WPI-CS-TR-20-06 August 2020

Comparison of TCP Congestion Control Performance over a Satellite Network

by

Saahil Claypool
Jae Chung and Mark Claypool

Computer Science Technical Report Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Comparison of TCP Congestion Control Performance over a Satellite Network

Saahil Claypool
Worcester Polytechnic Institute
Worcester, MA, USA
smclaypool@wpi.edu

Jae Chung
Viasat
Marlborough, MA, USA
jaewon.chung@viasat.com

Mark Claypool
Worcester Polytechnic Institute
Worcester, MA, USA
claypool@cs.wpi.edu

***Abstract*—Satellite connections are critical for continuous network connectivity when disasters strike and for remote hosts that cannot use traditional wired, WiFi or mobile networking. While Internet satellite bitrates have increased, latency can still degrade TCP performance. Assessment of TCP over satellites is lacking, typically done by simulation or emulation only, if at all. This paper presents experiments comparing four TCP congestion control algorithms – BBR, Cubic, Hybla and PCC – on a commercial satellite network. Analysis of the results shows similar steady-state bitrates for all, but with significant differences in start up throughputs and round-trip times caused by queuing of packets in flight. Power analysis combining throughput and latency shows that overall, PCC is the most powerful, due to relatively high throughputs and low, steady round-trip times, while for small downloads Hybla is the most powerful, due to fast throughput ramp-ups. BBR generally fares similarly to Cubic in all cases.**

I. Introduction

Satellite networks are an essential part of modern society, providing ubiquitous network connectivity even in times of disaster. The number of satellites in orbit is over 2100, a 67% increase from 2014 to 2019 [1]. As few as three geosynchronous (GSO) satellites can provide global coverage, interconnecting widely distributed networks and providing "last mile" connectivity to remote hosts. The idea of "always-on" connectivity is particularly useful for redundancy, especially in an emergency when traditional (i.e., wired) connections may not be available. Recent research in satellite technology has produced spot beam technology frequency reuse to increase transmissions capacity more than 20x and the total capacity of planned GEO satellites is over 5 Tb/s.

While throughput gains for satellite Internet are promising, satellite latencies can be a challenge. GSO satellites orbit about 22k miles above the earth, which means about 300 milliseconds of latency to bounce a signal up and down, a hurdle for TCP-based protocols that rely upon round-trip time communication to advance their data windows. The physics involved for round-trip time Internet communication between terrestrial hosts using a satellite accounts for about 550 milliseconds of latency at a minimum [2]. To combat these latencies that can limit TCP throughput, many satellite operators use middle-boxes (also known as "performance enhancing proxies" or PEPs) to short-circuit the round-trip time communication to the satellite (see Figure 1). Unfortunately, encrypted connections that use TCP (such as TLS) or alternative protocols such as QUIC, can render these satellite PEPs ineffective.

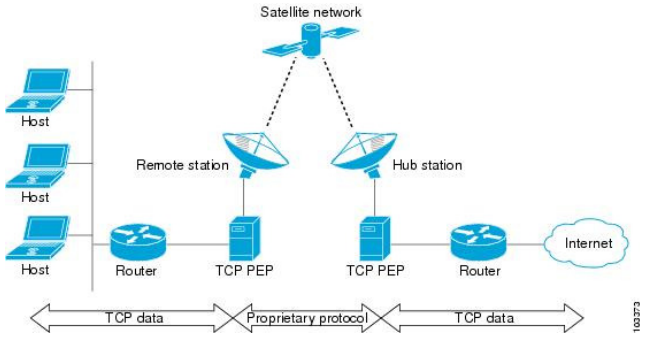


Fig. 1. TCP over a satellite with a performance enhancing proxy. [2].

TCP congestion control algorithms play a critical role in enhancing or restricting throughput in the presence of network loss and latency. TCP Cubic [3] is the default TCP congestion control algorithm in Linux and Microsoft Windows, but BBR [4] has been widely deployed by Google on Linux servers and is a congestion control option available in the QUIC transport protocol, as well [5]. A better understanding of TCP congestion control algorithm performance over satellite networks without PEPs is needed in order assess challenges and opportunities that satellites have to better support TCP and QUIC moving forward.

However, while TCP and BBR measurements have been done over wireless networks [6], there are few published studies measuring network performance over actual satellite networks [7], with most studies either using just simulation [8] or emulation with satellite parameters [9], [10], [11], [12].

This paper presents results from experiments that measure the performance of TCP over a commercial Internet satellite network. We compare four algorithms with different approaches to congestion control: default AIMD-based (Cubic [3]), bandwidth estimation-based (BBR [4]), utility function-based (PCC [12]), and satellite-optimized for startup (Hybla [13]). The network testbed and experiments are done over the Internet, but designed to be as comparable across runs as possible by interlacing runs of each protocol serially to minimize temporal differences and by doing 80 bulk downloads for each protocol to provide for a large sample. In addition, a

custom ping application provides several days worth of round-trip time and lost packet data to get a baseline on a "quiet" satellite network.

Analysis of the results shows the satellite link has consistent baseline round-trip times of about 600 milliseconds, but infrequently has round-trip times of several seconds. Loss events are similarly infrequent (less than 0.05%) and short-lived. For the TCP algorithms, all four congestion control algorithms have similar overall median throughputs. BBR achieves the maximum link capacity more often than Cubic, Hybla or PCC. However, during the start-up phase, Hybla has the highest throughput followed by PCC, BBR and Cubic, in that order – faster start-up means faster completion for short-lived downloads, such as Web pages. Overall, Hybla has the highest average throughput, owing to its higher throughput during start-up. PCC has the lowest overall round-trip time, and Hybla the highest, consistently 50% higher than PCC. BBR and Cubic round-trip times are similar and between those of PCC and Hybla. However, BBR, Cubic and PCC (to a lesser extent), have periods of high retransmission rates owing to their over-saturation of the bottleneck queue, while Hybla mostly avoids this. Power analysis that combines throughput and delay shows PCC is generally the most powerful, followed by Hybla with Cubic and BBR equally the least powerful.

The rest of this report is organized as follows: Section II describes research related to this work, Section III describes our testbed and experimental methodology, Section IV analyzes our experiment data, and Section V summarizes our conclusions and suggests possible future work.

