Fall 2012

VIDEO IN THE CLOUD TCP CONGESTION CONTROL OPTIMIZATION FOR CLOUD COMPUTING

Rafael Alvarez-Horine
San Jose State University

VIDEO IN THE CLOUD

TCP CONGESTION CONTROL OPTIMIZATION FOR CLOUD COMPUTING

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Rafael Alvarez-Horine

November 2012

VIDEO IN THE CLOUD

TCP CONGESTION CONTROL OPTIMIZATION FOR CLOUD COMPUTING

by

Rafael Alvarez-Horine

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

November 2012

Dr. Melody Moh	Department of Computer Science
Dr. Sami Khuri	Department of Computer Science
Dr. Chris Pollett	Department of Computer Science

ABSTRACT

VIDEO IN THE CLOUD

TCP CONGESTION CONTROL OPTIMIZATION FOR CLOUD COMPUTING

by Rafael Alvarez-Horine

With the popularity of video streaming, a new type of media player has been created called the adaptive video player that adjusts video quality based on available network bandwidth. Merging this technology with cloud computing will change the online video landscape by allowing providers to dynamically create media servers that take advantage of all the benefits of cloud computing.

This however is not a straightforward endeavor as unlike a traditional data center; a cloud-based infrastructure is subject to a greater amount of performance variability. While the adaptive video player is designed to cope with variability in general, a video server in the cloud will be less optimal compared to one running on dedicated hardware.

In this paper, we research maximizing the video streaming experience in the cloud from the adaptive video server perspective through TCP congestion control algorithms. Five major TCP congestion control variants are evaluated: Cubic, Bic, Vegas, H-TCP, and HighSpeed TCP. Additionally both private and public cloud environments are tested with the final evaluation based on video streaming performance as well as TCP friendliness.

ACKNOWLEDGEMENTS

I would like to thank Dr. Melody Moh for her kindness, encouragement, and above all belief in me that made this project possible.

I would also like to thank the San José State Department of Computer Science for all their support specifically, Dr. Khuri, Dr. Pollett, Dr. Chun, Dr. Araya, and Dr. Pearce for giving me a chance to work with some of the most dedicated educators I've had the privilege of knowing.

Above all, I would like to thank my wife Sarah for her ongoing support and patience throughout my academic journey.

TABLE OF CONTENTS

CHAPTER

LIST OF ACRONYMS

AVG	Average
AIMD	Additive Increase Multiplicative Decrease
AZ	(Amazon) Availability Zone
BDP	Bandwidth Delay Product
CDN	Content Delivery Network
EC2	(Amazon) Elastic Compute Cluster
IaaS	Infrastructure-as-a-Service
LFN	Long Fat Network
OSMF	Open Source Media Framework
RTSP	Real Time Streaming Protocol
SD	Standard Deviation
VM	Virtual Machine

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1. INTRODUCTION

The proliferation of video streaming on the Internet has resulted in a concerted effort to find the fastest, cheapest, and most reliable way to push video from a server to a media player. With analysts predicting that by 2014 upwards of 66% of mobile traffic will be streaming video [1], this is currently a popular area of research both in academia as well as industry. Increased video demand means additional computing resources will be needed to store and serve video online. With the unique challenges inherent in streaming media content, we propose the creation of a streaming video server that is optimized to run in a cloud-computing environment.

Moving video servers to a cloud computing infrastructure would realize numerous advantages specific to the cloud, features such as automated server scaling for viral videos, instantaneous global presence for international viewers, and built in redundancy for site availability.

While this may appear to be an ideal pairing, a cloud-based video server is also subject to the limitations of cloud computing, chief among them being the instability inherent in a public cloud. This is due, among other things, to the shared tenancy effect whereby every action a cloud user takes can impact other clients in the same segment of the cloud. Streaming video is especially sensitive to changes in resource availability, where a problematic network can result in unwatchable videos.

To work around these inconsistencies, we combined our proposed cloud-based video server with the adaptive video player. As the name implies, the adaptive video player dynamically adjusts how it plays a video according to the prevailing network conditions allowing for a customized viewing experience. This technology has become more popular recently with large video providers like Netflix [2] and Hulu [3] integrating it into their media players.

However this solution is suboptimal as the adaptive video player alone may not provide the best viewing experience when streaming from a cloud-based server. To that end we propose a further optimization via an analysis of TCP congestion control algorithms on adaptive video streaming in the cloud. While the adaptive video player seeks to bypass TCP congestion control altogether and provide a fully realized solution to network congestion, we believe a combination of the two technologies creates the optimal solution for streaming videos.

2. BACKGROUND

2.1 Video Streaming over the Internet

Historically streaming video over the Internet meant clicking a link and waiting for the video to start downloading until the local cache was full. The video would start playing and during playback the rest of the video would be downloaded in the background by the player. As long as the video bitrate did not exceed available bandwidth, the video would reliably play.

If available bandwidth changed suddenly, the end user experience would suffer as the video performance degraded. Problems such as stuttering video (video that stops and starts suddenly), dropped frames (lost portions of video) or video that stops playing altogether are familiar to longtime Internet users. Several techniques have been used by video sites to address these issues. One common one is providing lower quality videos that do not require as much bandwidth and are more likely to play reliably when available bandwidth is low. While practical, the resulting experience watching the video is poor with blurry video and hard to understand sound. While this may be acceptable for a short clip, it is not desirable for watching an entire television show or movie. Content Distribution Networks (CDNs) are also employed by sites to host multiple copies of videos closer to the client. This allows the end user to stream content from a server that is closer geographically which results in fewer network hops (minimizing the chance of congestion) and a shorter network delay. While this is a good best practice, it also adds an

additional layer of complexity to the environment and does not address potential last mile network fluctuations between the CDN and the end user.

2.2 Cloud Computing

The use of a cloud-based service for video streaming would seem to be a good fit as the dependency on network bandwidth can be addressed by the cloud. A computing cloud is by definition a pool of unlimited resources, which can be dynamically scaled up and down to meet computing demand. Assuming that underlying network deficiencies can be ameliorated by adding additional computing resources, a cloud based video solution would make sense.

The cloud however introduces its own set of challenges. While a cloud is theoretically an unlimited computing resource, its performance is inherently not as reliable compared to dedicated hardware. Because a cloud by definition is a shared resource in which the various cloud tenants can impact each other, a cloud-based video service must be architected to create a reliable service out of unreliable components. While some video sites have opted to go this route, the last mile question is still not addressed as unlimited bandwidth will not alleviate a bad network connection between the user's device and their network provider.

2.3 TCP Congestion Control

The TCP protocol is responsible for ensuring reliable host-to-host communication irrespective of the media being transmitted. One of the features of TCP is congestion

control which serves to limit the number of packets transmitted on the network in response to the perceived amount of congestion.

