Congestion Avoidance TCP Improvements for Video Streaming

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

***Abstract*—Video streaming has become the major source of Internet traffic nowadays. Considering that content delivery network providers have adopted Video over Hypertext Transfer Protocol/Transmission Control Protocol (HTTP/TCP) as the preferred protocol stack for video streaming, understanding TCP performance in transporting video streams has become paramount. In our previous work, we have shown how Slow Start of TCP variants play a definite role in the quality of video experience. In this paper, we research mechanisms within congestion avoidance phase of TCP 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 when playout buffer space is used within TCP congestion avoidance phase.**

***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, as well as rendering stalls due to the lack of timely data. Transport delays and data starvation 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 data transfer 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 [3] [4]. 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 [3], and another using a proportional plus derivative controller [4]. More recently, we have studied TCP performance of most popular TCP variants - Reno [2], Cubic (Linux) [12], Compound (Windows) [13] - as well as our proposed TCP variants: Capacity and Congestion Probing (CCP) [3], and Capacity Congestion Plus Derivative (CCPD) [4], 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 a previous work, we have proposed enhancements on Slow Start phase of TCP to improve video streaming performance [7]. In this paper, we show that it is possible to also alter Congestion Avoidance phase of TCP to improve video streaming over Internet paths with wireless access links. More specifically, we demonstrate that: i) Ensuring minimum throughput above video rendering rate may hurt streaming performance rather than help it; ii) Considering playout buffer size in the congestion avoidance as extra space for TCP packet storage results in consistent performance improvement across various network path 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, as well as additional congestion avoidance schemes 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

Modifications of TCP protocol to enhance video streaming have been recently proposed. Pu et al. [10] have proposed a proxy TCP architecture for higher performance on paths with last hop wireless links. The proxy TCP node implements a variation of TCP congestion avoidance for which congestion window $cwnd$ adjustment is disabled, being replaced with

 ISBN: 978-1-61208-435-0

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 [6]. Lu et al. [11] have proposed a receiver based scheme to avoid TCP congestion control in case of lost packets on a wireless link. Our CCP and CCPD variants already differentiate between packet losses due to congestion from wireless link layer losses, their main motivation.

Park et. al. [9] 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.

An analytical framework to the dimensioning of playout buffer has been developed by [14]. The goal is to mitigate buffer underflow as well as packet retransmissions along the path. Our work does not try to dimension the playout buffer, but rather take advantage of its size to improve video streaming performance.

In [8], a relationship between network, application (streaming), and user key performance indicators is studied. They conclude that "rebuffering frequency" impacts the most in user perceived video quality, which is one of our video performance measurers (underflow events).

A distinct aspect of our current work is that we propose improvements on congestion avoidance phase of TCP, and evaluate them on real client and server network stacks that are widely deployed for video streaming, via VLC open source video client, and standard HTTP server.

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. Figure 1 illustrates video streaming components.

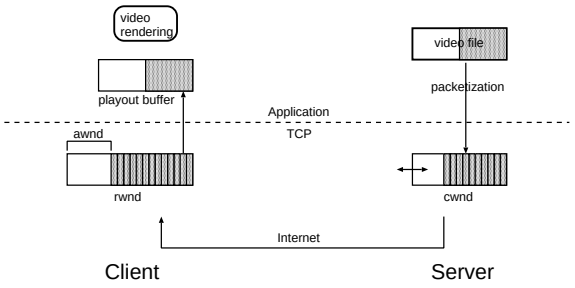


Fig. 1: Video Streaming over TCP

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 congestion avoidance scheme, 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 schemes, 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 [2] and Cubic [12] are examples of additive increase multiplicative decrease (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. Compound [13], CCP [3] and CCPD [4] are examples of delay based protocols.