II. Related Work

This section describes work related to our paper, including TCP congestion control (Section II-A), comparisons of TCP congestion control algorithms (Section II-B), and TCP performance over satellite networks (Section II-C).

A. TCP Congestion Control (CC)

There have been numerous proposals for improvements to TCP's congestion control algorithm since it's inception. This section highlights a few of the papers most relevant to our work, presented in chronological order.

Caini and Firrincielli [13] propose TCP Hybla to overcome the limitations traditional TCP flows have when running over high-latency links (e.g., a Satellite). TCP Hybla modifies the standard congestion window increase with: a) an extension to the "additive increase"; b) adoption of the SACK option and timestamps, which help in the presence of loss, and c) using packet spacing to reduce transmission burstiness. The slow start (SS) and congestion avoidance (CA) algorithms for Hybla are:

$$SS : cwnd = cwnd + 2^{\rho} - 1 \quad (1)$$

$$CA : cwnd = cwnd + \frac{\rho^2}{cwnd} \quad (2)$$

$$\rho = RTT/RTT_0 \quad (3)$$

where RTT_0 is typically fixed at a "wired" round-trip time of 0.025 seconds. Hybla is available for Linux as of kernel 2.6.11 (in 2005).

Ha et al. [3] develop TCP Cubic as in incremental improvement to earlier congestion control algorithms. Cubic is less aggressive than previous TCP congestion control algorithms in most steady-state cases, but can probe for more bandwidth quickly when needed. Cubic's window size is dependent only on the last congestion event, providing for more fairness to flows that share a bottleneck but have different round-trip times. TCP Cubic has been the default in Linux as of kernel 2.6.19 (in 2007), Windows 10.1709 Fall Creators Update (in 2017), and Windows Server 2016 1709 update (in 2017).

Cardwell et al. [4] develop TCP Bottleneck Bandwidth and Round-trip time (BBR) as a Cubic alternative. BBR uses the maximum bandwidth and minimum round-trip time observed over a recent time window to build a model of the network and set the congestion window size (up to twice the bandwidth-delay product). BBR has been deployed by Google servers since at least 2017 and is available for Linux TCP since Linux kernel 4.9 (end of 2016).

Dong et al. [12] propose TCP PCC that continuously observes performance based on measurements in the form of mini "experiments". Different actions taken these experiments are compared using a utility function, adopting the rate that has the best performance. The authors compare PCC against other TCP congestion control algorithms, including Cubic and Hybla, and emulate a satellite network based on parameters from a satellite Internet system. PCC is not generally available for Linux, but we were able to obtain a Linux-based implementation from Compira Labs.[1]

B. Comparison of CC Algorithms

Cao et al. [14] analyze measurement results of BBR and Cubic over a range of different network conditions. They produce heat maps and a decision tree that identifies conditions which show performance benefits from BBR over using Cubic. They find it is the relative difference between the bottleneck buffer size and bandwidth-delay product that dictates when BBR performs well. Our work extends this work by providing detailed evaluation of Cubic and BBR in a satellite configuration, with round-trip times significantly beyond those tested by Cao et al.

Ware et al. [15] model how BBR interacts with loss-based congestion control protocols (e.g., TCP Cubic). Their validated model shows BBR becomes window-limited by its in-flight cap which then determines BBR's bandwidth consumption. Their models allow for prediction of BBR's throughput when competing with Cubic with less than a 10% error.

Turkovic et al. [16] study the interactions between congestion control algorithms. They measure performance in a network testbed using a "representative" algorithm from three main groups of TCP congestion control – loss-based (TCP Cubic), delay based (TCP Vegas [17]) and hybrid (TCP BBR)

[1] https://www.compiralabs.com/

– using 2 flows with combinations of protocols competing with each other. They also do some evaluation of QUIC [18] as an alternative transport protocol to TCP. They observed bandwidth fairness issues, except for Vegas and BBR, and found BBR is sensitive to even small changes in round-trip time.

C. TCP over Satellite Networks

Obata et al. [7] evaluate TCP performance over actual (not emulated, as is typical) satellite networks. Specifically, they compare a satellite-oriented TCP congestion control algorithm (STAR) with TCP NewReno and TCP Hybla. Experiments with the Wideband InterNetworking Engineering test and Demonstration Satellite (WINDS) system show throughputs around 26 Mb/s and round-trip times around 860 milliseconds. Both TCP STAR and TCP Hybla had better throughputs over the satellite links than TCP NewReno.

Wang et al. [10] provide a preliminary performance evaluation of QUIC with BBR on a network testbed that emulates a satellite network (capacities 1 Mb/s and 10 Mb/s, RTTs 200, 400 and 1000 ms, and packet loss rates up to 20%). Their results confirm QUIC with BBR has some throughput improvements compared with TCP Cubic for their emulated satellite network.

Utsumi et al. [9] develop an analytic model for TCP Hybla for steady-state throughput and latency over satellite links. They verify the accuracy of their model with simulated and emulated satellite links (capacity 8 Mb/s, RTT 550 ms, and packet loss rates up to 2%). Their analysis shows substantial improvements to throughput over that of TCP Reno for loss rates above 0.0001%

Our work extends the above with comparative performance for four TCP congestion control algorithms on an actual, commercial satellite Internet network.

III. Methodology

In order to evaluate TCP congestion control over satellite links, we use the following methodology: setup a testbed (Section III-A), measure network baseline loss and round-trip times (Section III-B), bulk-download data using each congestion control algorithm serially (Section III-C), and analyze the results (Section IV).

A. Testbed

We setup a satellite link and configure our clients and servers so as to allow for repeated tests. Our setup is design to enable comparative performance measurements by keeping all conditions the same across runs as much as possible, except for the change in TCP congestion control algorithm.