The theory behind TCP congestion control is that if there is a large amount of network traffic sufficient to cause degradation in overall network performance, then the TCP host should send fewer packets while the network is compromised to allow it to stabilize. This is implemented via the additive-increase/multiplicative-decrease (AIMD) algorithm which dictates that data transmission rates should increase at a linear rate but decrease at a geometric one. AIMD controls the size of the TCP congestion window that dictates how much data can be sent at a time with a large congestion window resulting in more data being sent. This means that it takes a relatively long time to increase the amount of network traffic sent, but a short time to decrease it. Ideally once the network has returned to functioning normally, the amount of data transmitted can be ramped up to make optimal use of network resources. This is generally a reasonable course of action to take in most cases. However there are several scenarios where this will result in a suboptimal experience for video streaming.

A network event that causes a disruption in available bandwidth will result in a decrease in the amount of data that a TCP host will send via a reduction in the size of the congestion window. This reduction in data will continue until there have been several successful data transmissions after which the congestion window size will slowly increase. This behavior results in the familiar saw tooth pattern for TCP congestion window size as seen in Figure 1.

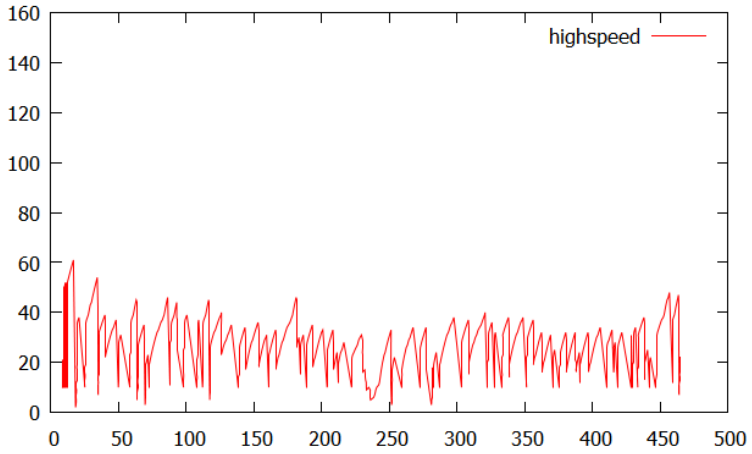

Figure 1: Sample Congestion Window Size fluctuation during an active session.

A relatively small number of network hiccups can play havoc with the congestion window size, which will result in less available bandwidth and a degraded end user experience. If the amount of bandwidth reaches a critical point, the video will begin to stutter as the cache is depleted. Once the cache is empty, the player will then stop altogether. The adaptive video player works around this limitation by independently monitoring available bandwidth and making decisions on what size of data to request in order to make sure that the bitrate of the video being played never exceeds available bandwidth. Ideally, the adaptive video player can ensure that the video bitrate being played is always within the available bandwidth so the end user maintains a continuous video stream with the assumption that the end user may occasionally see low quality video if there is a sudden drop in bandwidth.

3. RELATED WORKS

Using the TCP/IP protocol layers as an analogy, the adaptive video player provides an application layer workaround for the limitations of the network layer. This is not a new concept as other optimizations at this level have been proposed such as using multiple TCP streams [4] [6]. Multiple TCP streams provide additional pathways for video data to come through to the client without requiring additional network configuration, which is effective for artificially increasing bandwidth. This strategy however increases the chances of poor TCP-fairness with respect to other network traffic and runs the risk of saturating the network with traffic, effectively cancelling out the benefits of congestion control. In addition, limitations inherent in TCP such as send buffer size cannot be worked around easily from the application layer and are more efficiently dealt with at the network layer [7].

Several new strategies have been proposed for video streaming, such as TCP-Friendly Rate-based Control (TFRC) [8] and TCP Libra [9]. Other more radical ideas, such as implementing a form of congestion control for UDP, have also been proposed to allow the use of a protocol with less overhead while keeping the bandwidth management ability of TCP for long lived sessions [10].

There have been many studies done on improving TCP for high-speed networks in general through various methods such as sending “dummy” network packets to artificially maintain large TCP congestion windows [11], to creating newer, more highly

optimized congestion control algorithms such as Yet Another Highspeed TCP (YeAH-TCP) [12]. Still we have not found any formal publication that focused specifically on TCP performance for cloud computing (there are however some preliminary, informal works, for example, Zhu et al [13]).

4. ADAPTIVE VIDEO

4.1 Player

The adaptive video player addresses an inconsistent network by optimizing the video watching experience with the assumption that an uninterrupted video free of stutters and dropped frames is more desirable than a high quality one.

As video content is being streamed, the player is continuously requesting different parts of the video, referred to as segments [1]. Each segment has time duration and a bitrate, so it knows the size of the segment as well as a time index, which specifies when it should be played.

As the end user is watching the video, the adaptive video player continuously checks the available network bandwidth. If the bandwidth is decreasing, the player will request a lower quality segment. By requesting a smaller, lower quality portion of the video the player is confident that the amount of bandwidth available will be sufficient to retrieve the segment in a timely manner. In this way, the player prioritizes uninterrupted playback over video quality, as it is more likely that a smaller file will be transferred quickly compared to a larger one. If it sees that bandwidth is increasing (such as after a network event) it will request a higher quality segment and provide the user with a better viewing experience.

4.2 Server

The corresponding adaptive media server hosts multiple copies of the available videos each encoded at a different bitrate. The media itself is further categorized into segments with each segment corresponding to a particular time index.

As the video is being watched, the player will request a segment by bitrate and time so at any point in time any of the available segments may be played. This is illustrated in Figure 2 where a three second video file is shown on the corresponding server.

