Slow Start TCP Improvements for Video Streaming Applications

Gaku Watanabe, Kazumi Kumazoe, Dirceu Cavendish, Daiki Nobayashi, Takeshi Ikenaga, Yuji Oie
Department of Computer Science and Electronics
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: {i108132g@tobata.isc, kuma@ndrc, cavendish@ndrc, nova@ecs, ike@ecs, oie@ndrc}.kyutech.ac.jp

***Abstract*—Video streaming has become the major source of Internet traffic nowadays. In addition, content delivery network providers have adopted Video over HTTP/TCP as the preferred protocol stack for video streaming. In our previous work, we have shown how TCP variants play a definite role in the quality of video experience. In this paper, we research several mechanisms within TCP slow start phase to enhance video streaming experience. We utilize network performance measurers, as well as video quality metrics, to characterize the performance and interaction between network and application layers of video streams for various network scenarios. We show that video transport performance can be enhanced with small changes in TCP Slow Start.**

Keywords*—*Video streaming; high speed networks; TCP congestion control; Packet retransmissions; Packet loss.

I. INTRODUCTION

Transmission control protocol (TCP) is the dominant transport protocol of the Internet, providing reliable data transmission for the large majority of applications. For data applications, the perceived quality of experience is the total transport time of a given file. For real time (streaming) applications, the perceived quality of experience involves not only the total transport time, but also the amount of data discarded at the client due to excessive transport delays. Transport delays depend on how TCP handles flow control and packet retransmissions. Therefore, video streaming user experience depends heavily on TCP performance. TCP protocol interacts with video application in non trivial ways. Widely used video codecs, such as H-264, use compression algorithms that result in variable bit rates along the playout time. In addition, TCP has to cope with variable network bandwidth along the transmission path. Network bandwidth variability is particularly wide over paths with wireless access links of today, where multiple transmission modes are used to maintain steady packet error rate under varying interference conditions. As the video playout rate and network bandwidth are independent, it is the task of the transport protocol to provide a timely delivery of video data so as to support a smooth playout experience.

In the last decade, many TCP variants have been proposed, mainly motivated by performance reasons. As TCP performance depends on network characteristics, and the Internet keeps evolving, TCP variants are likely to continue to be proposed. Most of the proposals deal with congestion window size adjustment mechanism, which is called congestion avoidance phase of TCP, since congestion window size controls the amount of data injected into the network at a given time. In prior work, we have introduced a delay based TCP window flow control mechanism that uses path capacity and storage estimation [5] [6]. The idea is to estimate bottleneck capacity and path storage space, and regulate the congestion window size using a control theoretical approach. Two versions of this mechanism were proposed: one using a proportional controlling equation [5], and another using a proportional plus derivative controller [6]. More recently, we have studied TCP performance of most popular TCP variants - Reno [2], Cubic (Linux) [13], Compound (Windows) [14] - as well as our proposed TCP variants: Capacity and Congestion Probing (CCP) [5], and Capacity Congestion Plus Derivative (CCPD) [6], in transmitting video streaming data over wireless path conditions. Our proposed CCP and CCPD TCP variants utilize delay based congestion control mechanism, and hence are resistant to random packet losses experienced in wireless links.

In this paper, we show that it is possible to improve Slow Start mechanism of TCP to improve video streaming over Internet paths with wireless access links. More specifically, we demonstrate that: i) Open loop nature of slow start negatively affects video rendering quality; ii) Dampening congestion window growth during slow start may negatively affect video streaming performance; iii) Shortening slow start phase may improve video performance. The material is organized as follows. Related work discussion is provided on Section II. Section III describes video streaming over TCP system. Section IV introduces the TCP variants addressed in this paper, as well as additional mechanisms to enhance video streaming experience. Section VI addresses video delivery performance evaluation for each TCP variant and attempted enhancements. Section VII addresses directions we are pursuing as follow up to this work.

II. RELATED WORK

The impact of wide variability of TCP throughput caused by network packet losses on video streaming has been addressed [10] [4]. In [10], variable rate video encoders are considered, where video source adjusts its encoding rate according with network available bandwidth in the streaming path. In [4], a TCP Reno delay model is used by the video encoder to change encoding mode according with network conditions. Both approaches require a tight coupling between application and transport protocol. These approaches are opposite to what is taken in this work, which seeks to adjust video transport to arbitrary video encoders.

