re-transmission rate for both packet level and data level as well. The result shows little difference due to that the network bandwidth overwhelms the required data rate of the streaming video, which also indicates that Twitch is capable of handling large amount of traffic for each channel with the limitation of network speed.

4.2.3 Video Quality (360p vs. 1080p)

Twitch sets the video quality to “Auto” by default so that the video quality could change over time to adapt to viewers’ network performance. Similar to the comparison of channel popularity in last section, we examined the live video streaming of the same channel with two different prefixed video qualities, 360p and 1080p. This procedure is done by selecting the video quality on the streaming page and then restart the video streaming so that the browser remembers users’ last choice.

The experimental results are shown in Table 4.2.3, which shows a significant difference of the estimated startup delay. The streaming with 1080p has an approximately 10 seconds of delay that is much longer than it is of 360p. Data rates (bit/sec) are 659K and 4,826K for the streaming of 360p and 1080p respectively. However, the down side indicators of networking performance including protocol overhead and retransmission rate are closed in a reasonable range. This could infer that better video quality requires higher data rate in transmission of video content. However, the streaming of a 1080p video in a popular channel with 120K active viewers does not seem to exceed the capacity of the campus network. Although the startup latency is not the same, the video fluency shows not significant difference among two streams.



Trial #	Controlled variable							Monitored data					
	Network scenario	Channel	OS (Mac/Windows)	Video Quality	Connection (Wired/wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA	Startup latency*	Re-transmission rate - PACKET	Re-transmission rate - DATA
1	campus	Popular(119229)	Mac	360p	wireless	659K	49.79	0.46%	42.39%	2.75%	~3.06s	0.40%	0.68%
3	campus	popular(119229)	Mac	1080p	wireless	4,826K	340	0.46%	42.07%	1.97%	~10 sec	0.41%	0.70%

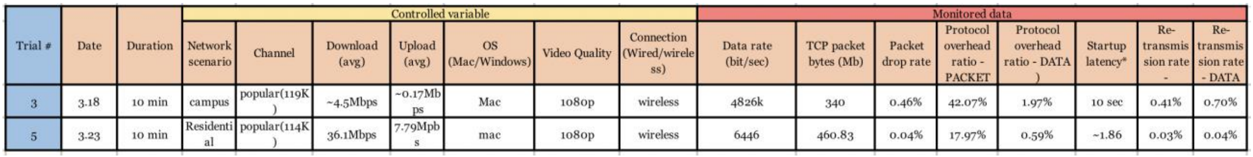
Table 4.2.3 Experiment comparison between different video resolutions (360p vs. 1080p)

According to Karine Pires et al. in 2014, the video is encoded at the broadcaster side in different resolutions and bitrates and then transmitted to Twitch data center [3]. This layered video encoding enables a tradeoff between video quality and playback fluency. Viewers with slow Internet speed could still maintain a relatively high video continuity by switching to a lower video resolution.

4.2.4 Network Scenario (campus vs. residential network)

Network bandwidth would be the first considerable element in a measurement of a network protocol. Trial #3 was measured on campus network scenario, which has much lower bandwidth limit compared to residential. (Uvic does not provide specific bandwidth number, it is modifiable in order to provide the best experience for all users.). The residential place we measured has the downlink data rate 75 Mbps and uplink data rate 7.5Mbps [4].

And in fact, during the measurement process, the average download rate for residential can reach around 36.1 Mbps and upload is ~7.8 Mbps. The actual average downlink and uplink data rate on campus are 4.5Mbps and 0.17 Mbps. So, it is a significant bandwidth difference between two trials.

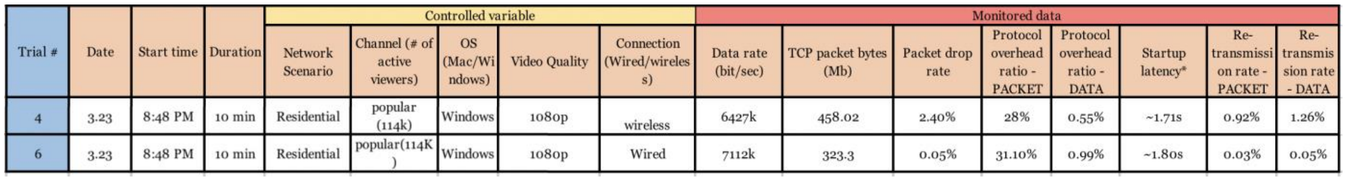


			Controlled variable							Monitored data							
Trial #	Date	Duration	Network scenario	Channel	Download (avg)	Upload (avg)	OS (Mac/Windows)	Video Quality	Connection (Wired/wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA)	Startup latency*	Re-transmission rate -	Re-transmission rate - DATA
3	3.18	10 min	campus	popular(119K)	~4.5Mbps	~0.17Mbps	Mac	1080p	wireless	4826k	340	0.46%	42.07%	1.97%	10 sec	0.41%	0.70%
5	3.23	10 min	Residential	popular(114K)	36.1Mbps	7.79Mbps	mac	1080p	wireless	6446	460.83	0.04%	17.97%	0.59%	~1.86	0.03%	0.04%