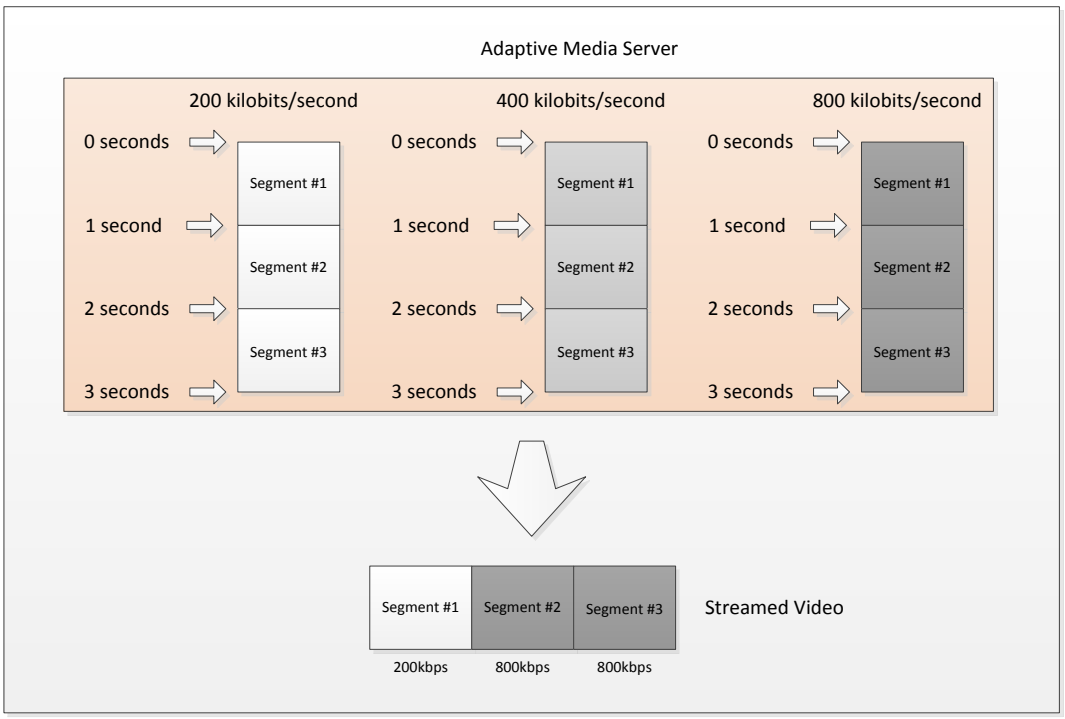

Figure 2: Adaptive Video Streaming Example Session

Note that there are several copies of the video each at a different bitrate and as the video is being played, a different segment is requested during each time period. In Figure 2, the first segment played is the lowest bitrate (200kbps) and the remaining two segments have the highest bitrate (800kbps). We can surmise that a transient network event during the beginning of the streaming session caused a decline in available bandwidth. In response, the adaptive video player requested a lower bitrate segment which was played. After the network recovered and bandwidth was plentiful again, the remaining portion of the video was played using the larger high quality segments.

4.3 Protocols

The adaptive video player uses HTTP exclusively as opposed to other more lightweight protocols such as UDP or video optimized ones such as Real-Time Streaming Protocol (RTSP) [5]. By using HTTP for streaming, content providers are able to realize several advantages enjoyed by normal web traffic such as the ability to seamlessly travel in network configurations that may otherwise restrict or interfere with traditional media streaming protocols such as firewalls or NAT routers. It also allows the use of existing HTTP optimization infrastructures such as CDNs to further improve video performance without having to make changes to how the video is streamed.

5. TCP CONGESTION CONTROL ALGORITHMS

For this paper the following five major TCP congestion control algorithms were considered. Each algorithm is included in the base install of the Ubuntu 12.04 x64 Linux distribution.

5. 1 CUBIC TCP (*cubic*)

Designed for high bandwidth and high delay networks (also known as long fat networks or LFN) *cubic* is one of the most recent and widely used modern congestion control algorithms. The name comes from the calculation of the congestion window which is a cubic function of time since the last time congestion occurred. The end result is less aggressive, more TCP friendly congestion control. *Cubic* is the current (as of version 12.04) default congestion control algorithm in Ubuntu Linux succeeding the previous default of *bic* (see below) and is representative of default TCP behavior [15].

5.2 BIC TCP (*bic*)

Bic (Binary Increase Congestion control) is also meant for use in LFN. It manages the congestion window by using a binary search algorithm to find the maximum congestion window value and maintain is as long as possible. It is seen as a high performance algorithm. It is also a more aggressive congestion control scheme which is less fair to other TCP traffic [16] [17]. It was succeeded as the default TCP congestion control algorithm in Ubuntu Linux by *cubic*.

5.3 TCP Vegas (*vegas*)

Created as a TCP congestion avoidance algorithm at the University of Arizona, *vegas* measures packet delay (as opposed to packet loss) to determine the congestion window. The goal of *vegas* is to use increases in packet delay as an indicator of impending network congestion. By doing so, it is able to detect congestion early and compensate before packet loss takes place [16] [18]. This is generally the smoothest TCP congestion control algorithm with the most consistent performance followed by *cubic* [19].

5.4 H-TCP (*htcp*)

Created by the Hamilton Institute in Ireland, *htcp* is also optimized for LFN. It works by increasing aggressiveness in high bandwidth delay product (BDP) paths by increasing the congestion window at a relatively higher rate while there is no observed packet loss. The net result is available bandwidth is more effectively used, and for smaller data flows it maintains TCP friendliness. However if there are multiple TCP flows and a competing one loses a packet, then *htcp* has the potential to use an unfair amount of resources [20].

5.5 HSTCP - HighSpeed TCP (*highspeed*)

Like *htcp*, *highspeed* is also optimized for LFN. When the congestion window reaches a certain threshold, *highspeed* continues to increase it as a function of the current window size; the larger the window, the greater the increase. As a result, the congestion window will grow at a faster rate and recover more quickly when losses occur [21]. For

slower networks or networks with lower latency, *highspeed* behaves much like other TCP variants and is TCP friendly.

6. EXPERIMENTAL SETUP

The experimental implementation was performed in two stages. The first consisted of the creation of a private cloud that was used to create a baseline optimal system with locally managed hardware and networking resources. After being deployed in the private test cloud, the virtual appliances were uploaded to a public cloud and additional data was collected and analyzed.

6.1 Private Cloud

The private cloud was built using the Eucalyptus Infrastructure-As-A-Service (IaaS) cloud computing platform [22]. It was used to provide a baseline set of measurements to determine how a streaming media server would perform in an optimal cloud computing environment with a large amount of available network bandwidth, no competing network traffic, and exclusive use of existing hardware.

The private cloud was built using two computers, a Hewlett Packard ProBook 8430s and a Dell XPS 17. Both computers were connected using Gigabit Ethernet network cards to a Gigabit Ethernet switch. The network was private and not connected to the Internet, so all traffic was limited to what was generated locally. The 64-bit version of Ubuntu Server 11.0.4 and Eucalyptus Version 2.0, bundled with Ubuntu Server, were installed and configured on both computers.

A virtual appliance consisting of a 64-bit Ubuntu Linux Server 12.0.4 was deployed on this infrastructure configured with 2.0 Gigabytes of RAM, 20 Gigabytes of

storage and allocated one Intel i7 2.3 GHz CPU. It was the only virtual machine (VM) running in the private cloud to ensure no competition for physical resources. Ubuntu Linux was chosen for its high compatibility with cloud computing platforms and its extensive use on the Internet, which facilitated its deployment to both the private and public clouds. The measurements taken from the running instance were used as the baseline for comparing the average data download rate (AVG) as well as the standard deviation (SD).

Finally, as an added test scenario, a simulation of an unreliable network with 5% packet loss was enabled on the client using the DummyNet network emulator [23].

6.2 Public Cloud

The virtual appliance used in the public cloud was also deployed on the public Amazon Elastic Compute Cloud (EC2) [24]. EC2 has 8 locations (known as Amazon Availability Zones or AZ) all over the world where VMs can be run [24]. For this study, the following four Amazon EC2 AZ were used; USWest (Northern California), USEast (Northern Virginia), EUWest (Europe, Ireland), and Asia Pacific (Japan, Tokyo). In each zone, the smallest available 64-bit VM was used which was configured with 1.7 Gigabytes of RAM, 160 Gigabytes of storage and 1 EC2 Compute Unit of processing power (equivalent to an early 2006 1.7 GHz Xeon processor).

For both the private and public cloud environments only a single server was created in each environment for a total of 5 VMs (1 server in the private cloud and 1 server in each of the Amazon AZ). A static IP address was assigned to each instance

along with a standard firewall allowing only secure shell (SSH) and HTTP web traffic through. All configuration was done using standard server cloud deployment tools provided by Eucalyptus and Amazon respectively.