Modifications of TCP protocol to enhance video streaming have been recently proposed. Pu et al. [12] have proposed a proxy TCP architecture for higher performance on paths with last hop wireless links. The proxy TCP node implements a

 ISBN: 978-1-61208-349-0

variation of TCP for which additive increase multiplicative decrease (AIMD) congestion window $cwnd$ adjustment is disabled, and replaced with a fair scheduler at the entrance of the wireless link. The approach, however, does not touch TCP sender at the video server side, which limits overall video streaming performance as characterized in [8].

A different approach to improving the transport of video streams is presented by [11]. Their work seeks to improve video streaming performance by streaming over multiple paths, as well as adapting video transmission rates to the network bandwidth available. Such approach, best suited to distributed content delivery systems, requires coordination between multiple distribution sites. In contrast, we seek to improve each network transport session carrying a video session by adapting TCP source behavior, independently of the video encoder.

Another distinct aspect of our current work is that we propose improvements on Slow Start, which is widely used by many TCP variants, on real client and server network stacks that are widely deployed for video streaming, via VLC open source video client, and standard HTTP server. We seek to understand how small changes in TCP Slow Start may affect the quality of user experience.

III. Video Streaming over TCP

Video streaming over HTTP/TCP involves an HTTP server side, where video files are made available for streaming upon HTTP requests, and a video client, which places HTTP requests to the server over the Internet, for video streaming. Fig. 1 illustrates video streaming components.

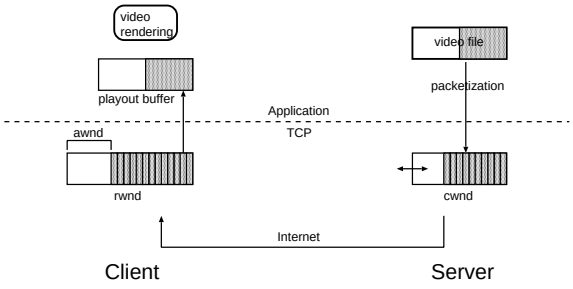


Fig. 1: Video Streaming over TCP

An HTTP server stores encoded video files, available upon HTTP requests. Once a request is placed, a TCP sender is instantiated to transmit packetized data to the client machine. At TCP transport layer, a congestion window is used for flow controlling the amount of data injected into the network. The size of the congestion window, $cwnd$, is adjusted dynamically, according to the level of congestion in the network, as well as the space available for data storage, $awnd$, at the TCP client receiver buffer. Congestion window space is freed only when data packets are acknowledged by the receiver, so that lost packets are retransmitted by the TCP layer. At the client side, in addition to acknowledging arriving packets, TCP receiver sends back its current available space $awnd$, so that at the sender side, $cwnd \leq awnd$ at all times. At the client application layer, a video player extracts data from a playout buffer, filled with packets delivered by TCP receiver from its buffer. The playout buffer is used to smooth out variable data arrival rate.

A. Interaction between Video streaming and TCP

At the server side, HTTP server retrieves data into the TCP sender buffer according with $cwnd$ size. Hence, the injection rate of video data into the TCP buffer is different than the video variable encoding rate. In addition, TCP throughput performance is affected by the round trip time of the TCP session. This is a direct consequence of the congestion window mechanism of TCP, where only up to a $cwnd$ worth of bytes can be delivered without acknowledgements. Hence, for a fixed $cwnd$ size, from the sending of the first packet until the first acknowledgement arrives, a TCP session throughput is capped at $cwnd/rtt$. For each TCP Slow Start variant, to be described shortly, the size of the congestion window is computed by a specific algorithm at time of packet acknowledgement reception by the TCP source. However, for all TCP variants, the size of the congestion window is capped by the available TCP receiver space $awnd$ sent back from the TCP client.

At the client side, the video data is retrieved by the video player into a playout buffer, and delivered to the video renderer. Playout buffer may underflow, if TCP receiver window empties out. On the other hand, playout buffer overflow does not occur, since the player will not pull more data into the playout buffer than it can handle.

In summary, video data packets are injected into the network only if space is available at the TCP congestion window. Arriving packets at the client are stored at the TCP receiver buffer, and extracted by the video playout client at the video nominal playout rate.