Our testbed is depicted in Figure 2. The client is a Linux PC with an Intel i7-1065G7 CPU @ 1.30GHz and 32 GB RAM. There are four servers, each with a different TCP congestion control algorithm: Cubic, BBR, Hybla and PCC. Each server has an Intel Ken E312xx CPUs @ 2.5 GHz and 32 GB RAM. The servers and client all run Ubuntu 18.04.4 LTS, Linux kernel version 4.15.0. The servers connect to the WPI LAN

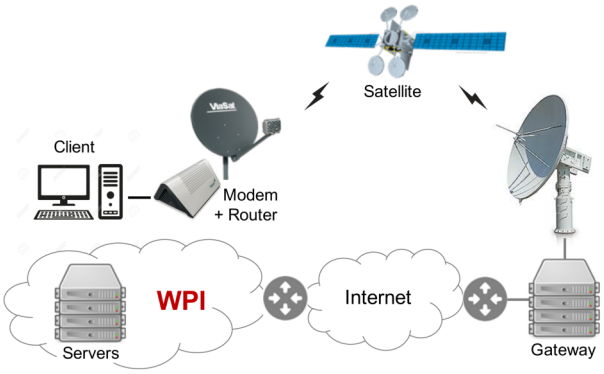


Fig. 2. Satellite measurement testbed.

via Gb/s Ethernet. The WPI campus network is connected to the Internet via several 10 Gb/s links, all throttled to 1 Gb/s.

The client connects to a Viasat satellite terminal (with a modem and router) via a Gb/s Ethernet connection.

The terminal communicates through a Ka-band outdoor antenna (RF amplifier, up/down converter, reflector and feed) through the Viasat 2 satellite[2] to the larger Ka-band gateway antenna. The terminal supports adaptive coding and modulation using 16-APK, 8 PSK, and QPSK (forward) at 10 to 52 MSym/s and 8PSK, QPSK and BPSK (return) at 0.625 to 20 MSym/s.

The connected Viasat service plan provides a peak data rate of 144 Mb/s.

The gateway does per-client queue management for traffic destined for the client, where the queue can grow up to 36 MBytes allowing the maximum queuing delay of about 2 seconds at the peak data rate. Queue lengths are controlled by Active Queue Management (AQM) that starts to randomly drop incoming packets when the queue grows over a half of the limit (i.e., 18 MBytes).

Wireshark captures all packet header data on each server and the client.

The performance enhancing proxy (PEP) that Viasat deploys by default is disabled for all experiments.

B. Baseline

For the network baseline, we run UDP Ping[3] consecutively for 1 week from a server to the client. UDP Ping sends one 20-byte packet every 200 milliseconds (5 packets/s) round-trip from the server to the client and back, recording the round-trip time for each packet returned and the number of packets lost.

C. Downloads

We compare the performance of the four congestion control algorithms: Cubic, BBR (version 1), Hybla and PCC [19]. The four servers are configured to provide for bulk-downloads

[2]https://en.wikipedia.org/wiki/ViaSat-2
[3]http://perform.wpi.edu/downloads/#udp

via `iperf3`[4] (v3.3.1), each server hosting one of our four congestion control algorithms.

Cubic, BBR and Hybla are used without further configuration. PCC is configured to use the Vivache-Latency utility function [20].

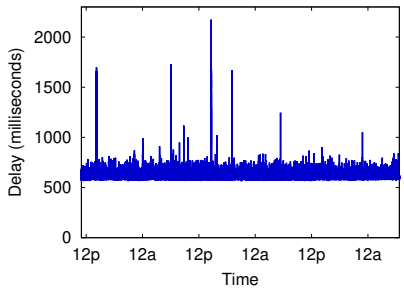
For all hosts, the default TCP buffer settings are changed on both the server and client so that flows are not flow-controlled and instead are governed by TCP's congestion window. These included setting `tcp_mem`, `tcp_wmem` and `tcp_rmem` to 60 MBytes.

The client initiates a connection to one server via iperf, downloading 1 GByte of data, then proceeding to the next server. After cycling through each server, the client pauses for 1 minute. The process repeats a total of 80 times – thus, providing 80 network traces of a 1 GByte download for each protocol over the satellite link. Since each cycle takes about 15 minutes, the throughput tests run for about a day total. We analyze results from a weekday in July 2020.

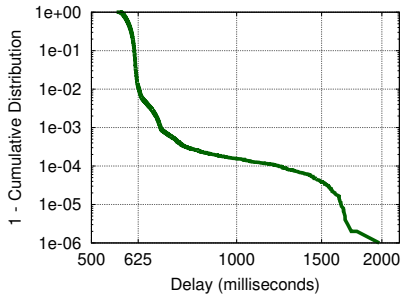
IV. Analysis

This section first presents network baseline metrics without any active TCP flows, followed by TCP download performance: overall, during steady state, and at start-up. To compare the TCP congestion control performance, we consider the metrics of throughput, delay (round-trip time) and loss (retransmissions) [21].

A. Network Baseline



(a) Versus time



(b) Distribution

Fig. 3. Baseline round-trip times.

[4]https://software.es.net/iperf/

We start by analyzing the network baseline loss and round-trip times, obtained on a "quiet" satellite link to our client – i.e., without any of our active bulk-downloads.

Figure 3 depicts analysis of about 2.5 days of round-trip times for the satellite network, obtained from UDP Ping measurements from our server to the client and back. Figure 3a show the round-trip times over that time period, with one data point every 200 milliseconds, and Figure 3b shows the complementary cumulative distribution of the same data. Table I provides summary statistics.

From the graphs and table, the vast majority (99%) of the round-trip times are between 560 and 625 milliseconds. However, the round-trip times have a heavy-tailed tendency, evidenced by the round-trip times extending from 625 ms to 1500 ms in Figure 3b and again from 1700 ms to 2200 ms on the bottom right. These high values show multi-second round-trip times can be observed on a satellite network even without any self-induced queuing. From the graph, there are no visual time of day patterns to the round-trip times.

TABLE I
Baseline round-trip time summary statistics.