For the client, a 2011 MacBook Pro running the most recent version of the Firefox web browser and Adobe Flash was used. During the private cloud testing, the client was connected to the Gigabit Ethernet switch using a Gigabit Ethernet Network card. For the public cloud tests, the client was connected to the Internet via 1.5 Mbps ADSL.

6.3 Software

The client and server software chosen for this study come from the Open Source Media Framework (OSMF) sponsored by Adobe Systems [25]. It is a free, open source development framework used for creating and distributing video on the web. The OSMF server software consists of a set of Apache web server plugins collectively known as the Origin HTTP modules. The video stored on the media server is accessed via a custom media player contained in an Adobe Flash (swf) file that is configured to dynamically call the different video segments. For this study, the OSMF Sample Player for HTTP Dynamic Streaming was used without any modification [26].

The adaptive video server was built using the 64-bit version of the Ubuntu 12.04 server with kernel 3.2.0 customized to run on cloud-computing platforms mentioned previously [27]. The configuration steps included downloading the latest operating

system security patches and software updates in addition to the following additional packages that were manually installed from the standard Ubuntu software repositories:

- apache2
- openssl
- expat
- libnspr4-0d

6.4 Media

The video used for streaming was “Big Buck Bunny” from the Peach open movie project [28]. The movie was chosen for its open licensing (Creative Commons), relatively long run time (almost 10 minutes), and availability of a High Definition (900 Megabyte MP4) video.

Prior to being placed on the server for streaming, the video was converted to make it suitable for adaptive streaming by taking the original high definition video and creating multiple copies with each copy having a slightly different bitrate as shown in Table 1. The resulting videos were then packaged using Adobe’s f4f file packager software to make it suitable for adaptive video streaming.

.

Table 1: Video bitrates used for media encoding.

2750 kbps
2040 kbps
1520 kbps
1130 kbps
845 kbps
630 kbps
470 kbps
350 kbps

The bitrates used are the same as those used by Microsoft in demonstrating their implementation of adaptive video streaming [29].

7. RESULTS

7.1 Private Cloud

In this section the experimental results are presented using the private cloud setting as described in Section 6.1.

Figure 3 shows the average (Avg) and standard deviation (SD) of the throughput in kilobytes/sec (Kbytes/Sec) of the adaptive video download for all five TCP variants.

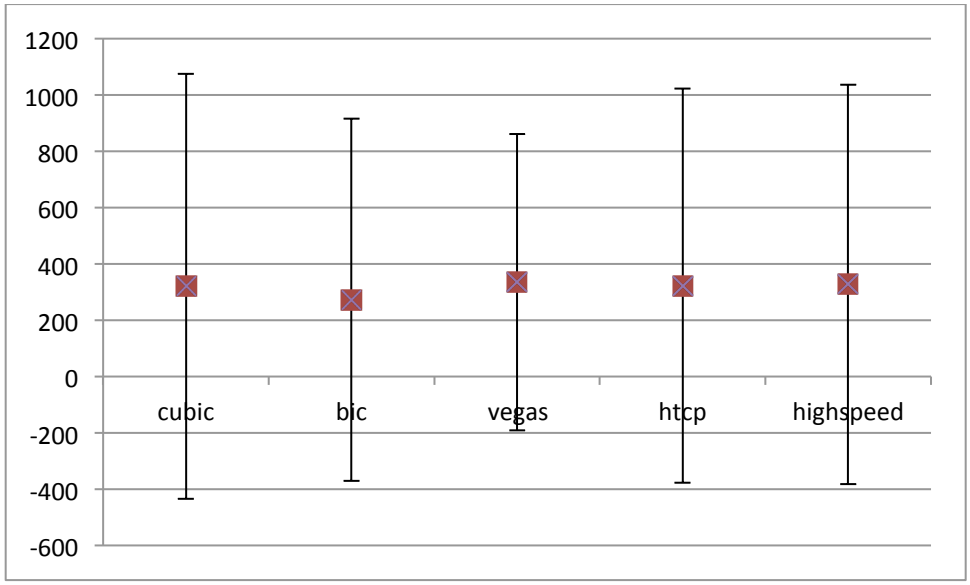

Figure 3: Avg and SD of Kbytes/Sec adaptive video download in private cloud.

In this ideal scenario (very high available bandwidth with no congestion or competition), all of the algorithms were able to deliver a consistently good download and video viewing experience with the video played in its entirety, at the highest quality with no discernible problems. The measured Avg throughput was 270-350 Kbytes/sec with a relatively high SD with values ranging from 550-770 Kbytes/sec.

For all five congestion control algorithms, the highest average throughput was observed using the *cubic* congestion control algorithm followed by *highspeed, htcp, bic,* and *vegas*. As expected *vegas* had the lowest SD. *Cubic* however had the highest SD contrary to its expected performance.

When a 5% packet loss was introduced in the private cloud, the observed behavior changed significantly as all 5 congestion control algorithms used additional network resources to compensate for the loss. Figure 4 shows the results as Avg throughput doubled to 560-730 Kbytes/sec. We believe this to be due to data retransmissions that occurred to compensate for the 5% packet loss. The SD however dropped to 350-370 Kbytes/sec, with no significant difference among the five variants.

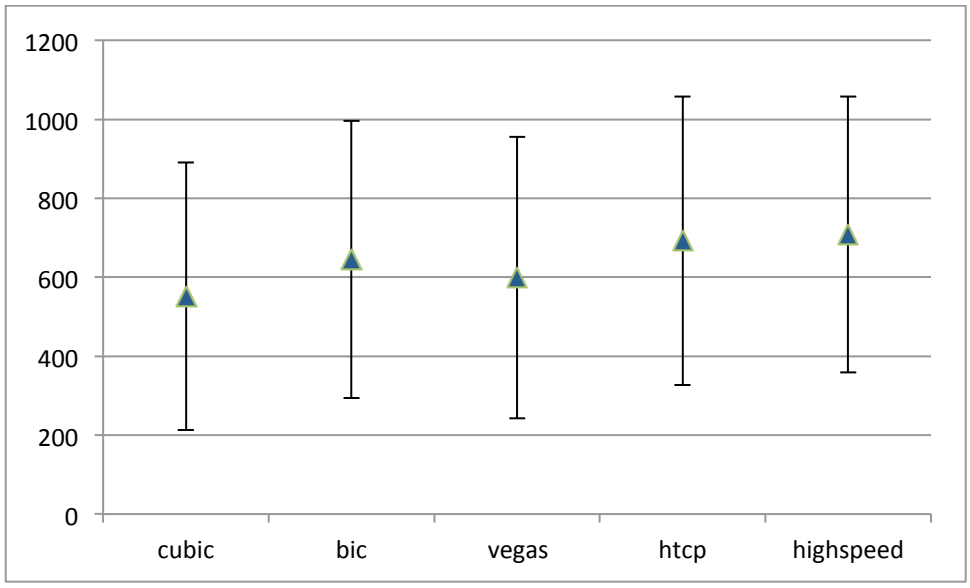

Figure 4: Average and SD of Kbytes/Second adaptive video download in private cloud with 5% packet loss.