IV. Anatomy of transmission control protocol

TCP protocols fall into two categories, delay and loss based. Advanced loss based TCP protocols use packet loss as primary congestion indication signal, performing window regulation as $cwnd_k = f(cwnd_{k-1})$, being ack reception paced. Most f functions follow an Additive Increase Multiplicative Decrease strategy, with various increase and decrease parameters. TCP NewReno and Cubic are examples of AIMD strategies. Delay based TCP protocols, on the other hand, use queue delay information as the congestion indication signal, increasing/decreasing the window if the delay is small/large, respectively. Vegas, CCP and CCPD are examples of delay based protocols.

Most TCP variants follow TCP Reno phase framework: slow start, congestion avoidance, fast retransmit, and fast recovery.

- **Slow Start(SS) :** This is the initial phase of a TCP session, where no information about the session path is assumed. In this phase, for each acknowledgement received, two more packets are allowed into the network. Hence, congestion window $cwnd$ is roughly doubled at each round trip time. Notice that the $cwnd$ size can only increase in this phase. Therefore, in this phase there is no flow control of the traffic into the network. This phase ends when the $cwnd$ size reaches a large value, dictated

 ISBN: 978-1-61208-349-0

by $ssthresh$ parameter, or when the first packet loss is detected, whichever comes first. All widely used TCP variants make use of the same slow start except Cubic [13].

- **Congestion Avoidance(CA) :** This phase is entered when the TCP sender detects a packet loss, or the $cwnd$ size reaches a target upper size called $ssthresh$ (slow start threshold). The sender controls the $cwnd$ size to avoid path congestion. Each TCP variant has a different method of $cwnd$ size adjustment.
- **Fast Retransmit and fast recovery(FR) :** The purpose of this phase is to freeze all $cwnd$ size adjustments in order to take care of retransmissions of lost packets.

Fig. 2 illustrates various phases of a TCP session. Our interest is in the Slow Start phase of TCP, at the left of the figure, which dictates how much traffic is allowed into the network before congestion avoidance takes place. A comprehensive tutorial of TCP features can be found in [1].

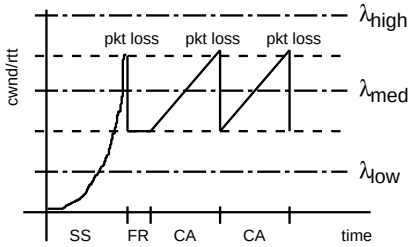


Fig. 2: TCP Congestion Window Dynamics vs Video Playout

For most TCP variants widely used today, with exception of Cubic, the slow start phase may negatively impact video experience, for two reasons: i) the amount of traffic injected into the network path is independent of the network path conditions, as well as the video playout rate; ii) Slow start phase typically ends with a large $cwnd$ size, which causes multiple packet losses, and may trigger long fast retransmit/recovery periods. During these periods, no new packets are admitted into the network until all lost packets are successfully delivered. For video streams, many of these lost packets may arrive too late for frame rendering, causing excessive frame discards. Finally, a responsive congestion avoidance mechanism affords quick adaptation to network conditions, which decreases playout buffer underflows as well as picture discards.

In this paper, we use CCP as a framework upon which we design Slow Start variation mechanisms. TCP CCP was our first attempt to design a delay based congestion avoidance scheme based on solid control theoretical approach. The cwnd size is adjusted according to a proportional controller control law. The cwnd adjustment scheme is called at every acknowledgement reception, and may result in either window increase and decrease. In addition, packet loss does not trigger any special cwnd adjustment. CCP cwnd adjustment scheme is as per 1:

$$cwnd_k = \frac{[Kp(B - x_k) - in\_flight\_segs_k]}{2} \quad 0 \leq Kp \qquad (1)$$

where Kp is a proportional gain, B is an estimated storage capacity of the TCP session path, or virtual buffer size, x_k is the level of occupancy of the virtual buffer, or estimated packet backlog, and in_flight_segs is the number of segments in flight (unacknowledged). Typically, CCP cwnd dynamics exhibit a dampened oscillation towards a given cwnd size, upon cross traffic activity. Notice that $cwnd_k$ does not depend on previous cwnd sizes, as with the other TCP variants. This fact guarantees a fast responsiveness to network bandwidth variations.

As CCP uses the same Slow Start mechanism as of most TCP variants widely deployed nowadays, all lessons learn on this paper are applicable to other TCP variants.