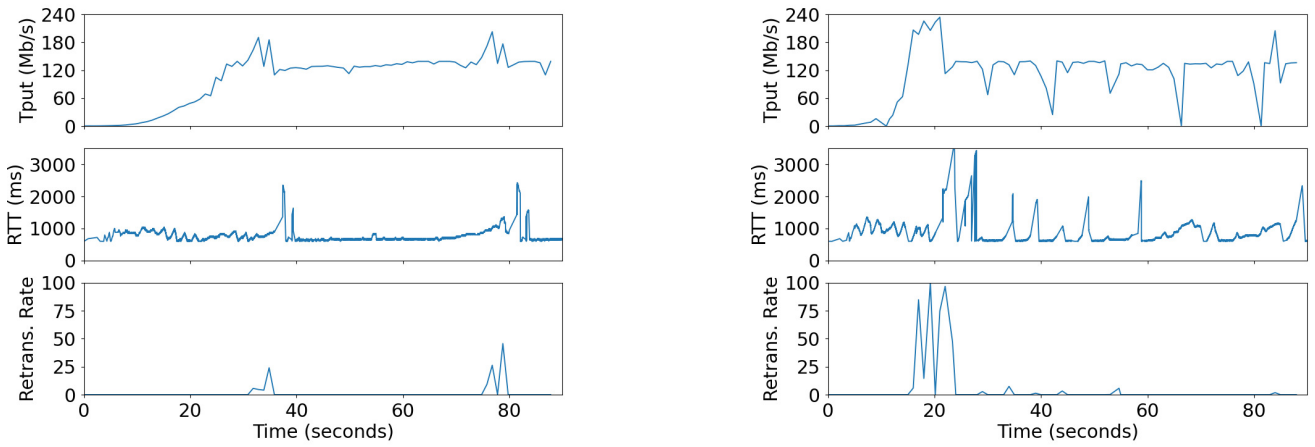
mean	597.5 ms
std dev	16.9 ms
median	597 ms
min	564 ms
max	2174 ms

In the same time period, only 604 packets are lost, or about 0.05%. Most of these (77%) are single-packet losses, with 44 multi-packet loss events, the largest 11 packets (about 2.2 seconds). There is no apparent correlation between these losses and the round-trip times (i.e., the losses do not seem to occur during the highest round-trip times observed). Note, these loss rates are considerably lower than the WINDS satellite loss of 0.7%, reported by Obata et al. [7].

B. Representative Behavior

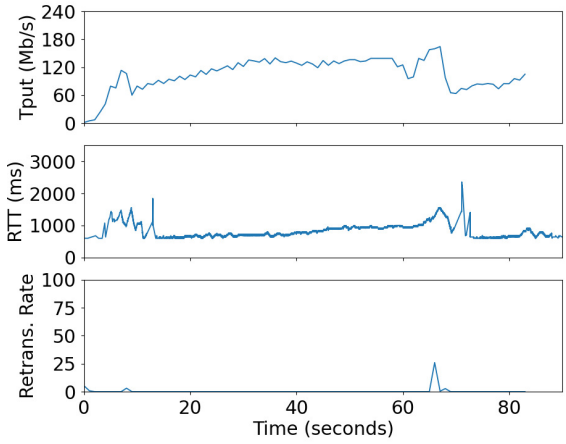
To compare the TCP congestion control protocols, we begin by examining the performance over time of a single flow that is representative of typical behavior for each protocol for our satellite connection. We analyze the throughput, round-trip time and retransmission rate, depicted in Figure 4, where each value is computed per second from Wireshark traces on the server and client.

TCP Cubic illustrates typical exponential growth in rates during start up, but exits slow start relatively early, about 15 seconds in where throughput is far lower than the expected 100 Mb/s or more. Thus, it takes Cubic about 45 seconds to reach a more expected steady state throughput of about 100 Mb/s. During steady state (post 45 seconds) the AQM drops enough packets to keep Cubic from persistently saturating the queue, resulting in round-trip times of about 1 second. However, several spikes in transmission rates yield corresponding spikes in round-trip time above 3 seconds and retransmission rates above 20 percent.

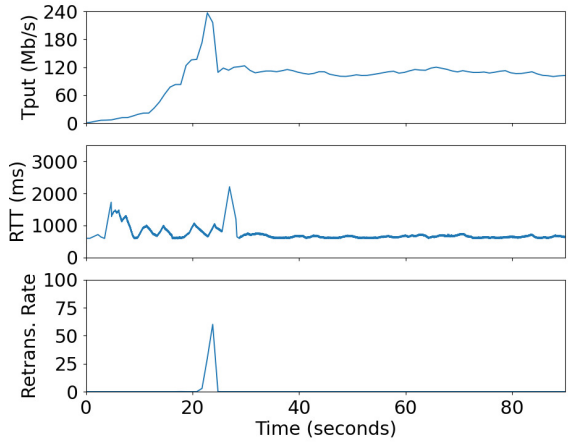


(a) CUBIC

(b) BBR



(c) Hybla



(d) PCC

Fig. 4. Stacked graph comparison. From top to bottom, the graphs are: throughput (Mb/s), round-trip time (milliseconds), and retransmission rate (percent). For all graphs, the x-axis is time (in seconds) since the flow started.

TCP BBR ramps up to higher throughput more quickly than Cubic, but this also causes high round-trip times and loss rates around 20 seconds in as it over-saturates the bottleneck queue. At steady state, BBR operates at a fairly steady 140 Mb/s, with relatively low loss and RTTs about 750 milliseconds as BBR keeps the queue below the AQM limit. However, there are noticeable dips in throughput every 10 seconds when BBR enters its PROBE_RTT state. In addition, there are intermittent round-trip time spikes and accompanying loss which occur when BBR enters PROBE_BW and increases its transmission rate for 1 round-trip time.

TCP Hybla ramps up quickly, faster than does Cubic, causing queuing at the bottleneck, evidenced by the high early round-trip times. However, there is little loss. At steady state Hybla achieves consistently high throughput, with a slight growth in the round-trip time upon reaching about 140 Mb/s. Thereupon, there is a slight upward trend to the round-trip time until the queue limit is reached accompanied by some retransmissions.

TCP PCC ramps up somewhat slower than Hybla but faster than Cubic, causing some queuing and some loss, albeit less than BBR. At steady state, throughput and round-trip times are quite consistent, near the minimum round-trip time (around 600 milliseconds), and the expected maximum throughput (about 140 Mb/s).

C. Overall

We next evaluate overall throughput, computed per second over the entirety of each flow's download. Figure 5 depicts throughput boxplot distributions at different percentiles taken across all flows and grouped by protocol. The top left is the tenth percentile, the top right the 50% (or median), the bottom left the ninetieth percentile and the bottom right the mean. Each box depicts quartiles and median for the distribution.