Despite the relatively high network congestion, each test run again resulted in a consistent viewing experience with the video played in its entirety at the highest resolution with no stutter or dropouts.

7.2 Public Cloud

In this section, we present the experimental results using the public cloud environment as described in Section 6.2.

During the deployment of the video server to the public Amazon EC2 cloud, an inconsistency was discovered with the sample OSMF adaptive video player. While the player is designed to adjust video bitrate based on the available bandwidth, it became apparent that regardless of bandwidth, the player consistently attempted to play the

highest bitrate file. This behavior was also observed by Akhshabi, Begen, and Dovrolis who theorized that the sample player was built to smooth out short variations in bandwidth as opposed to automatically adjust for optimal playback [30].

Based on the purpose of the experiment, which was to observe the effect of TCP variants in the cloud for video streaming rather than optimizing adaptive video player settings, it was decided to use the sample video player as is and instead modify the configuration on the server by removing the higher bitrate versions of the video. The files removed were those whose bitrate exceeded the last-mile bandwidth of 1.5 Mbps (or 187.5 Kbytes/sec). As a result, instead of streaming media with eight different bitrates, the main configuration file was modified to use only those bitrates lower than 1.5 Mbps; i.e. media encoded with the following five different bitrates: 1130 kbps, 845 kbps, 630 kbps, 470 kbps and 350 kbps (refer to Section 6.4.)

Even though this may somewhat disagree with the general video setting rule that the optimal video bitrate should be half of the available bandwidth [1], we felt that having a video stream at a bitrate close to network capacity was better suited for observing differences between TCP variants.

It was also discovered that the prevailing network connectivity between the public cloud providers and the test client was insufficient to provide a consistent streaming session as the time to stream the entire 10 minute video successfully was sometimes in excess of 30 minutes. Due to this limitation, it was decided that rather than streaming the entire video as was done in the private cloud, we would stream *8 minutes of video in each*

experiment and take measurements during that time period. This interval was chosen to allow enough time to obtain useful data that would not be affected by short-term variations in available bandwidth.

7.3 Throughput and Total Amount of Video Streaming

The results are shown in Figure 5, with the five TCP variants in each of the four AZ.

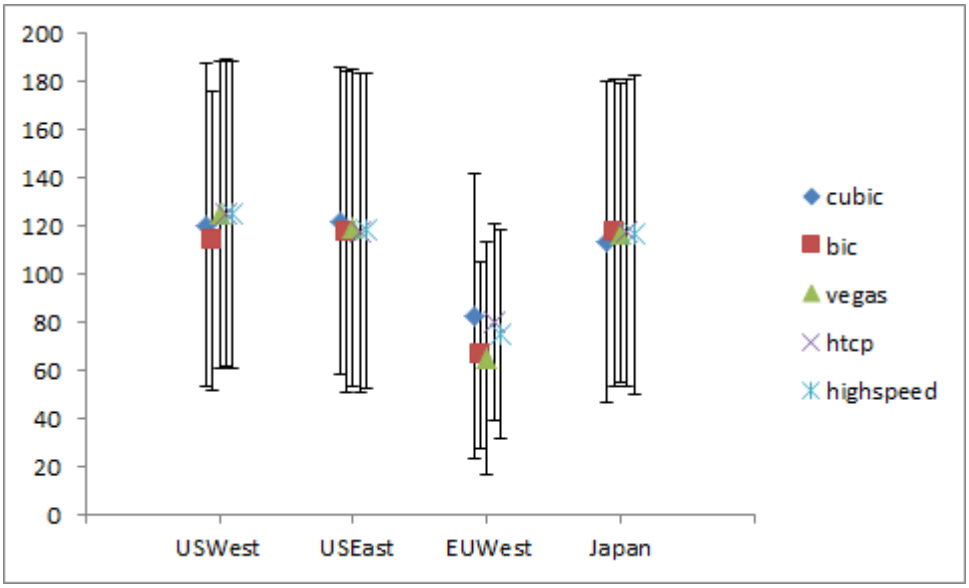

Figure 5: Average and SD of number of Kbytes/Second Downloaded

With the exception of EUWest (which was the AZ furthest away geographically from the test client) the algorithms in the other 3 AZ (USWest, USEast, and Japan) all

achieved similar throughput in terms of both Avg (approximately 120 Kbytes/sec) and SD (approximately 60 Kbytes/sec).

By contrast, EUWest showed a throughput performance between 63-83 Kbytes/sec, significantly lower compared to the other three regions. Within EUWest the highest overall throughput was achieved by *cubic*, followed by *htcp*, *highspeed*, *bic*, and finally *vegas*. The highest observed SD in EUWest was also *cubic* followed by *vegas*, *highspeed*, *htcp*, and *bic*. These results were unexpected as *cubic* is optimized for both performance and TCP friendliness which under heavy congestion, as we believe was occurring when these measurements were taken, we would have expected the other LFN optimized congestion control algorithms to have shown the best network performance. In addition both *cubic* and *vegas* are architected to be the most consistent congestion control algorithms with the least amount of variation. Yet both had the highest observed SD compared to the other more aggressive congestion control algorithms.

Shown in Figure 6 is the total amount of data downloaded during the 8 minute time period. For EUWest these results generally agree with the throughput results in Figure 5 with *cubic* showing the highest amount of overall data downloaded followed again by: *htcp*, *highspeed*, *bic*, and *vegas.*

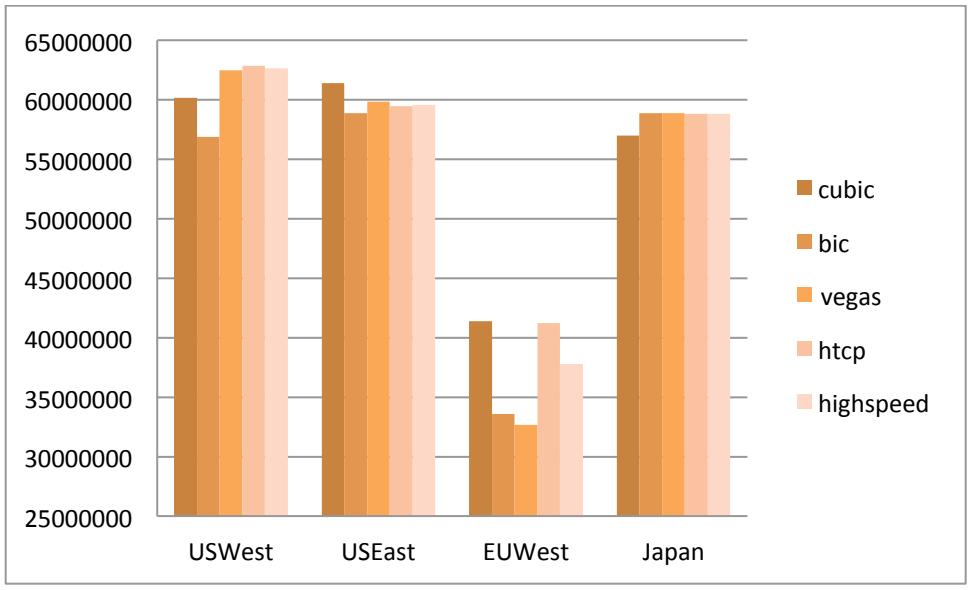

Figure 6: Total Amount of Megabytes Downloaded

7.4 Effectiveness of Video Streaming – Percentage of Video Played

The video watching experience for the client was also measured as we wanted to see if there was a correlation between network throughput and the end user video watching experience. To that end, the overall percentage of the sample movie that successfully played during the 8-minute time interval was recorded and is shown in Figure 7.