Let λ be the video average bit rate across its entire playout time. That is, $\lambda = VideoSize/TotalPlayoutTime$. Fig. 2 illustrates three video playout rate cases: $\lambda_{high}, \lambda_{med}, \lambda_{low}$:

λ_{high} The average playout rate is higher than the transmission rate. In this case, playout buffer is likely to empty out, causing buffer underflow condition.

λ_{med} The average playout rate is close to the average transmission rate. In this case, buffer underflow is not likely to occur, affording a smooth video rendering at the client.

λ_{low} The average playout rate is lower than the transmission rate. In this case, playout buffer may overflow, causing picture discards due to overflow condition. In practice, this case does not happen if video client pulls data from the TCP socket, as it is commonly the case. In addition, TCP receiver buffer will not overflow either, because $cwnd$ at the sender side is capped by the available TCP receiver buffer space $awnd$ reported by the receiver.

V. TCP Slow Start Improvements for Video Streaming

We focus on the slow start phase, since as mentioned before, it is a phase at which much harm can be done to video streaming due to its open feedback nature. The original idea of Slow Start was to increase $cwnd$ as quickly as possible to large values, so that more data throughput could be achieved on a short period of time. However, for video applications, the ideal throughput should be close to the video rendering rate. So, there is no use in targeting too high throughput. For these changes, we target our CCP TCP variant, since we have control over its implementation.

- **ShortSlowStart:** In this scheme, our TCP variant (CCPSSS) in slow start attempts to transition into congestion avoidance as quickly as possible. However, because CCP requires estimation of network path capacity in congestion avoidance phase, the scheme waits until a first capacity estimate is available to transition out of slow start.
- **VideoRateStart:** In this scheme, TCP variant (CCPVS) in slow start attempts to set its $cwnd$ to a size such that its resulting throughput ($cwnd/rtt$) be close to the video average playout rate. Hence, $cwnd$ is expected to stabilize on a given value, during Slow Start.
- **VideoRateSsthresh:** In this scheme (CCPLSS), $ssthresh$ parameter, which caps the maximum value of the $cwnd$ before TCP leaves slow start and transitions

into congestion avoidance, is adjusted to correspond to the average video playout rate. That is, the scheme delivers a throughput around the video playout rate at the time of transition out of slow start.

VI. Video Streaming Performance Characterization for various Slow Starts

Fig. 3 describes the network testbed used for emulating a network path with wireless access link. An HTTP video server and a VLC client machine are connected to two access switches, which are connected to a link emulator, used to adjust path delay and inject controlled random packet loss. All links are 1Gbps, ensuring plenty of network capacity for many video streams between client and server. No cross traffic is considered, as this would make it difficult to isolate the impact of TCP Slow Start mechanisms on video streaming performance.

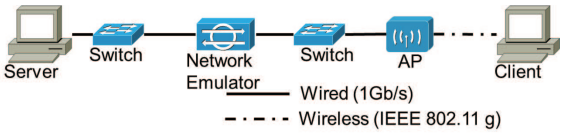


Fig. 3: Video Streaming Emulation Network

Video and network settings are as follows: video file size: $409Mbytes$; Playback time: $10min24sec$; Average playback rate: $5.24Mbps$; Encoding: MPEG-4; video codec: H.264/AVC; frame rate: 30fps; audio codec: MPEG-4 AAC; playout buffer size: $656Kbytes$, which drives initial buffering delay. TCP sender and receiver maximum buffer size: $256Mbytes$. The VLC client is attached to the network via a WiFi link. Iperf is used to measure the available wireless link bandwidth, to make sure it is higher than the average video playout rate. Packet loss is hence induced only by the wireless link, and is reflected in the number of TCP packet retransmissions.

Performance measurers adopted, in order of priority, are:

- **Picture discards:** number of frames discarded by the video decoder. This measurer defines the number of frames skipped by the video rendered at the client side.
- **Buffer underflow:** number of buffer underflow events at video client buffer. This measurer defines the number of "catch up" events, where the video freezes and then resumes at a faster rate until all late frames have been played out.
- **Packet retransmissions:** number of packets retransmitted by TCP. This is a measure of how efficient the TCP variant is in transporting the video stream data. It is likely to impact video quality in large round trip time path conditions, where a single retransmission doubles network latency of packet data from an application perspective.