Points higher or lower than 1.4 $\times$ the inter-quartile range are outliers, depicted by the circles. The whiskers span from the minimum non-outlier to the maximum non-outlier. Table II provides the overall throughput summary statistics.

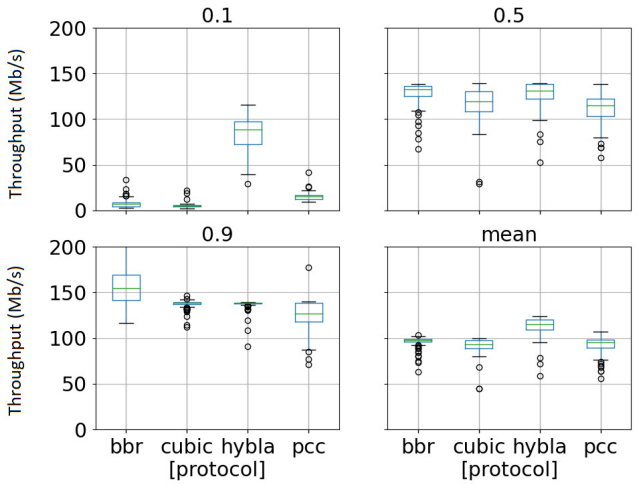


Fig. 5. Overall throughput distributions for 10%, 50%, 90% and mean.

TABLE II
OVERALL THROUGHPUT SUMMARY STATISTICS.

Protocol	Mean (Mb/s)	Std Dev
BBR	95.8	6.7
Cubic	91.0	9.5
Hybla	112.1	11.2
PCC	91.8	10.2

From the figure and table, Cubic, BBR and PCC all suffer from low throughput distributions at the tenth percentile. This is attributed to a slower start-up phase for both BBR and CUBIC, and the RTT probing phase by BBR. In contrast, Hybla has throughputs near 100 Mb/s even at the tenth percentile.

BBR and Hybla have similar throughput distributions at the median, but Cubic and PCC are still lower.

BBR has the highest throughput distributions at the ninetieth percentile, Cubic and Hybla are similar, and PCC trails by a bit. Both BBR and PCC have more variation in ninetieth percentile throughputs, evidenced by the larger boxes.

For the overall mean, Cubic and PCC are the lowest, with BBR a bit higher and Hybla the highest.

Since Cubic is the default TCP congestion control protocol for Linux and Windows servers, we compare the mean throughput for an alternate protocol choice – BBR, Hybla or PCC – to the mean of Cubic by doing independent, 2-tailed t tests ($\alpha = 0.05$) with a Bonferroni correction, as well as compute the effect sizes. The effect size provides a quantitative measure of the magnitude of difference – in our case, the difference of the means for two protocols. The Cohen's d effect size quantifies the differences in means in relation to the standard deviation. Generally small effect sizes are anything under 0.2, medium is 0.2 to 0.5, large 0.5 to 0.8, and very large is above 0.8. The t test and effect size results are shown in Table III. Statistically significance is highlighted in bold.

From the table, the mean overall throughput for BBR and Hybla are statistically different than Cubic, whereas PCC is not. The effect size for BBR versus Cubic is moderate, and the effect sizes for Hybla versus Cubic is very large.

TABLE III
OVERALL THROUGHPUT EFFECT SIZE (VERSUS CUBIC).

	t(158)	p	Effect Size
BBR	3.69	**0.0003**	0.58
Hybla	12.85	**<.0001**	2.03
PCC	0.5133	0.6084	0.08

Figure 6 shows the cumulative distributions of the round-trip times taken over the entire download for each flow. The x-axis is the round-trip time in seconds computed from the TCP acknowledgments in the Wireshark traces, and the y-axis is the cumulative distribution. There is one trendline for each protocol. Table IV provides the overall throughput summary statistics.

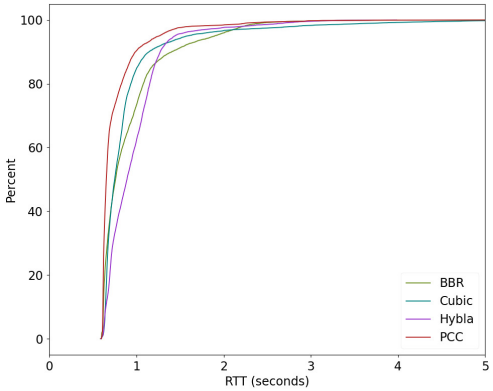


Fig. 6. Overall round-trip time distributions.

TABLE IV
OVERALL ROUND-TRIP TIME SUMMARY STATISTICS.

Protocol	Mean (ms)	Std Dev
BBR	827	85.4
Cubic	806	163.1
Hybla	906	129.3
PCC	722	49.6

From the table and figure, Hybla has a slightly higher distribution of round-trip times, with BBR and Cubic next, and PCC with the lowest round-trip time distribution. Conversely, Cubic and BBR have the heaviest tail distribution of RTTs, owing to the times they saturate the bottleneck queue. While BBR and Cubic have similar mean round-trip times, BBRs has considerably less variance. PCC has both a low round-trip time and a steady round trip time, with the smallest variance.

Cubic does have RTT values over 10 seconds, trimmed off to the right of the graph.

Figure 7 shows the cumulative distributions of the retransmissions. The axes and data groups are as for Figure 6, but the y-axis is the percentage of retransmitted packets computed over the entire flow.

From the figure, BBR and Cubic have the highest distributions of retransmissions, caused by saturating the bottleneck queue and having packets lost. Hybla has the lowest distribution of retransmissions, almost always under 1%. PCC is inbetween, consistently having a 0.5% retransmission rate, although it has a maximum near 4%.

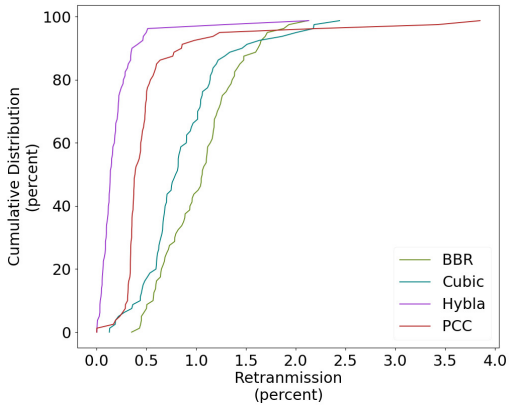


Fig. 7. Overall retransmission distributions.