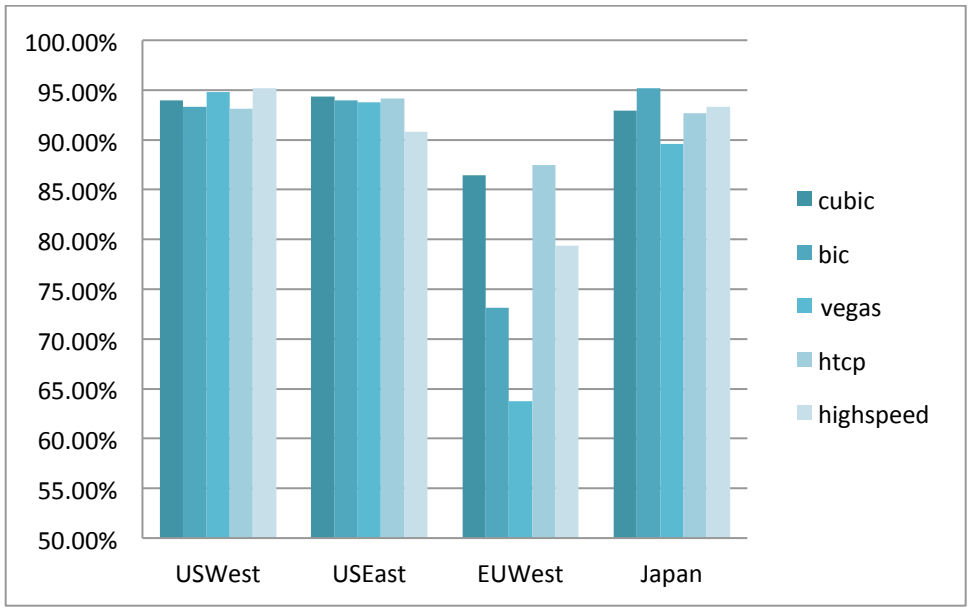

Figure 7: Percentage of Video Played

Again except for the EUWest, the other three regions show similar results for all five TCP variants with both US based AZ for all intents identical and Japan having the highest single overall percentage with *bic* and EUWest having the lowest overall percentage with *vegas*. EUWest also had the greatest observed performance variation with over 20% less of the video played using *vegas* compared to *cubic*.

An interesting distinction appears when comparing percentage of video played with overall amount of data downloaded. When comparing the two it was found that a higher throughput or download amount does not necessarily imply a higher percentage of video played.

Taking Japan as an example, the congestion control algorithm with the highest percentage of video played was *bic* followed by *highspeed*. In terms of absolute amount of data downloaded however *bic* had the highest amount (which would seem to make sense) followed by *vegas* which had the lowest overall percentage of video played in Japan.

In EUWest where there was the most variability in performance, and which we feel had the most meaningful results, *htcp* had the highest percentage of video played followed by *cubic*. In contrast *cubic* had the largest amount of data downloaded followed by *htcp*.

7.5 Number of Concurrent TCP Connections

Originally the number of TCP connections was not measured as it was not thought that the adaptive media player employed multiple TCP streams for enhancing network performance. During the study though, it was discovered that there was a significant difference in the number of TCP connections used when different congestion control algorithms were enabled. As a result, the number of TCP connections was recorded and analyzed as an additional data point.

It was observed that the OSMF adaptive video player used multiple TCP connections extensively when streaming in both the public and private cloud. This was surprising as the practice of using multiple TCP connections for video streaming is seen as a separate TCP unfriendly optimization method used instead of dynamic streaming.

For streaming video from the US based public cloud, the number of TCP connections were similar among the five congestion control algorithms. When streaming from both the international clouds, they varied much more averaging slightly less than 20 as shown in Figure 8. The lowest number of TCP streams was 9 and the highest was 28; both of these values occurred in EUWest.

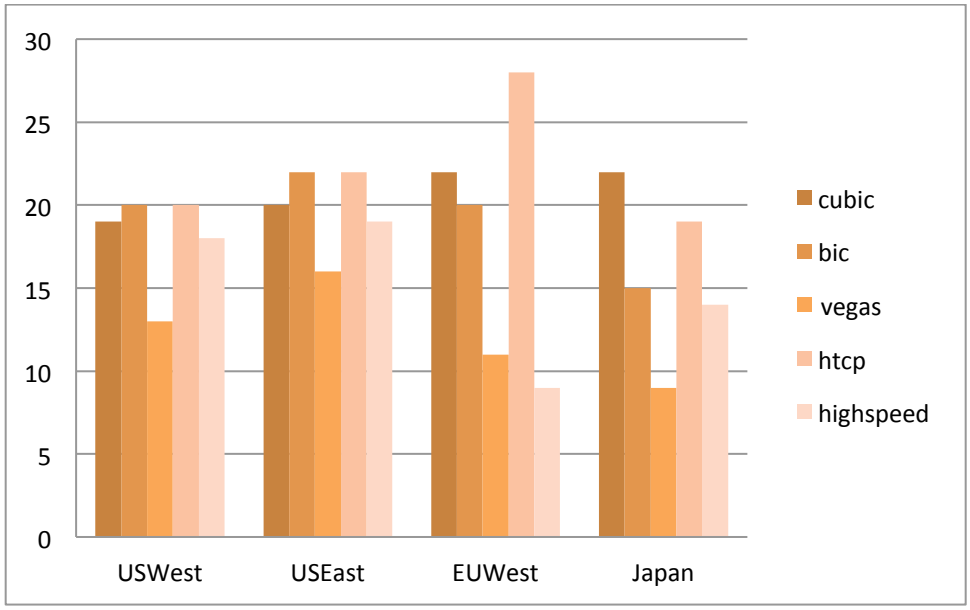

Figure 8: Total Number of TCP Connections.

Focusing again on EUWest, where the network condition was the worst, *htcp* used the most TCP connections followed by *cubic*. Both also streamed the highest percentage of video which would indicate a correlation between number of connections and streaming video experience. This however is confounded by *highspeed*, which used the

fewest number of TCP connections in EUWest, yet had the third best percentage of video played.

Comparing Figure 7 and Figure 8 we see that *highspeed* played almost 20% more of the sample video using 9 TCP streams compared to *vegas* which used 11 TCP streams.