We organize our test cases into the following categories:

- Typical round trip time
- Short round trip time
- Large round trip time

For each of these categories, we have run five trial experiments for each TCP Slow Start variant. Results are reported as average and standard deviation bars.

A. Typical round trip time

Fig. 4 zooms into the first 30 secs of video streaming, to highlight $cwnd$ dynamics during slow start (X-axis in units of 100msecs). CCP(1) shows a standard $cwnd$ ramp up (first 2 secs of session), where $cwnd$ size is doubled at every round trip time. CCPLSS also shows a similar $cwnd$ ramp up, which is because the only difference is the $ssthresh$ value used. In contrast, CCPVS and CCPSSS rapidly ramp up their $cwnd$ size to a high value. CCPSSS shows oscillations on $cwnd$ size due to early poor capacity estimation.

Fig. 5 reports on bottleneck capacity estimation of each TCP variant. The graphs show that there is no difference in capacity estimation across the TCP variants. This is not surprising, given that the capacity estimation method, based on a packet pair dispersion measurement, is the same across all variants. Each TCP session sees around 15 Mbps bottleneck capacity, which in our experiment topology corresponds to the available bandwidth of the wireless WiFi access link.

Fig. 6 reports on the average packet round trip time seen by each TCP variant along the video streaming session. The graphs' dynamics has a 100msec support line, which is the round trip propagation delay between video source and client. In addition, the higher and more dense these graphs are, the more aggressive the TCP variant is. All variants present a highly dynamic rtt variation, which poses a challenge in video rendering from the video playout buffer.

Fig. 7 depicts video streaming and TCP performance under typical propagation delay of 100msecs. In this case, CCP(1) delivers best video experience, followed by CCPLSS. All variants feature small to negligible packet retransmissions except CCP(1), which is evidence of how vanilla Slow Start TCP hurts video streaming.

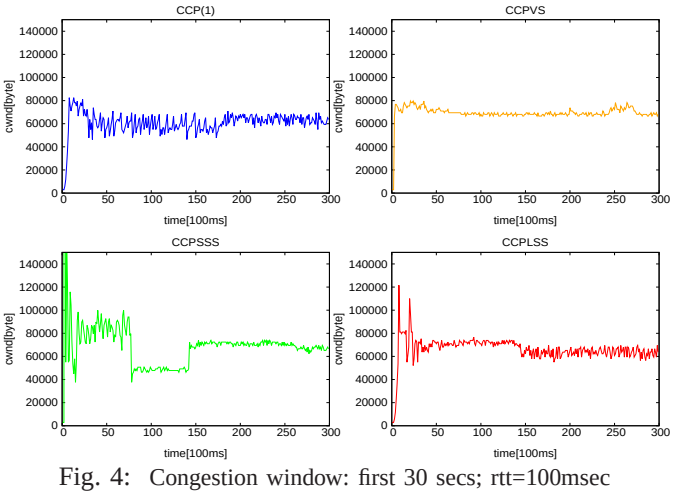


Fig. 4: Congestion window: first 30 secs; rtt=100msec

B. Short round trip time

In this experimental settings, VLC client and server are connected via a very short path, with propagation delay of