 ISBN: 978-1-61208-435-0

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. 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 $cwnd$ size can only increase in this phase. So, there is no flow control of the traffic into the network. This phase ends when $cwnd$ size reaches a large value, dictated by $ssthresh$ parameter, or when the first packet loss is detected, whichever comes first. All widely used TCP variants use slow start except Cubic [12].
- **Congestion Avoidance(CA):** This phase is entered when the TCP sender detects a packet loss, or the $cwnd$ size reaches the target upper size $ssthresh$ (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. Our interest is in the congestion avoidance phase of TCP, which dictates how much traffic is allowed into the network during periods of network congestion. A comprehensive tutorial of TCP features can be found in [1].

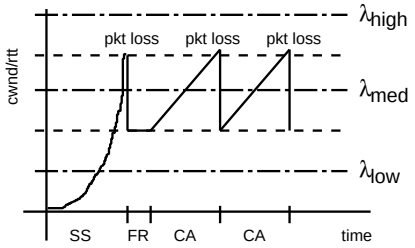


Fig. 2: TCP Congestion Window Dynamics vs Video Playout

Let λ be the video average bit rate across its entire playout time. That is, $\lambda = VideoSize/TotalPlayoutTime$. Figure 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.

For most TCP variants widely used today, congestion avoidance phase is sharply different. As we present comparative study of our proposal against Cubic and Compound TCP variants, in what follows we briefly introduce these TCP variants' congestion avoidance phase.

A. Cubic TCP Congestion Avoidance

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

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

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.

Cubic fast release fast recovery of bandwidth makes it one of the most aggressive TCP variants. Being very responsive, it quickly adapts to variations in network available bandwidth. However, because it relies on packet loss detection for $cwnd$ adjustments, random packet losses in wireless links may still impair Cubic's performance.

B. Compound TCP Congestion Avoidance

Compound TCP is the TCP of choice for most deployed 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 [13]. Compound TCP cwnd adjustment is as per 2:

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

where the delay component is computed as:

$$\begin{aligned} AckRec: dwnd_{k+1} = {} & dwnd_k + \alpha dwnd_k^K - 1, \text{ if } diff < \gamma \\ & dwnd_k - \eta diff, \quad \text{if } diff \geq \gamma \\ PktLoss: dwnd_{k+1} = {} & dwnd_k(1-\beta) - \frac{cwnd_k}{2} \end{aligned} \tag{3}$$

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

Compound TCP dynamics is often dominated by its loss based component. Hence, it presents a slow responsiveness to network available bandwidth variations, which may cause playout buffer underflows.

 ISBN: 978-1-61208-435-0

C. Capacity and Congestion Probing TCP

In this paper, we use CCP as a framework upon which we design congestion avoidance variation schemes. TCP CCP was our first proposal of 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 or decrease. In addition, packet loss does not trigger any special cwnd adjustment. CCP cwnd adjustment scheme is as per 4:

$$cwnd_k = \frac{[Kp(B - x_k) - in\_flight\_segs_k]}{2} \quad 0 \leq Kp \qquad (4)$$

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.

V. TCP Congestion Avoidance Improvements for Video Streaming

The original idea of congestion avoidance was to maintain $cwnd$ at large values without incurring in packet losses, so as to incur in highest throughput possible. However, for video applications, the ideal throughput should not deviate much from the video rendering rate, or else playout buffer underflow or frame discards may happen. For instance, there is no use in aiming at too high throughput, as packets belonging to frames whose playout time is in the future may clog the playout buffer. We introduce a couple of changes in congestion avoidance of our CCP TCP variant:

- **LimitedCongestionAvoidance:** In this scheme, our TCP variant (CCPLCA) in congestion avoidance computes its $cwnd_{ccp}$ as per Eq. 4. In addition, it computes the minimum $cwnd_{vr}$ value for which at current packet rtt experienced results on a throughput matching the video rendering rate ($cwnd_{vr} = VR/rtt$). The exercised $cwnd$ results to be the largest one, or $cwnd = MAX(cwnd_{vr}, cwnd_{ccp})$. The rational is to not allow the regulated throughput to ever go below the video rendering rate.
- **LargeBuffer:** In this scheme, TCP variant (CCPLB) uses the playout buffer length as part of its $cwnd$ computation, as follows:

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

where POB is the playout buffer size, and $POBRate$ represents a percentage of the playout buffer size used in the TCP congestion avoidance phase. The rational is to use the extra space of the playout buffer to increase throughput, reducing buffer underflow events, as well as decrease throughput when playout buffer is close to be full, avoiding frame discards.