7.6 Smoothness – TCP Congestion Control Window

The behavior of the congestion window is an indication of the "smoothness" of TCP and the algorithm's friendliness towards other network streams. The size of the congestion window for all five TCP congestion windows while the video was streaming in EUWest is shown in Figure 9. Only EUWest is shown as similar behaviors were observed for the other three AZ.

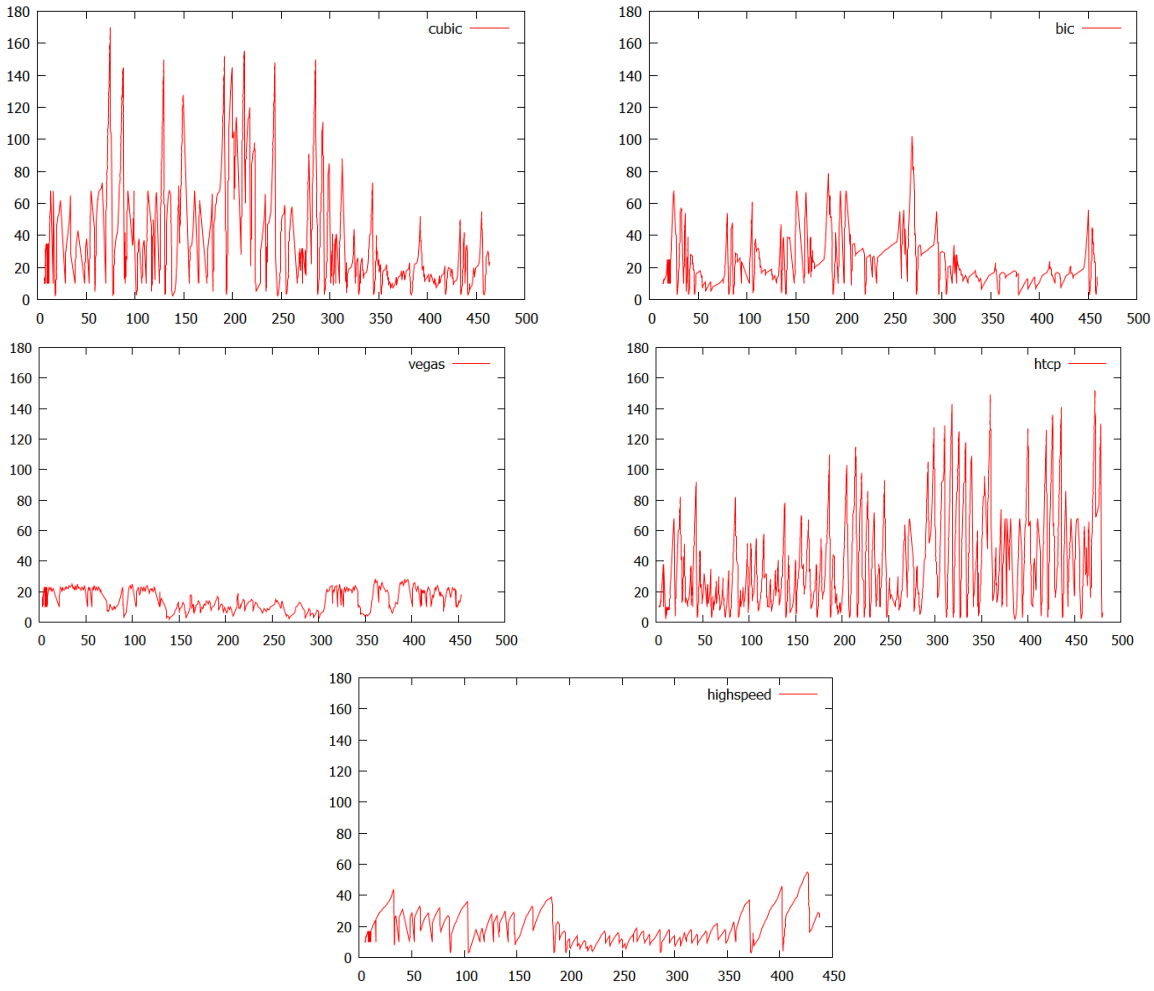

Figure 9: Congestion Windows for EUWest AZ

Vegas and *highspeed* had the smoothest TCP behavior of the 5 congestion control algorithms followed by *bic*, *cubic*, and *htcp*. It was expected that *vegas* would have the least variability and be the smoothest followed closely by *cubic*. However it is seen that *cubic* actually has a significant amount of variability (second to *htcp*) which leads us to believe that under congestion *cubic's* TCP friendliness may be compromised for the sake of performance.

7.7 Summary

The results from EUWest which presented the most problematic network path and is most indicative of the congestion control algorithm's behavior under severe real world congestion are summarized below in Table 2.

Table 2: Performance Summary (EUWest)

	1st	**2nd**	**3rd**	**4th**	**5th**
Percentage Played	*htcp*	*cubic*	*highspeed*	*bic*	*vegas*
Throughput	*cubic*	*htcp*	*highspeed*	*bic*	*vegas*
Amount Downloaded	*cubic*	*htcp*	*highspeed*	*bic*	*vegas*
# of TCP connections	*highspeed* (least)	*vegas*	*bic*	*cubic*	*htcp* (most)
Smoothness	*vegas*	*highspeed*	*bic*	*cubic*	*htcp*

In terms of video streaming performance, the first three criteria are the most important with the percentage of video successfully played being the most important to the end user.

For TCP friendliness and fairness, the bottom two are the most relevant as they define how much of the available network resources are being used (sometimes unfairly) by the player.

8. CONCLUSION

Extensive experimental studies were conducted to evaluate TCP performance in supporting cloud-based adaptive video streaming. Five major TCP variants that are part of the base Ubuntu Linux server distribution (including the default) were included in these experiments. A private cloud was first used to create a baseline measurement followed by four geographically different public cloud deployments.

To evaluate streaming video quality, the following network metrics were evaluated: TCP throughput, total amount of data downloaded, and TCP congestion window behavior. An additional metric of total percentage of video played was also measured to factor in the end user experience as a separate metric altogether. In addition, an unexpected fifth metric of concurrent TCP connections was discovered and evaluated after its importance in adaptive video streaming was discovered.

In a private cloud and a domestic public cloud, it was found that the choice of congestion control algorithm was not as impactful on the overall video streaming experience. By contrast, the choice of algorithm when streaming from an international cloud provider had a significant impact when using an adaptive video player.

It was found that *htcp* and *cubic* were the two best performing congestion control algorithms for streaming video providing the highest percentage of video playback coupled with the highest throughput and absolute amount of data downloaded. Showing

lower overall video streaming performance were *highspeed* and *vegas* which, by contrast, were more TCP friendly and had smoother network behavior in the cloud.

Amongst all five available algorithms, *highspeed* had the best balance between video quality and TCP friendliness.

It is believed that this work contributes significantly to the network and cloud computing communities towards optimizing TCP or choosing a good alternative for cloud computing [14].

9. FUTURE WORK

Future work may include designing an improved TCP variant or TCP alternative for the cloud that may be optimized for video streaming instead of LFN. It would also be of interest to use different streaming clients such as smartphones or tablets and see if there is a difference in video streaming quality when video is accessed by relatively low power device over a cellular service.

