Performance Characterization of Streaming Video over TCP Variants

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. In addition, content delivery network providers have adopted Video over HTTP/TCP as the preferred protocol stack for video streaming. In this paper, we characterize the performance of various TCP variants when transporting video traffic over various network scenarios. 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 no widely deployed TCP variant is able to deliver best performance across all scenarios evaluated.**

***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. 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 these two bit rates 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 [6], [7]. 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 [6], and another using a proportional plus derivative controller [7]. In this work, we study TCP performance of most popular TCP variants - Reno [2], Cubic (Linux) [11], Compound (Windows) [12] - as well as our most recently proposed TCP variants: Capacity and Congestion Probing (CCP) [6], and Capacity Congestion Plus Derivative (CCPD) [7], in transmitting video streaming data over wireless path conditions. The motivation for including our proposed TCP variants is that CCP and CCPD utilize delay based congestion control mechanism, and hence are resistant to random packet losses experienced in wireless links.

Our contributions are as follows. We show that most used TCP variants of today affect video quality differently over various network scenarios. Our results show that there is no single TCP variant that is able to best deliver video streams under all network scenarios. 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, their features and differences. Section V addresses video delivery performance evaluation for each TCP protocol. Section VI addresses directions we are pursuing as follow up to this work.

II. RELATED WORK

Research studies of TCP performance on wireless network environments abound. Many of these studies [4], [9], [13] focus on the issue of loss based TCP not being able to differentiate between random packet loss and buffer overflow packet loss [3]. In [4], throughput performance of TCP variants for various Packet Error Rates (PERs) on a mobile network is studied via simulations. In [9], TCP variants performance under various PERs is also studied, including investigation of the impact of routing protocols on TCP performance. Wireless network scenarios typically involve a low speed bottleneck link capacity, which limits the size of the congestion window to small values, masking the buffer overflow problem on routers. In our work, we study the impact of network random losses on video streaming.

Recently, the impact of wide variability of TCP throughput caused by network packet losses on video streaming has been addressed [5], [10]. 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 [5], 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. In contrast, our client video source and client are “loosely” coupled with TCP stack.

Another distinct aspect of our current work is that we analyze performance of widely used TCP variants, as well as our proposed delay based TCPs, CCP and CCPD, on real client

 ISBN: 978-1-61208-285-1

and server network stacks that are widely deployed for video streaming, via VLC open source video client, and standard HTTP server. As TCP variants have different dynamics when facing random losses, we seek to understand whether there are better TCP variants for video streaming, without having to tightly couple transport layer with video server/client.

III. Anatomy of 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.

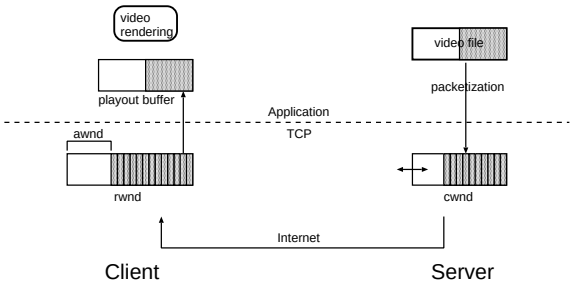


Fig. 1: Video Streaming over TCP

An HTTP server stores encoded video files, available upon HTTP request. 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 $cwnd \leq awnd$ at all times. At the client application layer, a video player extracts data from TCP receiver buffer into a playout buffer, 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 the $cwnd$ size. Hence, in case of HTTP server, the injection of video data into the TCP buffer is unrelated to 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 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 pulled 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. Transmission control protocol variants

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. We have not included Vegas on our study because Vegas performance is not competitive against well established TCP variants [6].

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. In this paper, all TCP variants make use of the same slow start except Cubic [11].
- **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.

Figure 2 illustrates various phases of a TCP session. A comprehensive tutorial of TCP features can be found in [1].