VI. Video Streaming Performance of Congestion Avoidance Schemes

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 congestion avoidance schemes on video streaming performance.

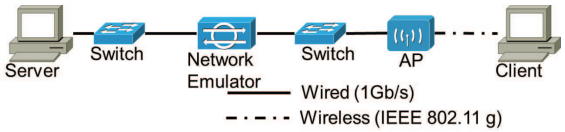


Fig. 3: Video Streaming Emulation Network

TCP variants used are: Cubic, Compound, CCP, CCPLBA, and CCPLB. Performance is evaluated for various round trip time path scenarios, as per Table I.

TABLE I: EXPERIMENTAL NETWORK SETTINGS

Video Size	409Mbytes
Playout time	10.24 secs
Encoding	MPEG-4
Video Codec	H.264/AVC
Audio Codec	MPEG-4 AAC4
Video Playout Buffer Size	448, 897, 1345 pkts
Network Delay (RTT)	3, 100, 200 msecs
TCP variants	Cubic, Compound, CCP, CCPLCA, CCPLB

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:

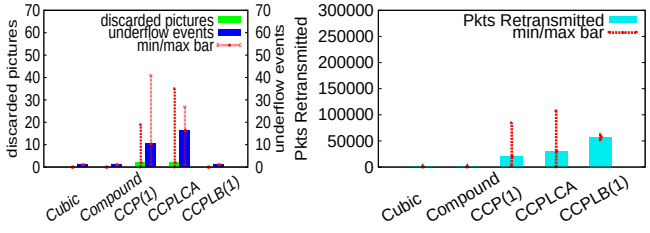
- **Picture discards:** number of frames discarded by the video decoder. This measurer defines the number of frames skipped by the video rendered at the client side.
- **Buffer underflow:** number of buffer underflow events at video client buffer. This measurer defines the number of "catch up" events, 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.

 ISBN: 978-1-61208-435-0

We organize our experimental results into the following: i)TCP variants performance comparison; ii)CCPLB sensitivity analysis. Each data point in charts represents five trials. Results are reported as average and min/max deviation bars.

A. TCP Variants Performance Comparison

Figure 4 reports on video streaming and TCP performance under short propagation delay of 3msec. In this case, legacy TCP variants Cubic and Compound deliver best video streaming performance with no discarded frames and very small number of playout buffer underflow events. CCP(1), our previous TCP variant, presents significantly more frame discards, as well as buffer underflow events. Even though CCP uses path storage capacity to regulate its input traffic, CCP ignores playout buffer depth. CCPLCA presents worst performance, which shows that simply being liberal in sizing $cwnd$ to large values may end up hurting video streaming performance, rather than helping. One needs to size $cwnd$ to large values only when the playout buffer is able to accommodate the traffic, and quickly use the extra packets to render frames on a timely manner. Finally, our new CCPLB(1) TCP variant (1 means full size of the playout buffer is used) delivers as good a performance as Cubic and Compound legacy TCPs, even though it retransmits more packets than all other TCP variants.

