International Journal of Future Generation Communication and Networking
Vol.7, No.1 (2014), pp.195-208
http://dx.doi.org/10.14257/ijfgcn.2014.7.1.20

Enhancing TCP to Improve Throughput of HTTP Adaptive Streaming

Xinying Liu[1], Aidong Men[2] and Peng Zhang[3]

[123]Beijing University of Posts and Telecommunications
No.10 Xitucheng Road, Haidian District, Beijing, China
[1] liuxinying2@gmail.com, [2] menad@bupt.edu.cn, [3] bupt.zhp@gmail.com

Abstract

With increasing video service consumption through IP network, HTTP Adaptive Streaming (HAS) technology has become popular. The chunked transmission and application-layer adaption create a different traffic pattern than traditional progressive download where the entire video is downloaded with a single request. Its start and stop activity cause burst and retransmission timeout (RTO) which result in more packet loss and longer packet transfer delay. Both of them can substantially degrade the achievable throughput of HAS streams. However, few works studied the impact of HAS activity pattern on TCP performance. In this paper, the reason caused the poor performance of HAS is pointed out, and the two server-based algorithms are proposed to improve the achievable throughput of HAS streams. The two algorithms are implemented at the beginning and end of a chunk transmission separately. They aim to improve the performance of HAS through smoothing the burst and avoiding RTO. The simulation results show that these proposed algorithms increase the achievable throughput of HAS streams greatly and keep HAS streams friendly with other TCP flows at the same time.

Keywords: *HTTP Adaptive Streaming, TCP congestion control, TCP pacing, Retransmission Timeout*

1. Introduction

At present there are many more Internet video users than ever before. Globally, consumer Internet video traffic will be 69% of all consumer Internet traffic in 2017, up from 57% in 2012. And advanced consumer Internet video (3D and HD) will increase 4-fold between 2012 and 2017 [1]. Much of the video traffic is transmitted over HTTP protocol. Nowadays, a new approach referred to as HTTP adaptive streaming (HAS) is becoming popular. It enables high quality streaming of media content over the Internet delivered from conventional web servers. Therefore, it can traverse easily through the network address translation (NAT) routers and firewalls widely deployed in the Internet. Moreover, it uses a flexible rate-adaptation scheme that delivers the highest quality video possible despite changing network conditions dynamically without stops and stutters.

HTTP adaptive streaming partitions the video content, such as a movie, into many small file chunks, each chunk containing a short interval of playback time. The content is made available at a variety of different bit rates. The clients automatically select from the alternatives the next chunk to download and play back based on current network conditions. The chunked transmission and application-layer adaption create a very different traffic pattern than traditional progressive video download where the entire video is downloaded with a single request [2]. This characteristic of HAS leads to the start and stop nature of traffic pattern.

The HAS traffic pattern affects the performance of TCP. When restarting data flow after an idle period, current implementations either decrease the congestion window by half or use the prior congestion window. The latter case sends a large burst of back-to-back packets. When the packet loss happened during the transmission of a chunk, if the server runs out of new data to transmit before the end of recovery phase, the RTO occurs. Thus, the interaction between HAS and TCP cause burst and retransmission timeout (RTO) at the beginning and end of a chunk's transmission respectively. These two problems degrade the achievable throughput of HAS streams. The available bandwidth cannot be fully utilized.

Some research works have been conducted to improve the performance of web-based streaming transmission [7, 8]. And Heidemann described several performance problems resulting from interactions between implementations of Persistent-HTTP (P-HTTP) and TCP [12]. However, few focus on the interaction between HAS and TCP. Ref. [2] investigated experimentally this issue and found conventional TCP pacing was ineffective at improving throughput for HAS streams.

In this paper, two algorithms are proposed to improve the achievable throughput of HAS streams. The one is novel pacing algorithm to smooth the burst and avoid slow start after RTO at the beginning of a chunk transmission. The other is Reducing Tail Loss (RTL) algorithm to reduce the probability of RTO by sending minimum dummy packets to keep the TCP connection active. The simulation results show that these proposed algorithms increase the achievable throughput of HAS stream by 2.5 times at the most and also keep HAS streams friendly with other TCP flows. Our algorithms are based on the modification to TCP protocol which requires only server-side changes. This makes it particularly beneficial for HAS servers, thus providing motivation for service provider to deploy.

The remainder of this paper is organized as follows. In Section II we introduce background and related work. Section III describes two enhanced algorithms. Simulation results and analysis are provided in Section IV. Finally, Section V concludes the paper.