Table 4.2.4 Experiment comparison between different network scenarios (campus vs. residential network)

Due to the network difference, the data rate does not show much difference, residential has 6.4 Mbps, while campus has 4.5 Mbps. This data could also reflect another monitored data, TCP transmission packets' bytes. Besides, according to the packet drop rate shown in Table 4.2.4, campus network has 0.46% drop rate, while the better networking scenario has nearly 0.04% drop rate, almost neglectable. Additionally, the rest of our monitored data, Protocol Overhead Ratio (Packet), they have 42.07% and 17.97% accordingly, and 1.97% vs. 0.59% in overhead data ratio. And it has bigger gap in startup latency, one has ~10s, and residential has ~1.86s. This data indicates the quality of experience, how users experience varies based on different networking scenario. Retransmission packets has similar functionality as the drop rate, residential network still has the overwhelming advantages than campus scenario. Thus, boosted bandwidth allows for quicker uploads and better streaming quality.

4.2.5 Connection (Wired vs. wireless)

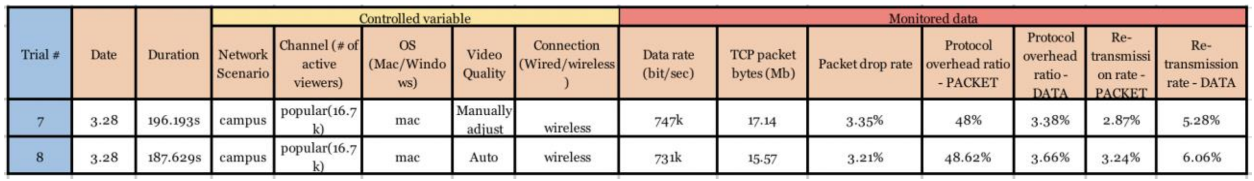


Trial #	Date	Start time	Duration	Controlled variable					Monitored data							
				Network Scenario	Channel (# of active viewers)	OS (Mac/Windows)	Video Quality	Connection (Wired/wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA	Startup latency*	Re-transmission rate - PACKET	Re-transmission rate - DATA
4	3.23	8:48 PM	10 min	Residential	popular (114k)	Windows	1080p	wireless	6427k	458.02	2.40%	28%	0.55%	~1.71s	0.92%	1.26%
6	3.23	8:48 PM	10 min	Residential	popular(114K)	Windows	1080p	Wired	7112k	323.3	0.05%	31.10%	0.99%	~1.80s	0.03%	0.05%

Table 4.2.5 Connection method comparison

As shown in Table 4.2.5, the two trials tested identical channel started at the same time. They have the same duration, network scenario, video quality and operating system. The only difference is the connecting method. Trial #4 is using the wireless fidelity and trial #5 is using the wired connection. Obviously, the wired connection has a better data rate. The wireless connection has a 6427K bitrate, the wired one is 685K bitrate faster than the wireless one. The wireless connection is suffering a huge packet drop rate. The TCP packet drop rate is 2.40% and it’s 48 times the wired connection drop rate, which is a massive difference. Besides, the retransmission rate of wireless connection is much higher than the wired one. The wireless connection ’s retransmission rate is 0.92%, almost 30 times the retransmission rate of the wired one. These huge differences explain why wireless connect has some higher total packet bytes than the wired connection, the wireless connection has more missing packets that needs to re-transmitted again, thus during the same period of time, it will transmit more packet bytes. However, for the protocol overhead ratio, the wired connection has a higher overhead. This is probably because the wireless connection takes more resources to establish the connection. Overall, wired connection is much superior than the wireless connection, it has less error rate and faster transmission speed.