D. Steady State

TCP's overall performance includes both start-up and congestion avoidance phases – the latter we call "steady state" in this paper. We analyze steady state behavior based on the last half (in terms of bytes) of each trace.

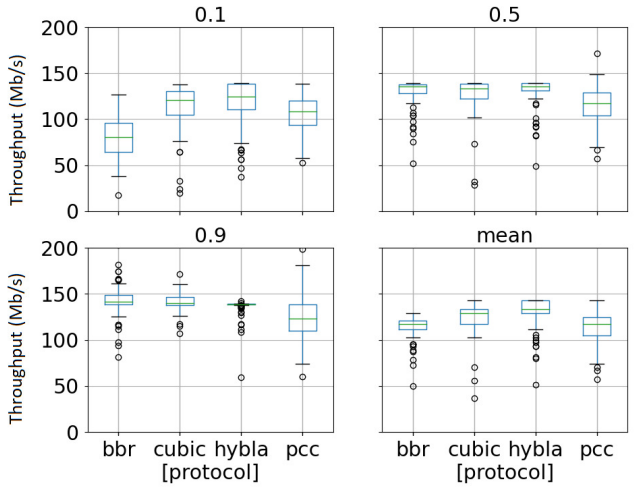


Fig. 8. Steady state throughput distributions for 10%, 50%, 90% and mean.

Figure 8 depicts throughput comparisons for the steady state of all downloads for each protocol. The graphs are as in Figure 5, but only include the last half of all downloads. Table V shows the corresponding summary statistics.

TABLE V
STEADY STATE THROUGHPUT SUMMARY STATISTICS.

Protocol	Mean (Mb/s)	Std Dev
BBR	112.9	12.2
Cubic	123.3	17.0
Hybla	130.1	17.2
PCC	112.6	17.9

From the graphs, BBR has lowest distribution of steady state throughput at the tenth percentile. This is attributed to the round-trip time probing phase by BBR, which, if there is no change to the minimum round-trip time, triggers every 10 seconds whereupon throughput is minimal for about 1 second. PCC's throughput at the tenth percentile is also a bit lower than Cubic's or Hybla's.

BBR, Cubic and Hybla all have a similar median steady state throughputs, while PCC's is a bit lower.

BBR has the highest distribution of throughput at the nineti-eth percentile, followed by Cubic, Hybla and PCC. Hybla's distribution here is the most consistent (as seen by the small box), while PCC's is the least.

From the table, Hybla has the highest mean steady state throughput, followed by CUBIC and then BBR and PCC are about the same. BBR steady state throughput varies the least.

Table VI is like Table III, but for steady state. From the table, the mean steady state throughputs are all statistically significantly different than Cubic. BBR and PCC have lower steady state throughput than Cubic with a large effect size. Hybla has a higher throughput than Cubic with a moderate effect size.

TABLE VI
STEADY STATE THROUGHPUT EFFECT SIZE (VERSUS CUBIC).

	t(158)	p	Effect Size
BBR	4.44	**<.0001**	0.7
Hybla	2.51	**0.0129**	0.4
PCC	3.88	**0.0002**	0.6

Figure 9 shows the cumulative distributions of the round-trip times during steady state. The axes and data groups are as in Figure 6, but taken only for the second half of each flow. Table VII shows the summary statistics.

TABLE VII
STEADY STATE ROUND-TRIP TIME SUMMARY STATISTICS.

Protocol	Mean (ms)	Std Dev
BBR	780	125.1
Cubic	821	206.4
Hybla	958	142.1
PCC	685	73.1

The steady state trends are similar in many ways to the overall trends, with Hybla typically having round-trip times

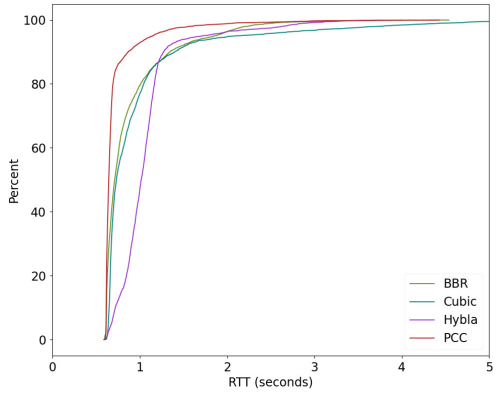


Fig. 9. Steady state round-trip time distributions.

about 200 milliseconds higher than any other protocol. PCC has the lowest and steadiest round-trip times, near the link minimum. BBR and Cubic are inbetween, with BBR being somewhat lower than Cubic and a bit steadier. Cubic, in particular, has a few cases with extremely high round-trip times. Across all flows, about 5% of the round trip times are 2 seconds or higher.

Figure 10 shows the cumulative distributions of the retransmissions during steady state. The axes and data groups are as for Figure 10, but the y-axis is the percentage of retransmitted packets computed over just the second half of each flow.

From the figure, Cubic has the highest retransmission distribution and Hybla the least. BBR and PCC are inbetween, with BBR moderately higher but PCC having a much heavier tail. Hybla and PCC are consistently low (0% loss) for about 3/4ths of all runs, compared to only about 20% for BBR and Cubic.

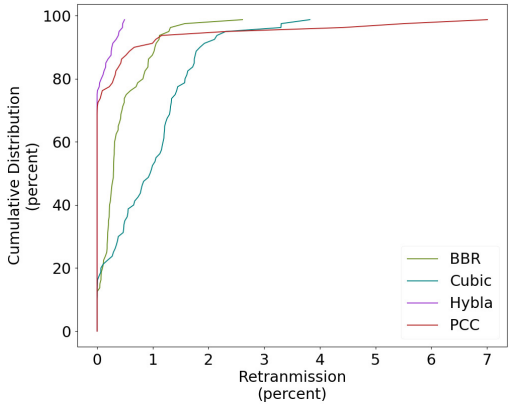


Fig. 10. Steady state retransmission distributions.