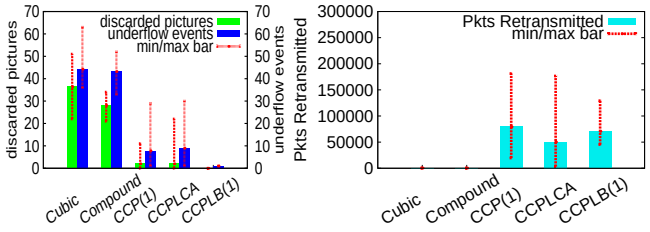


a) VLC performance b) TCP packets retransmitted

Fig. 4: Video Performance vs TCP performance; rtt=3msec

Figure 5 reports on video streaming and TCP performance under a typical propagation delay of 100msec. In this case, legacy TCP variants Cubic and Compound deliver worst video streaming performance among all TCP variants studied. CCP(1), our previous TCP variant, presents significantly less frame discards than the legacy ones, as well as buffer underflow events. Among all variants, CCPLB(1) performs best by maintaining a very low number of playout buffer underflow events, as well as no frame discards. CCPLB(1) is able to keep low number of underflow events and frame discards by taking into account the size of the playout buffer when regulating $cwnd$ window, even though it does not know the instantaneous filling level (number of packets) of the playout buffer. Notice also that the number of retransmitted packets of CCP and CCPLB are roughly the same, even though CCPLB delivers better video performance.

Figure 6 reports on video streaming and TCP performance under a large propagation delay of 200msec. Delays such as that may be experienced in paths with cellular network access links, where additional delays result from wireless access

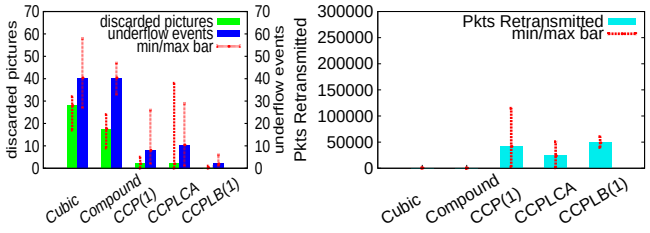


a) VLC performance b) TCP packets retransmitted

Fig. 5: Video Performance vs TCP performance; rtt=100msec

link level retransmissions. In this case, legacy TCP variants Cubic and Compound still deliver worst video streaming performance among all TCP variants. CCP(1) continues to present significantly less frame discards than the legacy ones, as well as buffer underflow events. In addition, CCPLB(1) performs best by maintaining a very low number of playout buffer underflow events, as well as no frame discards, even in the face of a very large round trip delay.

In conclusion, CCPLB is able to consistently deliver best video streaming performance across a wide range of round trip delay paths.



a) VLC performance b) TCP packets retransmitted

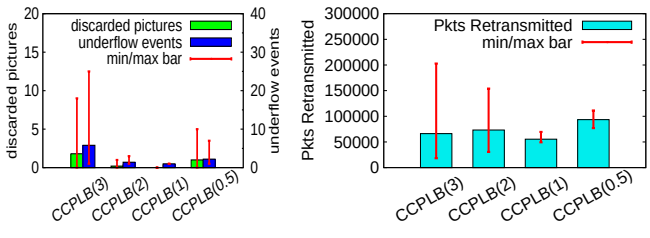
Fig. 6: Video Performance vs TCP performance; rtt=200msec

B. Playout Buffer Size Sensitivity Analysis

So far we have presented CCPLB results using the whole playout buffer size. Next we address performance sensitivity to two issues: playout buffer size itself, and the percentage of the playout buffer size used by CCPLB.

Figure 7 reports on video streaming and CCPLB performance under a typical propagation delay of 100msec and playout buffer size of 448 max size IP packets of 1600 bytes for various amounts of buffering. First notice the small amounts of picture discards, as well as underflow buffer events across all variants. CCPLB(2) uses half the buffer size of the playout buffer in its congestion avoidance $cwnd$ regulation, whereas CCPLB(0.5) uses twice as much buffer as the size of the playout buffer. The later case represents an overbooking of playout buffering, as CCPLB uses more buffering than it is really available at the client. We can see that overbooking hurts performance, whereas underbooking, using less buffering than the total playout buffer size, does not affect video streaming performance significantly. All variants present a reasonable amount of retransmitted packets at the TCP layer.