A. Reno TCP

Reno is a loss based TCP, and may be considered the oldest implementation of TCP to achieve widespread usage. Its congestion avoidance scheme relies on increasing the cwnd by $1/cwnd$ increments, and cutting its current size in half on packet loss detection, as per equation 1.

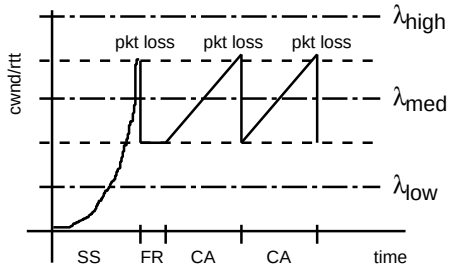


Fig. 2: TCP Congestion Window Dynamics vs Video Playout

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k} \\ PktLoss: \quad cwnd_{k+1} &= \frac{cwnd_k}{2} \end{aligned} \tag{1}$$

Notice that for large cwnd values, the increment becomes small. So, for large bandwidth delay product paths, Reno cwnd ramps up very slowly. A new version of Reno, TCP NewReno introduces an optimization of the Fast Recovery mechanism, but its congestion avoidance scheme remains the same.

B. Cubic TCP

TCP Cubic is a loss based TCP that has achieved widespread usage as the default TCP of the Linux operating system. Its congestion window adjustment scheme is:

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= C(t-K)^3 + Wmax \\ K &= (Wmax\frac{\beta}{C})^{1/3} \\ PktLoss: \quad cwnd_{k+1} &= \beta cwnd_k \\ Wmax &= cwnd_k \end{aligned} \tag{2}$$

where C is a scaling factor, Wmax is the cwnd value at time of packet loss detection, and t is the elapsed time since the last packet loss detection (cwnd reduction). The rational for these equations is simple. Cubic remembers the cwnd value at time of packet loss detection - Wmax, when a sharp cwnd reduction is enacted, tuned by parameter β. After that, cwnd is increased according to a cubic function, whose speed of increase is dictated by two factors: i) how long it has been since the previous packet loss detection, the longer the faster ramp up; ii) how large the cwnd size was at time of packet loss detection, the smaller the faster ramp up. The shape of Cubic cwnd dynamics is typically distinctive, clearly showing its cubic nature. Notice that upon random loss, Cubic strives to return cwnd to the value it had prior to loss detection quickly, for small cwnd sizes.

C. Compound TCP

Compound TCP is the TCP of choice for most Wintel machines. It implements a hybrid loss/delay based congestion avoidance scheme, by adding a delay congestion window dwnd to the congestion window of NewReno [12]. Compound TCP cwnd adjustment is as per Equation 3:

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k + dwnd_k} \\ PktLoss: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k} \end{aligned} \tag{3}$$

where the delay component is computed as:

$$\begin{aligned} AckRec: dwnd_{k+1} &= dwnd_k + \alpha dwnd_k^K - 1, \text{ if } diff < \gamma \\ & \quad dwnd_k - \eta diff, \quad \text{if } diff \geq \gamma \\ PktLoss: dwnd_{k+1} &= dwnd_k(1-\beta) - \frac{cwnd_k}{2} \end{aligned} \tag{4}$$

where α, β, η and K parameters are chosen as a tradeoff between responsiveness, smoothness, and scalability.

D. Capacity and Congestion Probing TCP

TCP CCP is 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 Equation 5:

$$cwnd_k = \frac{[Kp(B - x_k) - in\_flight\_segs_k]}{2} \quad 0 \leq Kp \tag{5}$$

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.