The use of other public clouds would also be useful as the bandwidth limitations may be addressed by using a smaller niche provider such as GoGrid [31] or even a different technology base altogether such as Microsoft Azure [32].

In addition, similar experiments may be carried out using other adaptive video players for a broader understanding of the effect of TCP congestion control algorithms on adaptive video streaming over the cloud in general. This may be further elaborated by using different adaptive video streaming algorithms on the video player itself as the OSMF player used in this study is an open source project which allows the modification of the adaptive video streaming algorithm.

This study also focused on the streaming characteristics using one server per client which is not a realistic real world scenario. Examining the streaming video performance using several simultaneous clients would be of great interest as the TCP friendliness of a congestion control algorithm would become a factor.

LIST OF REFERENCES

[1] Stefan Lederer, Christopher Müller, and Christian Timmerer. 2012. Dynamic adaptive streaming over HTTP dataset. In *Proc. of the 3rd Multimedia Systems Conf. (MMSys '12)*. ACM, New York, NY, USA, 89-94.

[2] C. Kaiser. (2010, December 22). HTML 5 and Video Streaming. [Web log comment]. Retrieved from http://techblog.netflix.com/2010/12/html5-and-video-streaming.html

[3] http://www.hulu.com/about/video_quality

[4] Sunand Tullimas, Thinh Nguyen, Rich Edgecomb, and Sen-ching Cheung. Multimedia streaming using multiple TCP connections. ACM Trans. Multimedia Comput. Commun. Appl. 4, 2, Article 12 (May 2008).

[5] Chia Jung Chen, Rong Guey Chang, Chih Wen Hsueh, "Wireless Smooth Data Streaming on Application Layer," in Proc. of 2011 IFIP 9th Int. Conf. on Embedded and Ubiquitous Computing, pp. 384-389.

[6] Kuschnig, I. Kofler, and H. Hellwagner. Improving InternetVideo Streaming Performance by Parallel TCP-basedRequest-Response Streams. In Proc. of the 7th AnnualIEEE Consumer Communications and Networking Conference (IEEE CCNC 2010), January 2010

[7] Ashvin Goel, Charles Krasic, and Jonathan Walpole. 2008. Low-latency adaptive streaming over tcp. ACM Trans. Multimedia Comput. Commun. Appl. 4, 3, Article 20 (September 2008)

[8] Soo Hyun Choi and Mark Handley. 2007. Fairer TCP-friendly congestion control protocol for multimedia streaming applications. In Proc. of the 2007 ACM CoNEXT Conf. (CoNEXT '07). ACM, New York, NY, USA.

[9] Gustavo Marfia, Claudio E. Palazzi, Giovanni Pau, Mario Gerla, Medy Y. Sanadidi, and Marco Roccetti. 2007. Balancing video on demand flows over links with heterogeneous delays. In Prof. of the 3rd ICST Int. Conf. on Mobile Multimedia Communications (MobiMedia '07). Brussels, Belgium.

[10] Eddie Kohler, Mark Handley, and Sally Floyd. 2006. Designing DCCP: congestion control without reliability. ACM SIGCOMM Comput. Commun. Review, 36, 4 (August 2006), 27-38.

[11] Amit Mondal and Aleksandar Kuzmanovic. 2010. Upgrading mice to elephants: effects and end-point solutions. IEEE/ACM Trans. Netw. 18, 2 (April 2010), 367-378. DOI=10.1109/TN

[12] F. Vacirca, A. Baiocchi, and A. Castellani. Yeah-TCP: yet another highspeed TCP. In International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet), pages 37--42, 2007.

[13] Jiang Zhu, et al., presentation "TCP in a World of Cloud Services," http://yuba.stanford.edu/trainwreck/train-wreck-presentation-Zhu.pdf

[14] FASP (Fast and Secure Protocol) Technology, Aspera Inc., http://www.asperasoft.com/en/technology/fasp_overview_1/fasp_technology_overview_1

[15] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review, 42, 5 (July 2008), 64-74.

[16] Callegari, C.; Giordano, S.; Pagano, M.; Pepe, T.; , "Behavior analysis of TCP Linux variants," Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2010 International Symposium on , vol., no., pp.218-225, 11-14 July 2010

[17] M. Escheikh and K. Barkaoui. 2007. Performance analysis of high-speed TCP protocols BIC and CUBIC with AQM in lossy networks. In Proceedings of the IASTED International Conference on Communication Systems, Networks, and Applications (CSNA '07), Pingyi Fan and Jie Li (Eds.). ACTA Press, Anaheim, CA, USA, 31-35.

[18] K. N. Srijith, Lillykutty Jacob, and A. L. Ananda. 2005. TCP Vegas-A: Improving the Performance of TCP Vegas. Computer Communications, 28, 4 (March 2005), 429-440.

[19] H. Jamal, K. Sultan, Performance Analysis of TCP Congestion Control Algorithms, Internation Journal of Computers and Communication, Issue 1, V2, 2008

[20] Grenville Armitage, Lawrence Stewart, Michael Welzl, and James Healy. 2008. An independent H-TCP implementation under FreeBSD 7.0: description and observed behaviour. ACM SIGCOMM Comput. Commun. Review, 38, 3 (July 2008), 27-38.

[21] Sally Floyd. Highspeed TCP for Large Congestion Windows. RFC 3649, Interent Engineering Task Force, Dec. 2003.

[22] *Open Source Private and Hybrid Clouds from Eucalyptus.* Eucalyptus Systems Incorporated. Web. 25 February. 2012. http://www.eucalyptus.com/,

[23] Luigi Rizzo. *The dummynet project.* Web. 1 April. 2012. http://info.iet.unipi.it/~luigi/dummynet/

[24] *Amazon Elastic Compute Cloud (Amazon EC2)* Amazon Web Services LLC 2012. Web. 1 April 2012. http://aws.amazon.com/ec2/

[25] *Open Source Media Framework.* Adobe Systems Incorporated 2010. Web. 25 February 2012. http://www.osmf.org

[26] *Open Source Media Framework Developers.* Adobe Systems Incorporated 2010. Web. 25 February 2012. http://www.osmf.org/developers.html

[27] *Ubuntu Cloud Images.* Canonical Ltd 2012. Web. 1 February 2012. http://cloud-images.ubuntu.com

[28] *Big Buck Bunny.* Blender Foundation. Web. 1 February 2012. http://www.bigbuckbunny.org/

[29] *Experience IIS Smooth Streaming*. Microsoft 2012. Web. 26 February 2012. http://www.iis.net/media/experiencesmoothstreaming

[30] S. Akhshabi, A. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In Proc. of the 2nd Annual ACM Conf. on Multimedia Systems (MMSys '11). ACM New York, NY, USA, 157-168. 2011

[31] *Welcome to GoGrid.* GoGrid 2012. Web. 10 June 2012. http://gogrid.com

[32] *Windows Azure: Microsoft's Cloud Platform.* Microsoft 2012. Web. 26 February 2012. http://www.windowsazure.com