2. Background and Related Work

2.1. Background

In order to understand the interaction between HAS stream's on-off pattern and TCP, the TCP transmission is divided into three phases, which is illustrated in Figure 1 [2]. The first phase is Initial Burst. It occurs when the server fills its congestion window quickly at the start of each response, which has grown over time from previous transmissions. Once the window is full, the sender waits for returning ACKs before sending more packets and growing the congestion window further. The following phase is ACK Clocking. It occurs when the server receives feedback from the client and has more packets to send. And the last phase is Trailing ACKs. It occurs when the server has sent all packets belonging to a chunk and is waiting for the feedbacks.

The impact of packet loss during each of these phases is described as follow. In the Initial Burst, server sends a burst of back-to-back packets. When this burst of data is too large, it can overrun queues at intermediate routers, lead to packet loss and possibly lower overall performance. A large burst of packets can also affect other connection sharing the same router. In the ACK Clocking, fast retransmit and fast recovery minimize impact of packet loss. In the Trailing ACKs, if the server does not have new data to send after the loss packet, the client will not generate enough duplicate ACKs to trigger fast retransmit. Then the loss is recovered by RTO only. This phenomenon is called tail loss. This RTO may increase, at minimum, one-minute delay on network and reduce the congestion window of next chunk transmission.

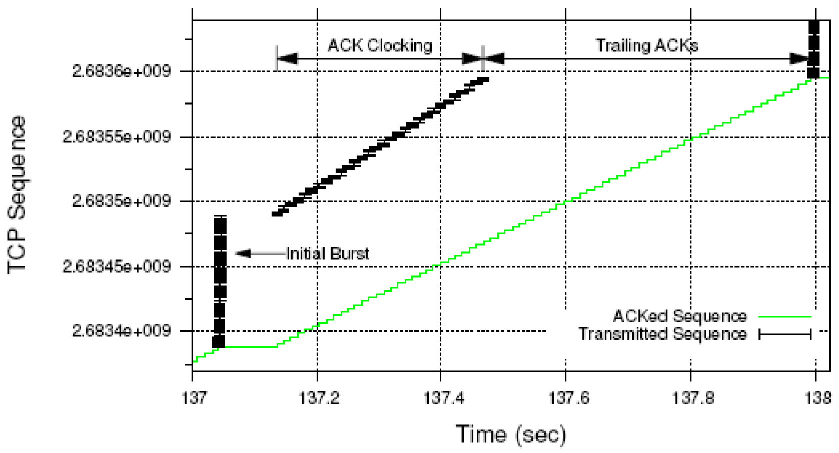

Figure 1. Three Phase of TCP Transmission [2]

2.2. Related Work

The burst of flow may cause higher queuing delay, more packet loss, and lower throughput. RTO will increase the download time and decrease the throughput. All of these are destructive for streaming transmission. Many solutions have been proposed to solve these problems. TCP pacing has been studied to reduce burst of TCP traffic [3] [4]. However using the TCP pacing algorithm in the whole transmission may brings other problems, such as weak competitiveness with other non-pacing flows. In addition, these methods do not solve the tail loss. The general method to reduce the probability of RTO is to convert retransmission timeout into fast recovery [5, 6]. The root cause of RTO problems is different in these two literatures. In [5], RTO happened when a connection’s congestion window is small or when a large number of segments are lost in a single transmission window. The RTO occurred because of retransmit packet is lost in [6]. However, the reason that RTO occurs in HAS transmission is that there are no new packets to trigger fast retransmission.

Although, a few of studies improve the performance of HAS by proposing the solution based on server-side. Ref. [7] proposed a server-based traffic shaping method that aims to eliminate the instability problem. But the shaping mechanism in [7] focus on the instability problem caused by HAS’s on-off activity, they did not consider the impact of HAS activity pattern on TCP performance. Ref. [8] introduced a server-side mechanism that uses TCP to rate limit YouTube video streaming, but the technique in [8] is for the traffic delivered using HTTP progressive download. The most related work to this paper is [2] and only [2] considered the impact of interaction between HAS and TCP on the performance of HAS. However, it found that using pacing only was ineffective at improving throughput for HAS streams and left tail loss problem unsolved. And packet losses at the end of chunk transmission have a greater impact on throughput.

3. Optimizing Methods

In this section, we present two algorithms to improve the performance of HAS. Firstly, we use the novel pacing algorithm to smooth the burst in the initial burst phase and skip the slow start after retransmission timeouts occurred in the trailing ACKs phase. Then we use the reducing tail loss algorithm to reduce the probability of RTO in the trailing ACKs phase by sending redundant packets. Both methods are server-based, *i.e.*, only TCP protocol of server-side needs to be modified. We do not alter other TCP behaviors, such as, connection set-up and the initial TCP slow start period.