E. Capacity and Congestion Plus Derivative TCP

TCP CCPD is our second attempt to design a delay based congestion avoidance scheme based on solid control theoretical approach, being a variant of CCP. The scheme cwnd adjustment follows the same strategy of CCP. The difference is that it uses a proportional plus derivative controller as its control equation. CCPD cwnd adjustment scheme is as per Equation 6:

$$cwnd_k = Kp[B - x_k - in\_flight\_segs_k] + \tag{6}$$

$$\frac{Kd}{t_k - t_{k-1}}[x_{k-1} + in\_flight\_segs_{k-1} + -x_k - in\_flight\_segs_k] \tag{7}$$

where Kp is a proportional gain, Kd is a derivative gain, and the other parameters are defined as per CCP congestion avoidance scheme. Typically, CCPD cwnd dynamics present similar dampened oscillatory behavior as CCP, with a much faster period, due to its reaction to the derivative or variation of the number of packets backlogged.

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: $\lambda_{high}, \lambda_{med}, \lambda_{low}$:

 ISBN: 978-1-61208-285-1

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. Video Streaming Performance Characterization over TCP Variants

Figure 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 variants on video streaming performance. An extended version of this paper is planned to include multiple video stream experiments.

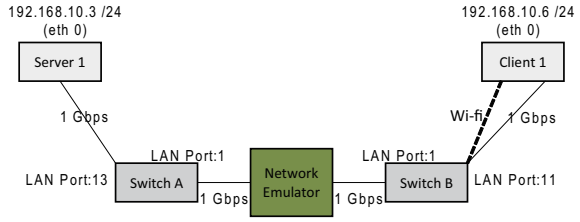


Fig. 3: Video Streaming Emulation Network

Video and network settings are as follows: video file size: $409Mbytes$; Playback time: $10min24sec$; Average playback rate: $5.24Mbps$; Encoding: MPEG-4;[1] video codec: H.264/AVC; frame rate: 30fps; audio codec: MPEG-4 AAC; playout buffer size: $656Kbytes$. TCP sender and receiver maximum buffer size: $256Mbytes$.

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 retransmission doubles network latency of packet data from an application perspective.

In the TCP variant performance comparison study that follows, no attempt was made to tune TCP parameters to best video streaming performance. In particular, for CCP(x), where x is Kp parameter of Eq. 5, and CCPD(x,y), where x and y are Kp and Kd parameters of Eq. 6, the parameters used were derived from [8], tuned to provide best file transfer performance, not video streaming, for a fair comparison with the other TCP variants.

We organize our test cases into the following categories:

- Network bandwidth smaller than video playout rate
- Network bandwidth larger than video playout rate
- Network bandwidth much larger than video playout rate
- Wifi access link scenario

For each of these categories, we have run ten trial experiments for each TCP variant with and without random packet losses, and various round trip times. Results are reported as average and standard deviation bars.

A. Network bandwidth smaller than video playout rate

Fig. 4 summarizes performance measurers when the network emulator is set to throttle network bandwidth to a value slightly lower than video nominal playout rate, when the video server and client are far apart (100msec rtt). In this case, Cubic is the TCP variant with least picture discards and playout buffer underflow events, event though it presents the largest number of packet retransmits. The high number of packet retransmits attests the aggressive behavior of Cubic in ramping up its congestion window, as illustrated in Fig. 5. A side effect of this aggressiveness is a lower number of playout buffer underflow events. Reno and Compound present the largest number of picture discards, which can be traced to their lack of aggressiveness, attested by their low number of packet retransmissions. Reno is the least aggressive TCP variant in ramping up $cwnd$ size, as illustrated in Fig. 5. The trade-off is the number of playout buffer underflow events, higher than Cubic.

Comparing $cwnd$ dynamics in Fig. 5 (X-axis in units of 100msec), one can see how slower to react to network packet loss Reno and Compound TCP variants are. Cubic reacts faster, but not as fast as CCP(1). CCPD(1,4000) has the highest range of variation; notice how steady CCPD(1,2000) $cwnd$ dynamic is, even in the presence of dropped packets due to network congestion. A large $cwnd$ size range makes more difficult to achieve a smooth video rendering experience.

We have also run the same scenario, but injecting a 0.01% packet loss. Comparative results are similar to the ones just presented, and are omitted for sake of space.