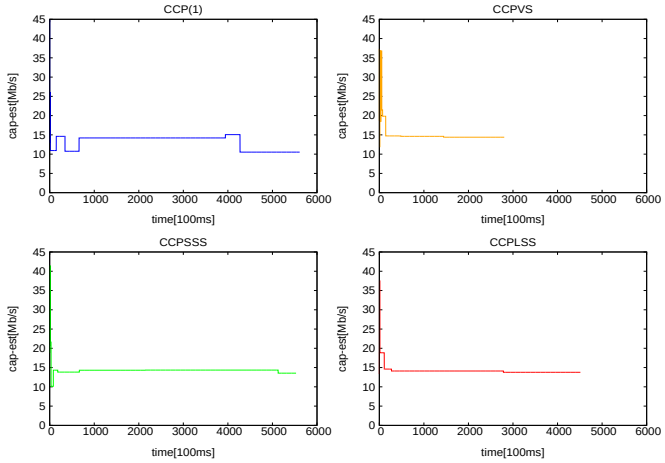


Fig. 5: Capacity estimation; rtt=100msec

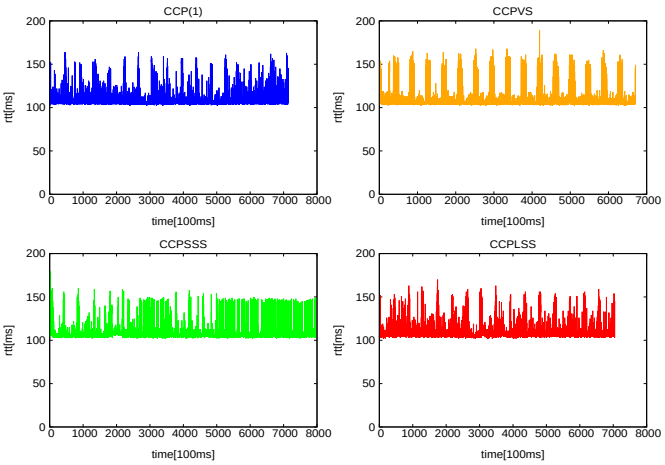
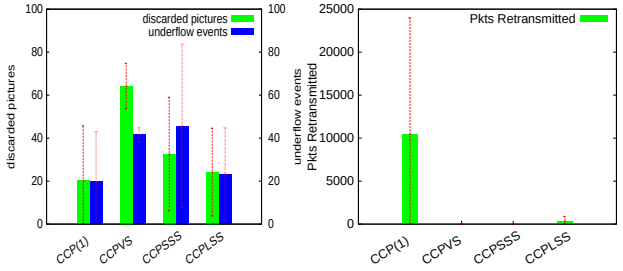


Fig. 6: Packet delay; rtt=100msec

around 3msec. Fig. 8 depicts the $cwnd$ evolution of the first 30 secs of a video experiment trial. For all variants, we can see various Slow Start periods, where $cwnd$ returns to small values and ramps up again. This is because the network path has very small space for packet storage, hence most congestion may be considered severe, which causes the TCP session to return to Slow Start phase.

Fig. 9 depicts packet round trip times measured between TCP sender and receiver. We first notice that this measurement includes retransmission at the network and WiFi link layers. The much inflated round trip time values measured as compared with propagation delay of around 3msec points to a large number of packet retransmissions, which is expected under short network path storage space. CCP(1) is clearly the most aggressive TCP variant, incurring in more packet delays than the other variants. CCPLSS seems to present a compromise between low and high round trip times. Notice that a constantly low rtt is not necessarily good, as it may indicate that the TCP variant is not responsive to network load and bottleneck variations throughout the streaming session.

Fig. 10 shows the video and TCP performance for this short rtt propagation delay scenario. In this case, CCPLSS delivers



a) VLC performance b) TCP packets retransmitted

Fig. 7: Video Performance vs TCP performance; rtt=100msec

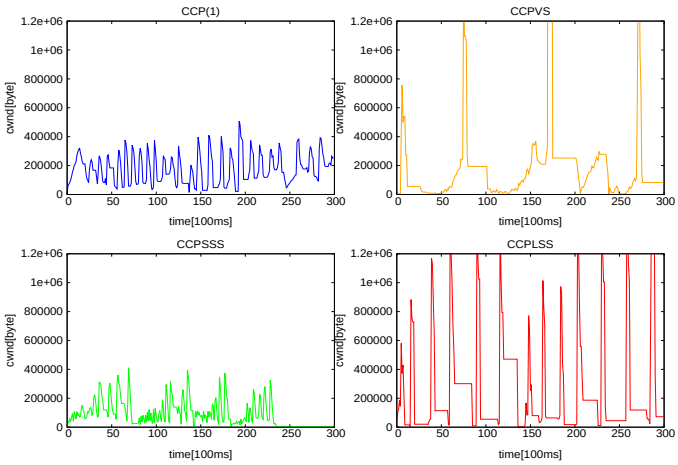


Fig. 8: Congestion window: first 30 secs; rtt=3msec

best video streaming quality, with least number of playout buffer underflows, as well as the least number of discarded pictures. Performance of all TCP variants are sharply different because of the repeated return of TCP session to Slow Start, due to the short network path (small storage space).