3.1. Novel Pacing Algorithm in Initial Burst

The burst in the initial burst phase may cause multiple packet loss and the RTO in the trailing ACKs phase will re-initialize congestion window by entering slow start. The following algorithm can improve the performance of HAS by solving these two issues.

The novel pacing algorithm uses two different pacing strategies for different situations. Figure 2 gives the flowchart of the novel pacing algorithm. It is implemented during the initial burst period. When starting a new chunk transmission, the server detects whether there is a RTO in the last Trailing ACKs phase of last chunk. If RTO does not happen in the last Trailing ACKs phase, packets are scheduled to send at the interval of *rtt/cwnd*, where *rtt* is the round-trip time (RTT) and *cwnd* is the current congestion window. This ensures that packet transmission spreads across the whole duration of the first RTT instead of sending burst. When the first packet of the new chunk is acknowledged, the novel pacing algorithm is terminated.

When RTO is detected in the last Trailing ACKs phase, the transfer is entering slow start. However, RTO is not the only element that should be considered here. When TCP has not received a segment for a relatively long idle period, the connection should reduce its congestion window by half for every RTT [9]. So, if RTO triggered slow start, packets are scheduled to send at the interval of *2rtt/cwnd'* where *cwnd'* is the congestion window before RTO happens. The novel pacing algorithm is terminated, once the first packet of the new chunk is acknowledged. If long idle time occurred, the new chunk transmission restarts with decayed congestion window skipping the pacing algorithm.

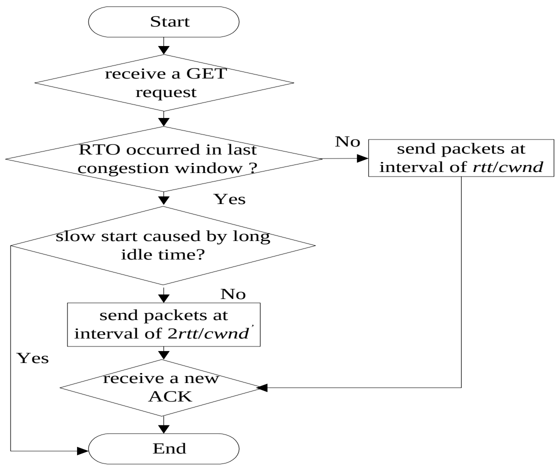
Start
receive a GET request
RTO occurred in last congestion window ?
No
send packets at interval of rtt/cwnd
Yes
slow start caused by long idle time?
No
send packets at interval of 2rtt/cwnd'
Yes
receive a new ACK
End

Figure 2. Flowchart of the Novel Pacing Algorithm

3.2. Reducing Tail Loss Algorithm in Trailing ACKs Phase

The server has sent all packets belong to a chunk in the ACK clocking phase. When at least one of the last three packets is lost, the retransmission timeout will happen during the trailing ACKs phase due to not having triple duplicate ACKs to trigger fast retransmit. RTOs are detrimental to application latency, especially for streaming over HTTP.

In order to protect the transport from RTOs, we propose reducing tail loss algorithm to resend multiple dummy packets. The dummy packet is the last packet of each chunk. The dummy packets can lengthen the TCP data stream to keep TCP flow active and reduce the probability of RTO. Let us suppose that the last congestion window of the chunk is n packets. And the amount of the lost packets among the last three packets is j, where j = 1, 2, 3. When the server receives the feedback of packet belongs to one chunk's last congestion window, it will resend the dummy packet. Then after three duplicate ACKs received, the server triggers fast retransmit.

Initially we plan to send the dummy packet for every feedback which is received in the tailing ACK phase. For example, if the penultimate packet is lost but the last one is delivered successfully, illustrated in Figure 3, then we have n-1 dummy packets to send until all packets belong to this chunk are acknowledged. However, it may bring unnecessary burden to the network especially when the last congestion window is large. In contrast, we choose an optimal value for the amount of dummy packets to reduce the probability of RTO to a given threshold.