4.2.6 Streaming Resolution Adjustment (Manual vs Auto)



Trial #	Date	Duration	Controlled variable					Monitored data						
			Network Scenario	Channel (# of active viewers)	OS (Mac/Windows)	Video Quality	Connection (Wired/wireless)	Data rate (bit/sec)	TCP packet bytes (Mb)	Packet drop rate	Protocol overhead ratio - PACKET	Protocol overhead ratio - DATA	Re-transmission rate - PACKET	Re-transmission rate - DATA
7	3.28	196.193s	campus	popular(16.7k)	mac	Manually adjust	wireless	747k	17.14	3.35%	48%	3.38%	2.87%	5.28%
8	3.28	187.629s	campus	popular(16.7k)	mac	Auto	wireless	731k	15.57	3.21%	48.62%	3.66%	3.24%	6.06%

Table 4.2.6 Resolution adjustment method comparison

From the research paper [5] by C. Zhang and J. Liu in 2015, they measured Twitch.tv and in the future research section, they mentioned an idea that the player devices could predict the network behavior and select the best suitable video quality. Low bandwidth cannot satisfy both the video smoothness and low latency. If the user realizes the discontinuous network and adjust the high-latency video quality to the low-latency one, there will a period of live streaming missing because of the high switch latency.

Nowadays, 3 years after the publication, we assume that the auto function is helping viewer using the best video quality automatically. We designed two trials to test this issue.

The first one is manual selection, the other one is using the auto function and let the software decide the perfect resolution. When looking at the data, the manual selection one has a higher data rate than the auto, but the difference is only 16K bitrate. The manual selection has some bigger packet bytes since the duration of it is longer than auto. The packet drop rate of them are close. Even through, the auto has a bigger protocol overhead ratio and a bigger retransmission rate, the difference is still too small that can be neglected.

From the experiment data point of view, there two video quality adjustment methods have no differences. However, the overall QoE differs due to the number of video freeze on the manual adjustment while the auto adjustment does not freeze.

5. Results and Discussion

5.1 How Twitch enhances scalability

Based on our experiment result in trial 1 and 2 with the comparison between two different channel popularities, the networking performance shows no obvious difference between two channels with different number of active viewers. However, a measurement study in 2007 suggests that the scalability of a live video streaming application is dependent to the network architecture [2]. In this study, Hei et al. performs a measurement on a large scale P2P IPTV system and concludes that peer to peer architecture plays an important role in extending the network scalability and streaming performance. With the comparison of PPLive, we found both live video streaming platforms are scalable and supports large amount of concurrent active viewers (in millions). We assume Twitch also adapts a P2P like networking architecture to support event driven live streaming when a great number of viewers enter the channel in a short period of time. The network load hits the peak but the streaming quality and fluency have to be held during the event. In a peer-to-peer like networking architecture, a newly inserting node (a client) enters the channel by initiating itself to the streaming server and querying for a peer list from tracking server to start, during which the new node determines the sources of required video chunks. The scheduling schemes varies among different applications. Twitch, however, may have some complicated scheduling schemes because simple schemes like push-based, pull-based, rarity based and deadline based may not be sufficient to fulfill the demand of P2P network [6].

In addition, how peers exchange data availability information also has a great impact on the overall performance. One of the advanced information exchange schemes is reviewed in this paper, a gossip-like communication method. Each peer broadcasts its current peer list to all the neighbors and also updates the tracking server about the network peers in a certain amount of time. This time could be prefixed or adjust during the streaming process. Although Twitch may have better resolution for data availability information exchange, peers in the network always keep themselves updated about all the other available information of nodes and video chunks [7][8].

5.2 Scalable Video Coding (SVC)

On the Twitch website, it is said that the streamers need to use the H.264 AVC. it suggests the user to use broadcast software that can provide h.264 encoding. H.264 encoding is a video compression standard that provide motion compensation and it's the most common used one around the world. The advantage of h.264 is that it can provide high video quality at a comparably lower bitrate than most of other standards. Also, that it supports flexibility on different platform, network and servers.

Scalable Video coding is an extension of H.264. It encoded a high-quality video bitstream into subset bitstreams. The high quality bitstream drops packets to get the subset bitstreams. The subset is usually with a smaller screen, a lower frame rate, or a lower quality. These features give SVC Temporal scalability (time scalability), Spatial(image) scalability and SNR(Quality) scalability. The video might skip few seconds frames, or lower the resolution or even lower the video quality to fit different situation.

We assume SVC is the technology behind Twitch. On the twitch video player, you can see different resolution. On the content server side, Twitch encode the video into different layers, each layer has different resolution and video quality. During the transmission, user choose one resolution and request other peers for the required video chunk of that resolution. So that this video can support various bandwidth and situation. Furthermore, the content server provide different layers is the fundamental for DASH.