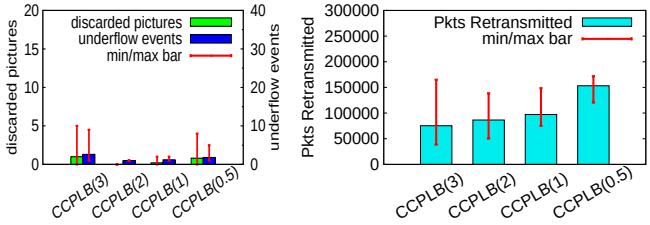
 ISBN: 978-1-61208-435-0



a) VLC performance b) TCP packets retransmitted

Fig. 7: Video Performance vs TCP performance; POB=448pkts

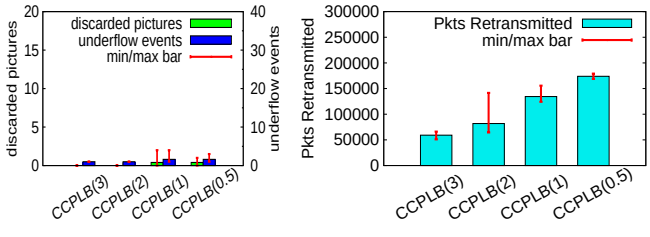
Figure 8 reports on video streaming and CCPLB performance under a typical propagation delay of 100msec and playout buffer size of 897 max size IP packets of 1600 bytes for various amounts of buffering. Comparing CCPLB VLC performance with previous case, there is much less variation in discarded frames as well as underflow events, with half the playout buffer. There is also roughly the same level of packet retransmissions at the TCP level performance from the previous case.



a) VLC performance b) TCP packets retransmitted

Fig. 8: Video Performance vs TCP performance; POB=897pkt

Finally, Figure 9 reports on video streaming and CCPLB performance under a typical propagation delay of 100msec and a large playout buffer size of 1345 max size IP packets of 1600 bytes for various amounts of buffering. Comparing CCPLB VLC performance with previous results, there is no significant improvement in VLC performance. This shows that beyond a certain size, there is no appreciable gain in increasing playout buffer size.



a) VLC performance b) TCP packets retransmitted

Fig. 9: Video Performance vs TCP performance; POB=1345pkt

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. We have simply used parameter values from our previous study of CCP performance of file transfers [5]. Finally, changes to the congestion avoidance phase of CCP can be equally applied to CCPD TCP variant.

VII. Conclusion and Future work

In this paper, we have introduced and evaluated a couple of variations of the congestion avoidance phase of our TCP protocol variant CCP to improve TCP transport performance of video streams. We have characterized CCP performance with these schemes when transporting video streaming applications over wireless network paths via open source experiments. Our experimental results show that taking into account playout buffer size in the regulation of congestion window $cwnd$ results in better video streaming experience, with fewer frame discards as well as less video rendering stalls, across a wide range of path round trip times. As future work, we are currently exploring how playout buffer size may be estimated by the video server. We are also researching how video streaming over multiple paths may affect video rendering experience.

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] 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.

[4] 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.

[5] H. Ishizaki, K. Kumazoe, T. Ikenaga, D. Cavendish, T. Masato, 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.

[6] 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.

[7] G. Watanabe, K. Kumazoe, D. Cavendish, D. Nobayashi, T. Ikenaga, and Y. Oie, “Slow Start TCP Improvements for Video Streaming Applications,” IARIA Sixth International Conference on Evolving Internet, best paper award, June 2014, pp. 22-27.

[8] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, “Measuring the Quality of Experience of HTTP Video Streaming,” Proceedings of IEEE International Symposium on Integrated Network Management, Dublin, Ireland, May 2011, pp. 485-492.

[9] J-W. Park, R. P. Karrer, and J. Kim,, “TCP-Rome: 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.

[10] 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.

[11] Z. Lu, V. S. Somayazulu, and H. Moustafa, “Context Adaptive Cross-Layer TCP Optimization for Internet Video Streaming,” In Proceedings of IEEE ICC 14, 2014, pp. 1723-1728.

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

[13] 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.

[14] J. Yan, W. Muhlbauer, and B. Plattner, “Analytical Framework for Improving the Quality of Streaming Over TCP,” IEEE Transactions on Multimedia, Vol.14, No.6, December 2012, pp. 1579-1590.

 ISBN: 978-1-61208-435-0