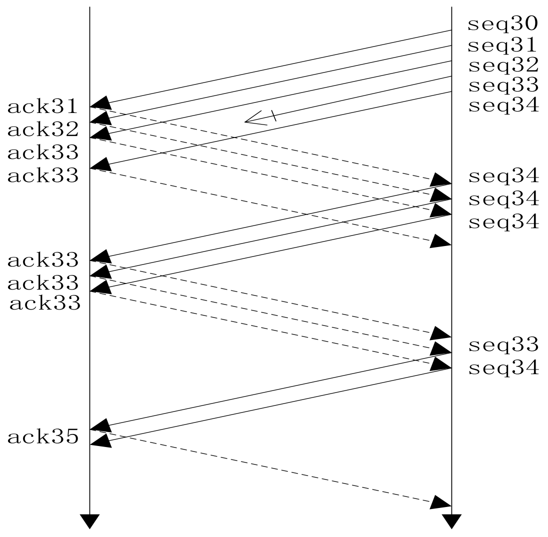

Figure 3. Reducing Tail Loss Algorithm (One of the Last Three Packets from the Chunk is Lost)

Let us assume each packet belong to a chunk has identical loss probability *p*. And Let *m* be the number of dummy packets needed to be sent. *P(1)* is the probability that at least one of *m* dummy packets is received by client. *P(2)* is the probability that at least two packets of *m* dummy packets are received by client. And *P(3)* is the probability that at least three of *m* dummy packets are received by client. $P_B(RTO)$is the probability of RTO before using the algorithm, and $P_A(RTO)$ is the probability of RTO after using the algorithm, and $P_A(\overline{RTO})$ is the probability that RTO is not happening after using the algorithm. So we can get the following equations:

$$P(1) = 1 - p^m \tag{1}$$

$$P(2) = 1 - p^m - \binom{m}{1} p^{m-1}(1-p) \tag{2}$$

$$P(3) = 1 - p^m - \binom{m}{1} p^{m-1}(1-p) - \binom{m}{2} p^{m-2}(1-p)^2 \tag{3}$$

$$P_A(\overline{RTO}) = p(1-p)^2 \times [P(1) + P(2) + P(3)] + p^2(1-p) \times [2P(2) + P(3)] + p^3 P(3) + (1-p)^3 \tag{4}$$

$$P_A(RTO) = 1 - P_A(\overline{RTO}) \le \varepsilon \tag{5}$$

From (1) – (5), we can derive the optimal value for the amount of dummy packets which is represented by *m*. The value of ε is chosen as small as possible. For example, ε is set to 10^{-6} in the simulation. If *m* is larger than the total amount of feedbacks which are received by server in the trailing ACKs phase, we send a dummy packet once the server received a feedback in the trailing ACKs phase. Otherwise, we send a dummy packet for each feedback until all *m* dummy packets are sent.

4. Evaluation

Efficient bandwidth distribution and friendliness are keys to quality of experience consistency in video delivery. To verify the effectiveness of the proposed algorithms in these two aspects, simulations are conducted in this section. Two algorithms are implemented in NS2 [10] by extending the TCP implementation. Three simulation scenarios are set: HAS without cross-traffic, HAS with cross-traffic, and HAS with TCP flows.

4.1. HAS without Cross-traffic

Figure 4 shows the network topology without cross-traffic used in the simulation. The server and client denote the HTTP streaming server and client. TCP CUBIC is used in the server, because cubic is used as the default congestion control algorithm in the Linux kernel, which again is used by more than 31% of the websites whose operating system is known [11]. Switch1 is the access switch of HTTP streaming server. The switch uses Drop-tail policy which is widely deployed in current data centers and the buffer size of each port is 30Kbytes. The link between switch1 and switch2 is the bottleneck link. In order to simulate the dynamic condition of wide area network, packets are uniformly dropped with a probability of 1%. The simulation lasts 600 seconds. In Figure 4 the bandwidth (Mbps) and delay (ms) are given for the link. To evaluate the effectiveness of our algorithms, four sets of simulations were conducted with a single HTTP streaming client without cross-traffic. We vary the round trip time, chunk size, bottleneck bandwidth, and packet loss rate for each set.

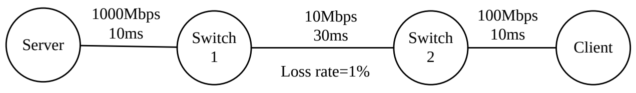

Figure 4. Network Topology without Cross-traffic

The first set of simulations compare the effectiveness of our algorithms as the round-trip time varies. We ran simulations with the RTT set to 20, 50, 80, 100, 150 and 200ms.

From the results, shown in Figure 5, we can see that HAS connection experience a drop in throughput for standard TCP and enhanced TCP. This is related to the fact that the performance penalty for dividing a video into chunks and requesting them separately. However, the throughput of enhanced TCP drops less drastically than standard TCP. This phenomenon occurs because of our enhanced algorithms. In the initial burst phase, the novel pacing algorithm smoothes the burst to reduce the packet losses. In the trailing ACKs phase, the reducing tail loss algorithm lowers the probability of RTO to reduce the download time. For the 20ms RTT case, the throughputs of standard TCP and enhanced TCP are almost the same where both equal to 9.7Mbps. Since our router buffer is sized to 30Kbytes, this phenomenon occurs because the bandwidth delay product (BDP) is smaller than the router buffer size. Then, the improvement in throughput increases along with RTT. When the RTT is 200ms, the throughput of enhanced TCP is 2.4 times larger than that of standard TCP.