5.3 How Twitch dynamically adapts video quality (DASH)

Twitch allow gamers to broadcast their gameplay over the Internet with a various of video resolution and bitrate. Motivated by the comparison between trial #7 and #8 shown in Table 5.3, the “Auto” function on Twitch live streaming plays a significant role for adaptive bitrate streaming which reduces the delivery bandwidth cost and enhances QoE of viewers.

Dynamic Adaptation works in Twitch by breaking the content into a sequence of small HTTP-based video segments, each segment containing a short interval of playback time of content, such as a live broadcast. The content is made of a variety of different bit rates. While the content is being played back by an DASH client, the scheduler or rate adaptation agent automatically selects from the alternatives the next segment to download and play based on current network conditions. The client automatically selects the segment with the highest bitrate possible that can be downloaded in time for playback without causing stalls or obvious latency in the playback [3]. See Figure 3 below.

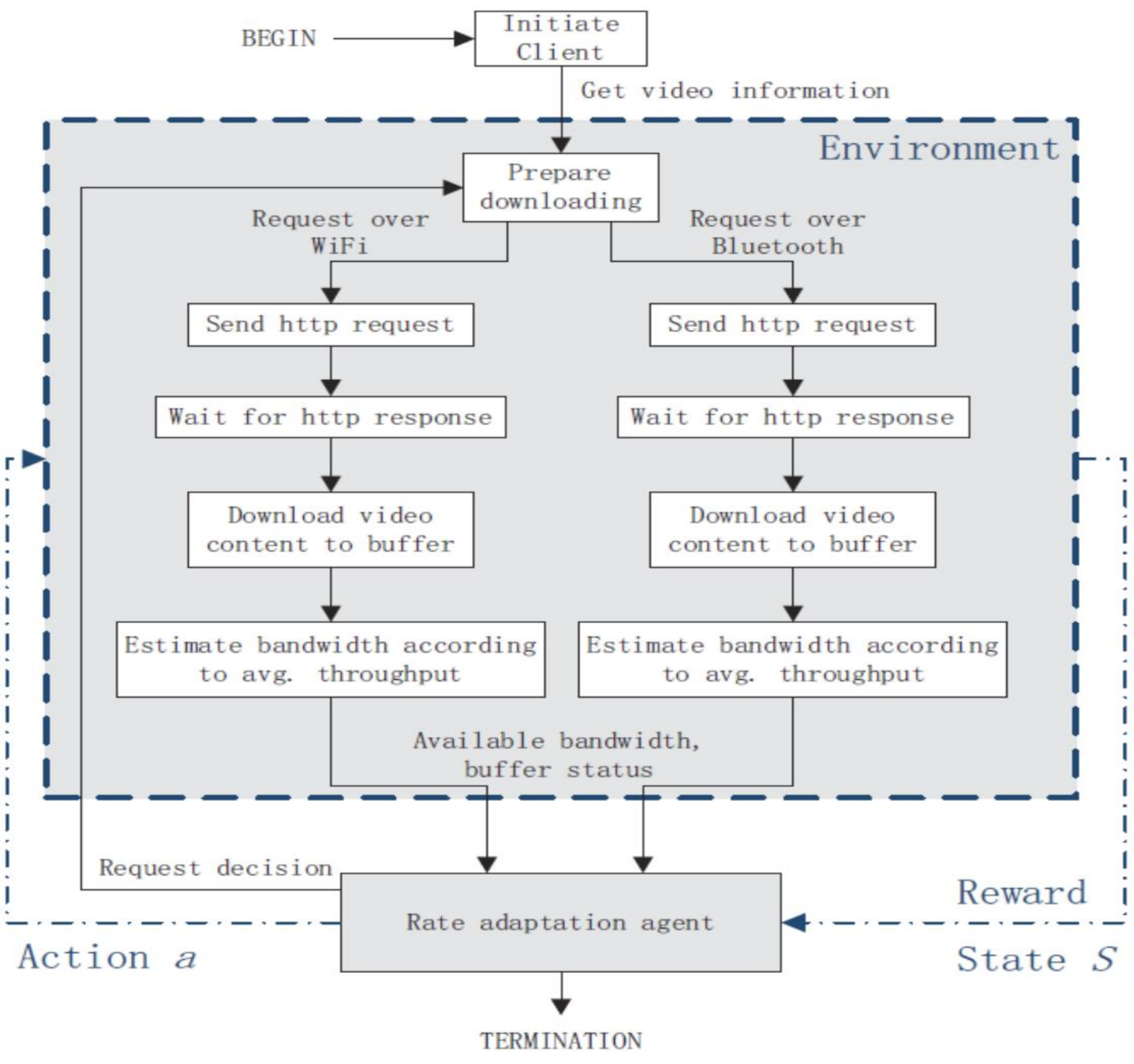


Figure 3. Dynamic adjustment model of video quality version

In the illustrated figure above, the algorithm of rate adaptation agent/scheduler auto-selecting segment by evaluating the current bandwidth based on previous transmitted data rate/bit rate, and provides the feedback to the environment in order to optimize the video quality performance.