B. Network bandwidth larger than video playout rate

In this experiment, network available bandwidth is set to a value slightly larger than the average video playout rate, and video server and client are far apart (100msec rtt). Performance results are shown in Fig. 6. In this case, the number of picture discards and playout buffer underflow events is negligible

 ISBN: 978-1-61208-285-1

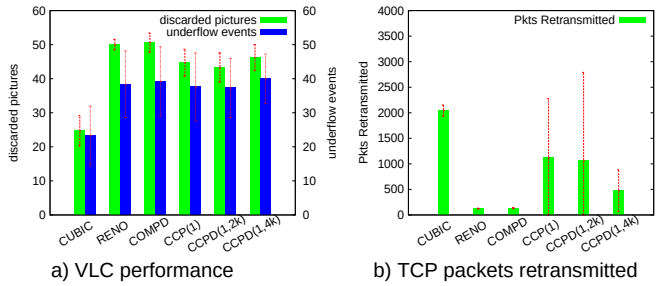


Fig. 4: Perf: AvgVR>NetBW; NoRanLoss; rtt=100msec

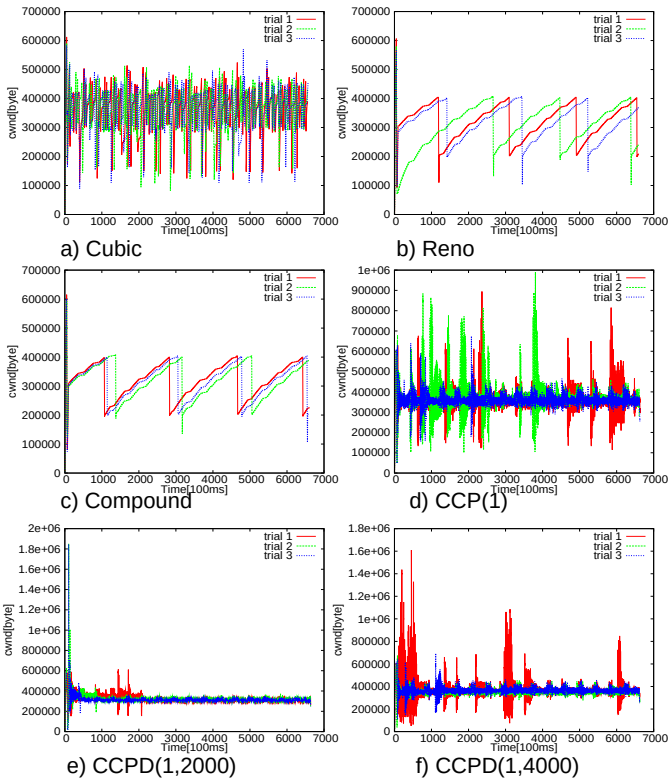


Fig. 5: Cwnd: AvgVR>NetBW; NoRanLoss; rtt=100msec

across all TCP variants. However, the least number of packet retransmits is presented by Reno and Compound, the least aggressive TCP variants. Cubic presents the largest number of packet retransmits. In contrast, in a similar lossless scenario, but with server and client close to each other (10msec rtt), is presented in Fig. 7. In this case, picture discards are again not significant for all TCP variants, even though packet retransmits are about the same for most variants, except Cubic. In general, the longer the path between video source and client, the more picture discards the streaming session will experience. This is because the client needs to render 30 frames/sec, which means a frame being rendered every 33msecs. If network latency is large and the buffer playout is not deep enough, retransmitted packets with additional rtt delay will likely arrive too late for the frame to be rendered.