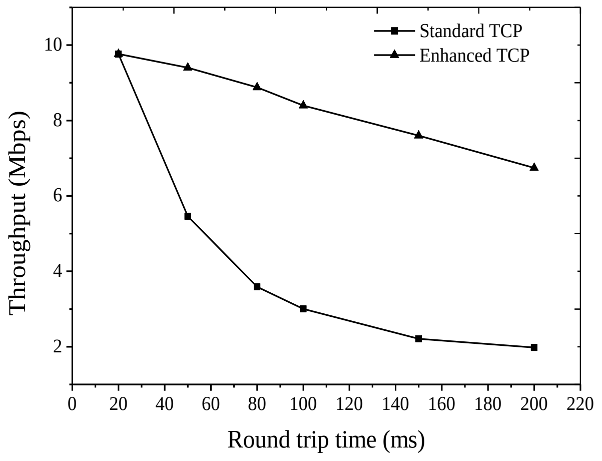

Figure 5. The Throughput with Different Round-Trip Time

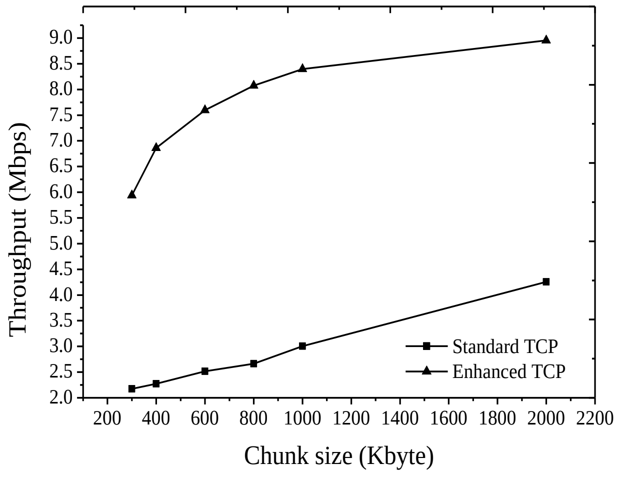

Figure 6. The Throughput with Different Chunk Size

The second set of simulations compare the effectiveness of our algorithms as the quality level of video varies. In order to check the effectiveness of our algorithms for all quality level, six chunk sizes are used to simulate the encoded video with fixed segment duration of two seconds and six quality levels. We ran simulations with the chunk size set to 300, 400, 600, 800, 1000, and 2000Kbytes.

From the results, shown in Figure 6, we can see that the throughput is higher for longer chunk size. This is related to the fact that the ACK clocking phase increases when transmitting larger chunks. In this phase, the server has filled the congestion window, and additional packets are clocked into the network by returning ACKs. Two curves in Figure 6 show that our algorithms achieve higher throughput for all quality levels than standard TCP. For the case where chunk size equals to 400, 600, and 800Kbytes, the throughput of enhanced TCP is triple high as that of standard TCP.

The third set of simulations compare the effectiveness of our algorithms as the bottleneck bandwidth varies. The bottleneck bandwidth changes from 2 to 10Mbps with a step of 2Mbps.

Figure 7 shows the throughput of HAS stream measured on a scale of percentage. The results in Figure 7 show that HAS connection also experience a drop in throughput for both standard TCP and enhanced TCP. This is the consequence of large BDP. As the bottleneck bandwidth increases, the BDP exceeds the router's buffer limit. It causes more burst traffic. Therefore, the throughput of standard TCP drops more drastically than enhanced TCP. For the 2Mbps bottleneck bandwidth case, the throughputs of standard TCP and enhanced TCP are almost the same where both equal to 97%. Since our router buffer is sized to 30Kbytes, this phenomenon occurs because the BDP is smaller than the router buffer size.