Although the result of Wireshark does not demonstrate any distinguishable advantages on packets loss or retransmission rate when comparing the manual and automated switching streaming resolution in same network environment. However, there is such an apparent QoE difference when monitoring the result. No matter considering QoE or the resultant from Wireshark, which indicates the packets and data lost/retransmission rate is similar with the one has DASH algorithm (allow system to auto select the most efficient channel), allowing rate adaptation selection would largely solve the frame skipping and switching latency problem, which significantly increases streaming performance.

6. Conclusion

We can summarize the experiment results into the following conclusion. Operating system has trivial effect on the network performance. The wired connection could provide a much better network performance than the wireless network, including faster data transmission rate and less error rate. From the experiment data point of view, auto and manual quality adjustment methods have no obvious differences in Wireshark level, but it has much better QoE due to the existence of dynamic adaptation. Network scenario, which is closely related to bandwidth limits, would have a severe effect on both monitored data and QoE level. Because the poorer network environment has uplink and downlink boundary which limits the data transmission rate. Due to the limitation on experiment time period, there are difficulties to capture more than 1 million active viewers in one channel simultaneously, which is hard to cross the bottleneck of the Twitch channel capacity. Unlike networking scenario, different popularity in channels does not indicate distinguishable result. In brief, under 1 million active viewers in one channel would not have conspicuous differences.

The experiment proposed in this paper creates a direct visualization of the live video streaming performance on Twitch. However, it does not provide an insights of Twitch server due to the limited access. Future works could be considered such as monitoring an event driven channel, accessing streaming information through Twitch APIs, and comparing with other streaming applications that are known as non-P2P architecture.

7. Team Members Contribution

Saige Liu

As one of the research team, Saige contributed in research on the technologies behind live video streaming, did experiments using network performance monitoring tool, engaged in group discussion and devoted in presentation and research report. She Maintained and updated the project website, and proposed the idea to compare the auto function with manually selection, thus team could measure the auto function and study the technology behind it.

Zhaoxuan Xu

During the project process, Zhaoxuan made a positive contribution to group discussion, report writing, topic research, experiment design, data collection and analysis. He introduced the control variable experimental methodology and optimized the data chart with other team members. He did several research readings regarding to DASH and other streaming techniques and brainstormed with team members in every meeting to improve experiment integrity.

Chenkai Hao

As a group member, Chenkai lead in the project topic refinement and research process. He contributed to the understanding of the concept of video streaming technology and explained in plain language during several group meetings. To obtain a better experiment result, Chenkai created the data entry table for experiment trials and optimized the calculation formula together with the other group members. He actively participant in every group meetings as well as the report writing and formatting.

8. References

1. Smith, C. (2018, April 06). 46 Amazing Twitch Stats and Facts. Retrieved April 12, 2018, from https://expandedramblings.com/index.php/twitch-stats/
2. Hei, X., Liang, C., Liang, J., Liu, Y. and Ross, K. (2007). A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9(8), pp.1672-1687.
3. Pires, K., & Simon, G. (2014). DASH in Twitch. *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming - VideoNext 14*. doi:10.1145/2676652.2676657
4. Pande, G. (2013). Metrics for Video Quality Assessment in Mobile Scenarios. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1304/1304.3758.pdf [Accessed 12 Apr. 2018].
5. Zhang, C., & Liu, J. (2015). On crowdsourced interactive live streaming. *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video - NOSSDAV 15*. doi:10.1145/2736084.2736091

6. Pal, K., Govil, M. C., & Ahmed, M. (2018). Priority-based scheduling scheme for live video streaming in peer-to-peer network. *Multimedia Tools and Applications*. doi:10.1007/s11042-018-5741-y
7. Zou, S., Wang, Q., Ge, J., & Tian, Y. (2018). Peer-Assisted Video Streaming With RTMFP Flash Player: A Measurement Study on PPTV. *IEEE Transactions on Circuits and Systems for Video Technology, 28*(1), 158-170. doi:10.1109/tcsvt.2016.2601962
8. Xing, M., Xiang, S., & Cai, L. (2014). A Real-Time Adaptive Algorithm for Video Streaming over Multiple Wireless Access Networks. *IEEE Journal on Selected Areas in Communications, 32*(4), 795-805. doi:10.1109/jsac.2014.140411