When we introduce a 0.01% packet loss in the long (100msec rtt) path (Fig. 8), Reno and Compound performance

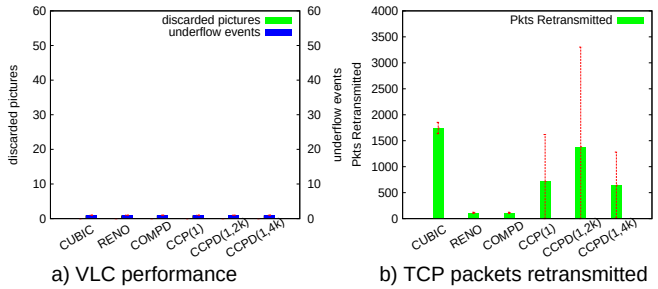


Fig. 6: Perf: AvgVR<NetBW; NoRanLoss; rtt=100msec

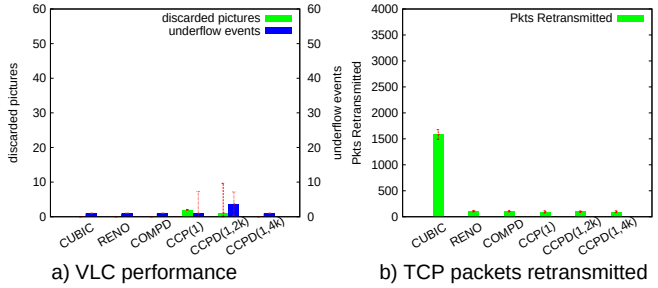


Fig. 7: Perf: AvgVR<NetBW; NoRanLoss; rtt=10msec

present the largest number of picture discards and playout buffer underflow events. Cubic, CCP and CCPD variants present negligible number of picture discards and playout buffer underflows, albeit with larger number of packet retransmits. Overall, random losses drag Reno and Compound TCP variants to a lower throughput, which in this case is below the average video playout rate, increasing playout buffer underflow events. One may conclude that responsive TCP variants deliver better streaming performance in the presence of random packet losses.

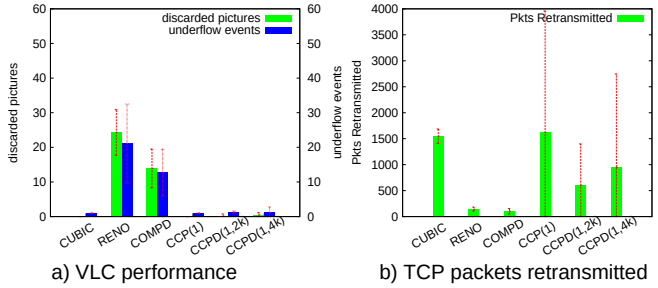


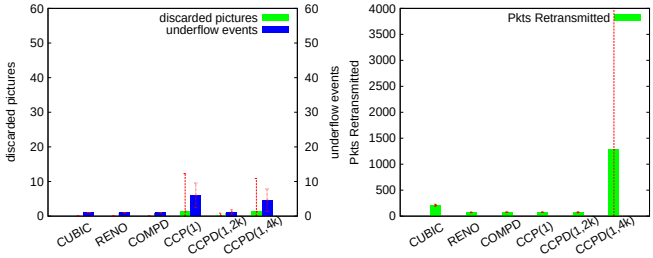
Fig. 8: Perf: AvgVR<NetBW; 0.01 % RLoss; rtt=100msec

C. Network bandwidth much larger than video playout rate

In this experiment, network bandwidth is set to a typical wireless link bandwidth, 20Mbps. Fig. 9 presents results with no random packet losses. We first notice that, when network bandwidth is plenty, there is negligible playout buffer underflow events across all TCP variants. In addition, packet retransmissions are much reduced in all TCP variants except CCPD(1,4000). In contrast, when a random packet loss rate of 0.01% is injected (Fig.10), most TCP variants increase playout buffer underflows, most notably Reno and CCPD(1,2000). All TCP variants continue to present few packet retransmissions.