C. Large round trip time

Fig. 11 zooms into the first 30 secs of video streaming, for the case of long round trip propagation delay. Notice that the $cwnd$ oscillations are larger, which is expected for feedback loops with large dead time (propagation delay). Despite large $cwnd$ oscillations, packet delays, not shown for space considerations, are more flat, due to more storage capacity in the long network path.

Finally, Fig. 12 reports VLC and TCP performance for long propagation delay. This long path case confirms that for high path storage conditions CCP(1) delivers best video streaming performance. This is evidence that aggressive TCP protocols deliver better video streaming performance when network path has enough storage space.

In our performance evaluation, we have not attempted to tune VLC client to minimize frame discards, even though VLC settings may be used to lower the number of frame discards. In addition, no tuning of TCP parameters was performed to better video client performance for any of the TCP variants studied. We have simply used parameter values from our previous study of CCP performance of file transfers [7]. All changes were limited to the Slow Start phase of TCP.

 ISBN: 978-1-61208-349-0

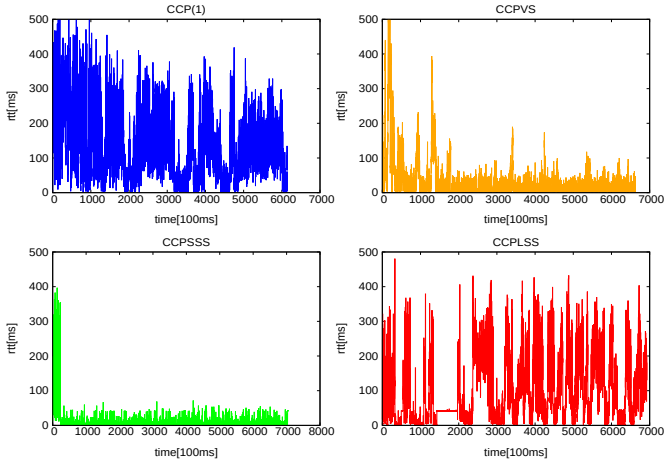


Fig. 9: Packet delay; rtt=3msec

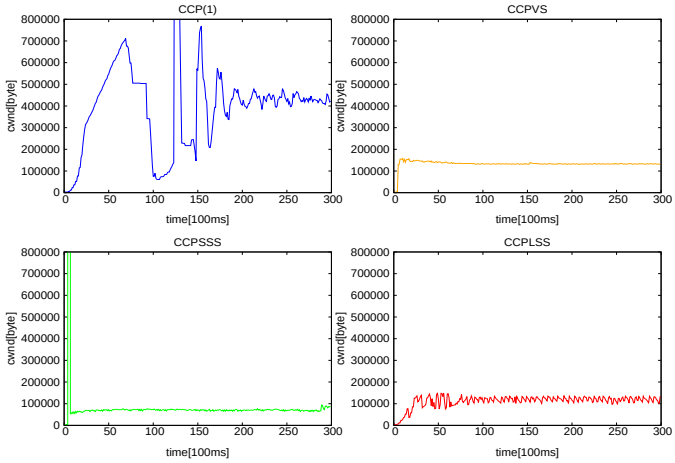
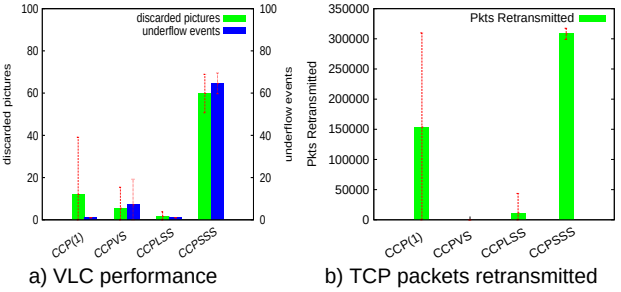
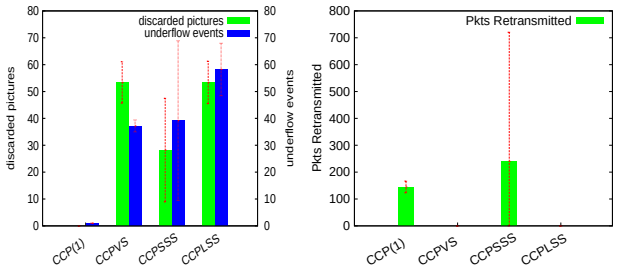