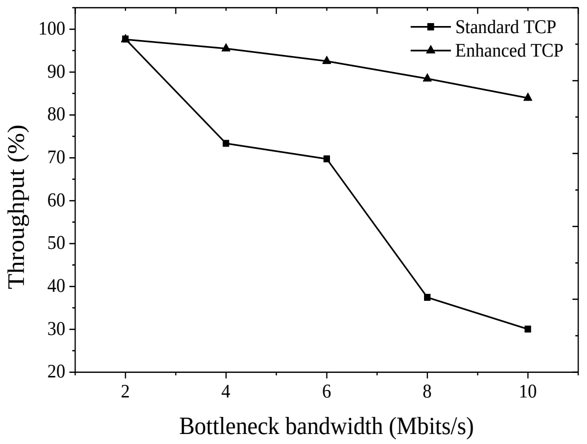

Figure 7. The Throughput with Different Bottleneck Bandwidth

The last set of simulations compare the effectiveness of our algorithms as the loss rate varies in the bottleneck link. We ran simulations with the loss rate varying from 1% to 10% with a step of 2%.

From the results shown in Figure 8, we can see that HAS connection also experience a drop in throughput for standard TCP and enhanced TCP. This is the consequence of packet loss in the bottleneck link. Although the throughput of enhanced TCP drops more drastically than standard TCP, the throughput of enhanced TCP is at least 1.5 times as high as that of standard TCP.

Above all, Figures 5-8 verify the effectiveness of our algorithms in the network without cross-traffic. The HAS stream over enhanced TCP achieves a high improvement than standard TCP in throughput.

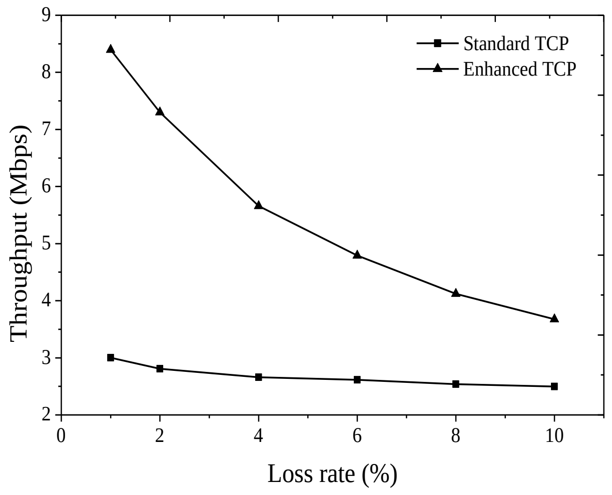

Figure 8. The Throughput with Different Packet Loss Rate in the Bottleneck Bandwidth

4.2. HAS with Cross-traffic

In this section, the simulation of topology in Figure 9 is set up. The server and client denote the HAS server and client. To simulate the varying delay and bandwidth, an exponential traffic generator (Exp_S) and receiver (Exp_R) are used as cross-traffic with average “on” time of 500ms, average “off” time of 500ms. And the average sending rate is set as 5Mbps during the “on” period. Hence the overall bit rate of the exponential traffic during the whole period is equal to 2.5Mbps. In addition to cross-traffic, a constant bit rate traffic generator (CBR_S) and receiver (CBR_R) are added. The constant bit rate (CBR) traffic is used as a competitive traffic at the bottleneck bandwidth between switch_1 and switch_2. In Figure 9 the bandwidth (Mbps) and delay (ms) are given for each link. To evaluate the effectiveness of our algorithms, two sets of simulations were conducted in the network with cross-traffic. We vary the chunk size and bitrates of competitive traffic source, *i.e.*, the CBR traffic generator.

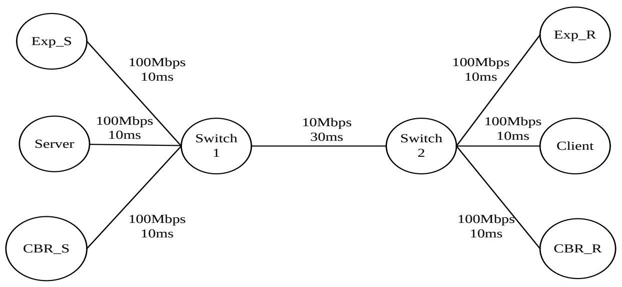

Figure 9. Network Topology with Cross-traffic

The first set of simulations compare the effectiveness of our algorithms as bitrates of competitive traffic varies. The bitrates of CBR traffic vary from 500 to 3000Kbps with a step of 500Kbps.

From the results, shown in Figure 10, we can see that HAS connection also experience a drop in throughput for standard TCP and enhanced TCP. Although the throughput of enhanced TCP drops more drastically than standard TCP, the throughput of enhanced TCP is 1.4 times as large as that of standard TCP for the case where the bitrates is 500Kbps and 1.1 times for the case where the bitrates is 3000Kbps.