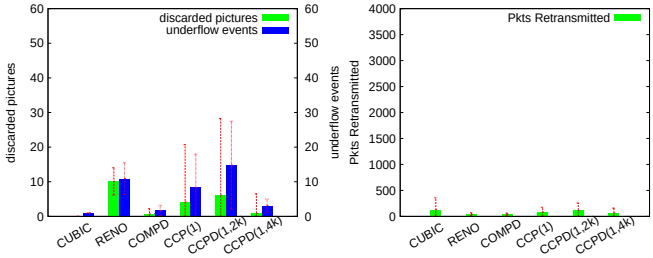
 ISBN: 978-1-61208-285-1

Overall, Cubic, Compound TCP and CCPD(1,4000) variants present the least number of picture discards.



a) VLC performance b) TCP packets retransmitted

Fig. 9: Perf: AvgVR<<NetBW; NoRanLoss;rtt=100msec

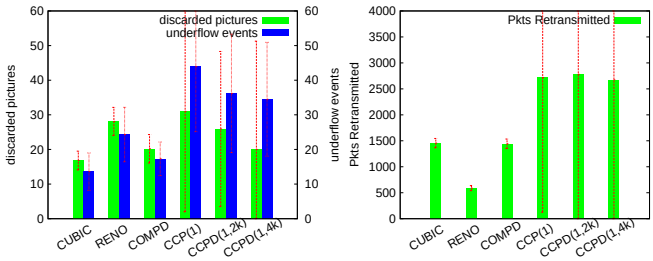


a) VLC performance b) TCP packets retransmitted

Fig. 10: Perf:AvgVR<<NetBW; 0.01%RLoss;rtt=100msec

D. WiFi access link experiment

In this experiment, the VLC client is attached to the network via a WiFi link. Before running the experiments, Iperf was used to measure the available wireless link bandwidth: 31.9Mbps, which is higher than the average video playout rate. Results are as per Fig. 11. We see that in case of plenty WiFi bandwidth, Cubic, Reno, and Compound TCP variants present the least number of discarded pictures and buffer playout underflows, followed closely by CCP and CCPD variants. In addition, CCP and CCPD protocols have the largest number of packet discards, as compared with Cubic, Reno, and Compound TCP variants. This attests to the aggressiveness of CCP and CCPD variants in pushing packets through.



a) VLC performance b) TCP packets retransmitted

Fig. 11: Perf: AvgVR<WiFiBW; rtt=100msec

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, as mentioned earlier, no tuning of TCP parameters was performed to better video client performance for any of the TCP variants studied. For our variants, we have simply used parameter values from our previous study of CCP/CCPD performance of file transfers [8].

VI. Conclusion and Future Work

In this paper, we have characterized TCP variants performance when transporting video streaming applications over wireless network type of paths via open source experiments. For widely used TCP variants, Cubic, Reno, and Compound, as well as our delay based variants, CCP and CCPD, the following can be said: i) A number of picture discards is commonplace in video streaming across all TCP variants, especially when video source and client are far apart; ii) When network bandwidth is scarce or in the presence of (wireless) packet loss, aggressive TCP variants, such as Cubic, ensure low number of picture discards; iii) Delay based TCP variants, such as CCP and CCPD, are effective in combatting random packet losses commonplace in wireless links.

Our next step is the design of a TCP variant tailored specifically for video streams. The goal is to minimize picture discards in all network conditions, as well as to avoid retransmissions of packets that are likely to be part of discarded frames at the client. This current work may also serve as a motivation for new video encoder/renderer and TCP coupling approaches, such as dynamic playout buffer re-sizing according to network bandwidth conditions.

Acknowledgment

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] M. Alnuem, J. Mellor, and R. Fretwell, “New Algorithm to Control TCP Behavior over Lossy Links, ” IEEE International Conference on Advanced Computer Control, Jan 2009, pp. 236-240.

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

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

[6] 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.

[7] D. Cavendish, Hiraku 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.

[8] D. Cavendish, K. Kumazoe, H. Ishizaki, T. Ikenaga, M. Tsuru, and Y. Oie, “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.

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

[12] 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.

[13] S. Waghmare, A. Parab, P. Nikose, 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: 978-1-61208-285-1