E. Start-Up

We compare the steady state behavior for each protocol by analyzing the first 30 seconds of each trace, approximately long enough to download 50 MBytes on our satellite link. This is indicative of protocol performance for some short-lived flows, such as for a large Web page.

The average Web page size for the top 1000 sites worldwide was around 2 MBytes as of 2018, a steady increase over previous years [22]. This includes the HTML payloads, as well as all linked resources (e.g., CSS files and images). The distribution's 95th percentile was about 6 MBytes and the maximum was about 29 MBytes (a shared-images Website). Today's average total Web page size is probably about 5 MBytes [23], dominated by images and video. Note, these sizes are upper bounds for the average download sizes from a Web server since the individual components of a page are obtained from different servers [24], hence different TCP flows.

Many long-lived TCP flows carry video content and these may be capped by the streaming video rate, which itself depends upon the video encoding. However, assuming videos are downloaded completely, about 90% of YouTube videos are less than 30 MBytes [25].

Traditionally, the initial congestion window for TCP is one or two maximum segment sizes (MSS) (subsequently four [26]), and during slow-start, the window increases by the MSS for each non-duplicate ACK received [27]. While there are advantages to throughputs from a larger initial window, there are risks to overshooting the slow start threshold and overflowing router queues [8].

The initial congestion window settings can vary from server to server, ranging from 10 to 50 MSS for major CDN providers [28]. The default initial window in Linux since kernel version 2.6 in 2011 is 10. Hybla multiplies the initial congestion window by ρ from Equation 3 based on the TCP handshake round-trip time measurement; for our testbed, this is an increase of about 25x over the default, so a starting window of 50. Our servers use an initial congestion window of 10 for both BBR and Cubic, with Hybla adjusting it's initial congestion window size as above. PCC's initial congestion window over our satellite connection is about 25.

Figure 11 depicts the time that would have been required to download an object on the x-axis (in seconds) for an object sized on the y-axis (in MBytes). The object size increment is 1 MByte. Each point is the average time required a protocol run to download an object of the indicated size, shown with a 95% confidence interval.

From the figure, for the smallest objects (1 MByte), Hybla and PCC download the fastest, about 4 seconds, owning to the larger initial congestion windows they both have – both PCC and Hybla have initial congestion windows about 2.5x to 5x larger than either BBR or Cubic. In general, Hybla downloads small objects fastest followed by PCC up to about 20 MBytes, then BBR and Cubic. After 20 MBytes, BBR downloads objects faster than PCC. For an average Web page download (5 MBytes), Hybla takes an average of 4 seconds, PCC 7 seconds, BBR 10 seconds and Cubic 13 seconds. For 90% of all videos and the largest Web pages (30 MBytes), Hybla takes about 8 seconds, BBR and PCC about twice that and Cubic about thrice.

Table VIII presents the summary statistics for the first 30 seconds of each flow for each protocol. During startup, Cubic

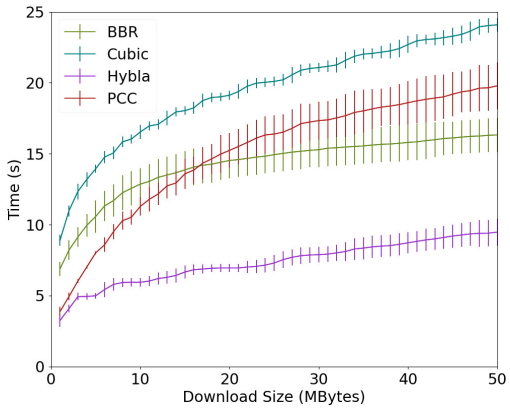


Fig. 11. Download time versus download object size.

has a low round-trip time, mostly because it takes a long time to ramp up its data rate, hence a low throughput. BBR has the highest round-trip time despite not having the highest throughput – that is had by Hybla, despite having a lower round-trip time than BBR. PCC has average throughputs and round-trip times, but the steadiest round-trip times.

TABLE VIII
START-UP SUMMARY STATISTICS.

	Tput (Mb/s)		RTT (ms)	
Protocol	Mean	Std Dev	Mean	Std Dev
BBR	23.1	1.8	917	42.9
Cubic	16.6	0.3	757	22.3
Hybla	40.8	2.9	799	130.8
PCC	20.3	1.6	806	15.1

Table VI is like Table III, but for start-up (the first 30 seconds). From the table, the start-up throughputs are all statistically significantly different than Cubic. The effect sizes for throughput for PCC, BBR and Hybla are all very large compared with Cubic.

TABLE IX
STARTUP THROUGHPUT EFFECT SIZE (VERSUS CUBIC).

	t(158)	p	Effect Size
BBR	31.9	<**.0001**	5
Hybla	74.2	<**.0001**	12
PCC	20.3	<**.0001**	3.2

F. Power

In addition to examining throughput and round-trip time separately, it has been suggested that throughput and delay can be combined into a single "power" metric by dividing throughput by delay [21] – the idea is that the utility of higher throughput is offset by lower delay and vice-versa.

Doing power analysis using the mean throughput (in Mb/s) and delay (in seconds) for each protocol for each phase (start-up, steady state and overall) yields the numbers in Table X (units are MBytes). The protocol with the most power in each phase is indicated in bold.

TABLE X
TCP POWER - THROUGHPUT ÷ DELAY

	Power (MBytes)		
Protocol	Overall	Steady	Start-up
BBR	115	144	35
Cubic	112	150	22
Hybla	123	136	**51**
PCC	**127**	**165**	25

From the table, Overall, Cubic has the least power owing to its low throughput and moderate round-trip times. PCC has the most power, with moderate throughputs but relatively low round-trip times. BBR is only slightly better than Cubic whereas Hybla is almost as good as PCC.

During steady state, PCC remains the most powerful based on high throughput with the lowest round-trip times. Cubic is more powerful than BBR or Hybla since it has good throughput and round trip times, whereas BBR is deficient in throughput and Hybla in round-trip times.

At start-up, Hybla has the most power by far, primarily due to its high throughput. BBR has moderate power, while Cubic and PCC are similar at about half the power of Hybla.