The second set of simulations compare the effectiveness of our algorithms as the quality level of video varies. In order to check the effectiveness of our algorithms for all quality level, six chunk sizes are used to simulate the encoded video with fixed segment duration of two seconds and six quality levels. We ran simulations with the chunk size set to 300, 400, 600, 800, 1000, and 2000Kbyte.

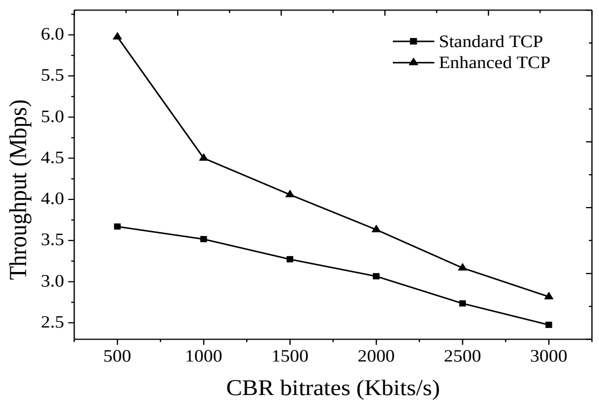

Figure 10. The Throughput with Different CBR Bitrates

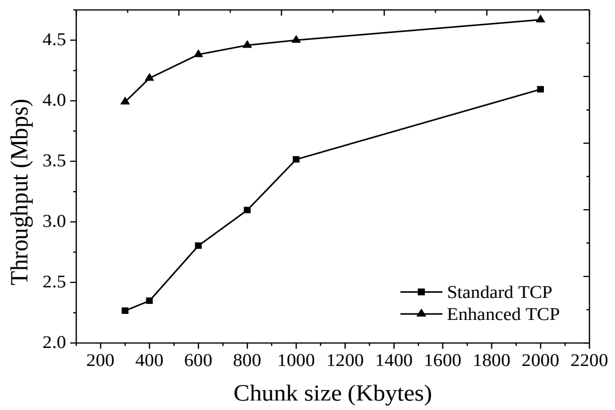

Figure 11. The Throughput for Different Quality Level

From the results, shown in Figure 11, we can see that the throughput is higher for longer chunk size. This trend is the same as a single HAS client in the network without cross-traffic. However, the throughput of enhanced TCP is 1.7 times as large as that of standard TCP for the case where the chunk size is 300Kbytes and 1.1 times for the case where the bitrates is 2000Kbytes.

Figures 10 and 11 verify the effectiveness of the proposed algorithms in the network with cross-traffic. The HAS stream over enhanced algorithms achieve higher improvement than standard TCP in throughput.

4.3. HAS with TCP Flows

To survey the friendliness between HAS stream and TCP flows, the simulation of topology with TCP flow, shown in Figure 12, is set up. Packets are uniformly dropped in the bottleneck with a probability of 1%. The server and client denote the HAS server and client. TCP and sink denote TCP source and TCP receiver. The simulation lasts 600 seconds. In Figure 12, the bandwidth (Mbps) and delay (ms) are given for each link.

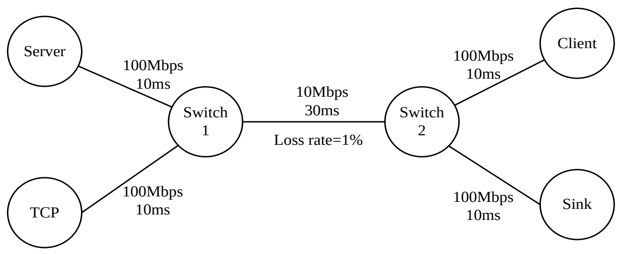

Figure 12. Network Topology with TCP Flow

Table 1 gives the throughput of HAS and TCP streams. The results in Table 1 show that HTTP streaming over enhanced TCP is more competitive than over the standard TCP. But as shown in Figure 13, throughput of HAS stream is below 5Mbps and that of TCP flow is above 5Mbps for all quality levels. It verifies that our algorithms make HTTP streaming more competitive, but keep friendliness with TCP flow at the same time.