Fig. 11: Congestion window: first 30 secs; rtt=200msec



a) VLC performance b) TCP packets retransmitted

Fig. 10: Video Performance vs TCP performance; rtt=3msec



a) VLC performance b) TCP packets retransmitted

Fig. 12: Video Performance vs TCP performance; rtt=200msec

VII. Conclusion and Future Work

In this paper, we have introduced and evaluated a few variations of the slow start phase of TCP protocol to improve TCP transport performance of video streams. We have characterized TCP performance with these schemes when transporting video streaming applications over wireless network paths via open source experiments. Our experimental results show that tuning $ssthresh$ parameter to deliver video playout rate throughput when TCP transitions out of Slow Start delivers better Video Streaming performance on short network paths. As future work, we are currently exploring changes in the congestion avoidance phase of TCP, in order to further improve video streaming.

Acknowledgments

Work supported in part by JSPS Grant-in-Aid for Scientific Research (B) (No 23300028).

References

[1] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP, ” IEEE Communications Surveys & Tutorials, Third Quarter 2010, Vol. 12, No. 3, pp. 304-342.

[2] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control,” IETF RFC 2581, April 1999.

[3] A. Ahmed, S.M.H. Zaidi, and N. Ahmed, “Performance evaluation of Transmission Control Protocol in mobile ad hoc networks, ” IEEE Int. Networking and Communication Conference, June 2004, pp. 13-18.

[4] A. Argyriou, “Using Rate-Distortion Metrics for Real-Time Internet Video Streaming with TCP, ” IEEE ICME06, 2006, pp. 1517-1520.

[5] D. Cavendish, K. Kumazoe, M. Tsuru, Y. Oie, and M. Gerla, “Capacity and Congestion Probing: TCP Congestion Avoidance via Path Capacity and Storage Estimation,” IEEE Second International Conference on Evolving Internet, best paper award, September 2010, pp. 42-48.

[6] D. Cavendish, H. Kuwahara, K. Kumazoe, M. Tsuru, and Y. Oie, “TCP Congestion Avoidance using Proportional plus Derivative Control,” IARIA Third International Conference on Evolving Internet, best paper award, June 2011, pp. 20-25.

[7] D. Cavendish et al., “On Tuning TCP for Superior Performance on High Speed Path Scenarios,” IARIA Fourth International Conference on Evolving Internet, best paper award, June 2012, pp. 11-16.

[8] G. Watanabe, K. Kumazoe, D. Cavendish, D. Nobayashi, T. Ikenaga, and Y. Oie, “Performance Characterization of Streaming Video over TCP Variants,” IARIA Fifth International Conference on Evolving Internet, best paper award, June 2013, pp. 16-21.

[9] S. Henna,“A Throughput Analysis of TCP Variants in Mobile Wireless Networks,” Third Int. Conference on Next Generation Mobile Applications, Services and Technologies - NGMAST, Sept. 2009, pp.279-284.

[10] P. Papadimitriou, “An Integrated Smooth Transmission Control and Temporal Scaling Scheme for MPEG-4 Streaming Video,” In Proceedings of IEEE ICME 08, 2008, pp. 33-36.

[11] J-W. Park, R. P. Karrer, and J. Kim,, “TCP-RomeL A Transport-Layer Parallel Streaming Protocol for Real-Time Online Multimedia Environments,” In Journal of Communications and Networks, Vol.13, No. 3, June 2011, pp. 277-285.

[12] W. Pu, Z. Zou, and C. W. Chen, “New TCP Video Streaming Proxy Design for Last-Hop Wireless Networks,” In Proceedings of IEEE ICIP 11, 2011, pp. 2225-2228.

[13] I. Rhee, L. Xu, and S. Ha, “CUBIC for Fast Long-Distance Networks,” Internet Draft, draft-rhee-tcpm-ctcp-02, August 2008.

[14] M. Sridharan, K. Tan, D. Bansal, and D. Thaler, “Compound TCP: A New Congestion Control for High-Speed and Long Distance Networks,” Internet Draft, draft-sridharan-tcpm-ctcp-02, November 2008.

[15] S. Waghmare, A. Parab, P. Nikose, and S.J. Bhosale, “Comparative analysis of different TCP variants in a wireless environment,” IEEE 3rd Int. Conference on Electronics Computer Technology, April 2011, Vol.4, pp.158-162.

 ISBN: 978-1-61208-349-0