Table 1. The Throughput of HAS Streams and TCP Flow

Chunk Size(Kbyte)	Throughput(Mbps)			
	HAS (TCP)	TCP	HAS (enhance TCP)	TCP
300	2.225	7.231	3.732	5.537
400	2.321	6.071	3.898	5.493
600	2.4	6.933	4.157	5.477
800	2.539	7.029	4.155	5.495
1000	2.76	6.898	4.41	5.357
2000	3.736	6.061	4.634	5.221

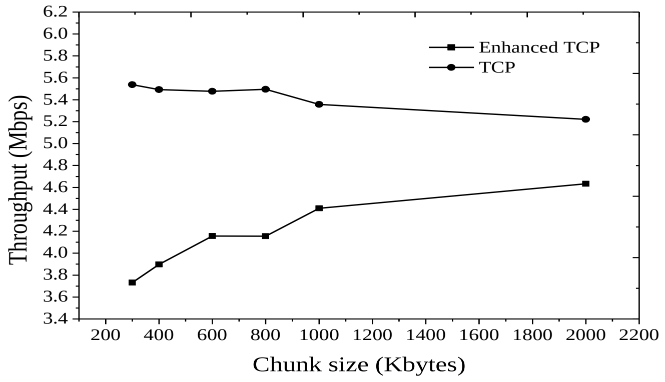

Figure 13. The Throughput of HAS Streams Over Enhanced TCP and TCP Flows

5. Conclusion

In this paper, we enumerated two reason that cause the poor performance of HAS. Two server-based algorithms are proposed targeting them. The two algorithms are novel pacing algorithm in the initial burst phase and reducing tail loss algorithm in the trailing ACKs phase. The novel pacing algorithm aims to reduce burst in the initial burst phase, and skip the slow start after retransmission timeouts occurred in the trailing ACKs phase. The reducing tail loss algorithm aims to reduce the probability of RTO in the trailing ACKs phase. We evaluated the effectiveness of these two algorithms in simulation experiments. Simulation results prove that our algorithms increase the achievable throughput of HAS streams and reduce the impact of HAS activity pattern on TCP performance. Meanwhile, they keep HAS streams friendly with other TCP flows.

References

[1] Cisco Systems, Cisco Visual networking Index: Forecast and Methodology, **(2012-2017)**, White Paper, **(2013)** May 29.

[2] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt and I. Rimac, "Interactions Between HTTP Adaptive Streaming and TCP", Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, Toronto, Canada, **(2012)** June 7-8.

[3] D. Wei, P. Cao, S. Low and Caltech EAS, "TCP pacing revisited", Proceedings of IEEE INFOCOM, Barcelona, Spain, **(2006)** April 23-29.

[4] A. Aggarwal, S. Savage and T. Anderson, "Understanding the Performance of TCP Pacing", Proceedings of IEEE INFOCOM, IEEE, vol. 3, **(2000)**, pp. 1157-1165.

[5] M. Allman, H. Balakrishnan and S. Floyd, "Enhancing TCP's loss recovery using limited transmit", RFC 3042, **(2001)** January.

[6] B. Kim, Y.-H. Kim, M.-S. Oh and J.-S. Choi, "Microscopic behaviors of TCP loss recovery using lost retransmission detection", Consumer Communications and Networking Conference, CCNC. 2005 Second IEEE, IEEE, 2005, **(2005)**, pp. 296-301.

[7] S. Akhshabi, L. Anantakrishnan, C. Dovrolis and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players", Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, ACM, **(2013)**, pp. 19-24.

[8] M. Ghobadi, Y. Cheng, A. Jain and M. Mathis, "Trickle: Rate limiting YouTube video streaming", In Proceedings of the USENIX Annual Technical Conference (ATC), **(2012)**, pp. 6.

[9] M. Handley, J. Padhye and S. Floyd, "TCP Congestion Window Validation", RFC 2861 (Experimental), **(2000)** June.

[10] Information Sciences Institute, University of Southern California. 2006. The Network Simulator – no. 2, **(2006)** July 13.

[11] W3techs, "Usage statistics and market share of unix for websites – September 2013", http://w3techs.com/technologies/details/os-unix/all/all, **(2013)**.

[12] J. Heidemann, K. Obraczka and J. Touch, "Modeling the performance of HTTP over several transport protocols", Networking, IEEE/ACM Transactions, vol. 5.5, **(1997)**, pp. 616-630.

Authors

Xinying Liu, she received her M.S. degree in Communication and Information system from Beijing University of Posts and Telecommunications in 2007. Now she is a Ph.D. candidate in the School of Information and Communication at informatics at Beijing University of Posts and Telecommunications. Her current research interests include network protocol and multimedia communication.

Aidong Men, he received his B.S. in Electronic Information Engineering (1987) and Ph.D. in Communication and Electronic System (1994) from Beijing University of Posts and Telecommunications. Now he is professor of the Multimedia Communication Center in Beijing University of Posts and Telecommunications. His current research interests include multimedia communication, digital television, and image and video signal processing.

Peng Zhang, he received his PhD in computer science and technology from Beijing University of Posts and Telecommunications in 2013. Now he is an engineer of China Telecom Corporation Limited Beijing Research Institute. His research interests include cloud computing and data center networking.