V. CONCLUSION

Satellite Internet connections are important for providing reliable, ubiquitous network connectivity, especially for hard to reach geographic regions and when conventional networks fail (e.g., in times of natural or human-caused disaster). While research in satellites has increased satellite network throughputs, the inherent latencies satellites bring are a challenge to TCP connections. Moreover, conventional approaches to split a satellite's TCP connection via a middle-box Performance Enhancing Proxy (PEP) are ineffective for encrypted TCP payloads and for the increasingly popular QUIC protocol. Alternate TCP congestion control algorithms – such as BBR or Hybla instead of the default Cubic – can play a key role in determining performance in latency-limited conditions with congestion. However, to date, there are few published research papers detailing TCP congestion control performance over actual satellite networks.

This paper presents results from experiments on a production satellite network comparing four TCP congestion control algorithms – the two dominant algorithms, Cubic and BBR, a commercial implementation of PCC, as well as the satellite-tuned algorithm Hybla. These algorithms together represent a different approaches to congestion control: default AIMD-based (Cubic), bandwidth estimation-based (BBR), utility function-based (PCC), and satellite-optimized for startup (Hybla) Results from 80 downloads for each protocol, interlaced so as to minimize temporal differences, are analyzed for overall, steady state and start-up performance. Baseline satellite network results are obtained by long-term round-trip analysis in the absence of any other traffic.

Overall, the production satellite link has consistent baseline round-trip times near the theoretical minimum (about 600 milliseconds) and very low (about a twentieth of a percent) loss rates. For TCP downloads, during steady state, the four protocols evaluated – Cubic, BBR, Hybla and PCC – all have similar median throughputs, but Hybla and Cubic have slightly higher mean throughputs owing to BBR's bitrate reduction when probing for minimal round-trip times (probing lasts about a second and happens about once per second). During start-up, Hybla's higher throughputs allow it to complete small downloads (e.g., Web pages) about twice as fast as BBR ($\sim$5 seconds versus $\sim$10), while BBR is about 50% faster (10 seconds versus 15 seconds) than Cubic. Hybla is able to avoid some of the high retransmission rates brought on by Cubic and BBR, and to a lesser extent PCC, saturating the bottleneck queue, too. However, as a cost, Hybla has a consistently higher round-trip time, an artifact of more packets in the bottleneck queue, while PCC has the least. Combining throughput and round-trip into one "power" metric shows PCC the most powerful, owing to high throughputs and steady, low round-trip times.

There are several areas we are keen to pursue as future work. Settings to TCP, such as the initial congestion window, may have a significant impact on performance, especially for small object downloads. Since prior work has shown TCP BBR does not always share a bottleneck network connection equitably with TCP Cubic [29], future work is to run multiple flow combination with homo- and heterogeneous congestion control protocols over the satellite link. Lastly, experiments with the increasingly popular QUIC protocol are warranted since QUIC can use BBR and has encrypted payloads, making it difficult to use with satellite PEPs.

Acknowledgments

Thanks to Amit Cohen, Lev Gloukhenki and Michael Schapira of Compira Labs for providing the implementation of PCC.

References

[1] S. I. Association, "Introduction to the Satellite Industry," Online presentation: https://tinyurl.com/y5m7z77e, 2020.

[2] Cisco, *Interface and Hardware Component Configuration Guide, Cisco IOS Release 15M&T*. Cisco Systems, Inc., 2015, chapter: Rate Based Satellite Control Protocol.

[3] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, 2008.

[4] N. Cardwell and Y. Cheng and C. S. Gunn and S. H. Yeganeh and Van Jacobson, "BBR: Congestion-based Congestion Control," *Communications of the ACM*, no. 2, pp. 58–66, Jan. 2017.

[5] N. Cardwell, Y. Cheng, S. H. Yeganeh, and V. Jacobson, "BBR Congestion Control," *IETF Draft draft-cardwell-iccrg-bbr-congestion-control-00*, Jul. 2017.

[6] F. Li, J. W. Chung, X. Jiang, and M. Claypool, "TCP CUBIC versus BBR on the Highway," in *Proceedings of the Passive and Active Measurement Conference (PAM)*, Berlin, Germany, Mar. 2018.

[7] H. Obata, K. Tamehiro, and K. Ishida, "Experimental Evaluation of TCP-STAR for Satellite Internet over WINDS," in *Proceedings of the International Symposium on Autonomous Decentralized Systems*, Tokyo, Japan, Mar. 2011.

[8] C. Barakat, N. Chaher, W. Dabbous, and E. Altman, "Improving TCP/IP over Geostationary Satellite Links," in *Proceedings of GLOBECOM*, Rio de Janeireo, Brazil,, Dec. 1999.

[9] S. Utsumi, S. Muhammad, S. Zabir, Y. Usuki, S. Takeda, N. Shiratori, Y. Katod, and J. Kimb, "A New Analytical Model of TCP Hybla for Satellite IP Networks," *Journal of Network and Computer Applications*, vol. 124, Dec. 2018.

[10] Y. Wang, K. Zhao, W. Li, J. Fraire, Z. Sun, and Y. Fang, "Performance Evaluation of QUIC with BBR in Satellite Internet," in *Proceedings of the 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Huntsville, AL, USA, Dec. 2018.

[11] V. Arun and H. Balakrishnan, "Copa: Practical Delay-Based Congestion Control for the Internet," in *Proceedings of the Applied Networking Research Workshop*, Montreal, QC, Canada, Jul. 2018.

[12] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting Congestion Control for Consistent High Performance," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Oakland, CA, USA, 2015.

[13] C. Caini and R. Firrincieli, "TCP Hybla: a TCP Enhancement for Heterogeneous Networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, Sep. 2004.

[14] Y. Cao, A. Jain, K. Sharma, A. Balasubramanian, and A. Gandhi, "When to Use and When not to Use BBR: An Empirical Analysis and Evaluation Study," in *Proceedings of the Internet Measurement Conference (IMC)*, Amsterdam, NL, Oct. 2019.

[15] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry, "Modeling BBR's Interactions with Loss-Based Congestion Control," in *Proceedings of the Internet Measurement Conference (IMC)*, Amsterdam, Netherlands, Oct. 2019.

[16] B. Turkovic, F. A. Kuipers, and S. Uhlig, "Interactions Between Congestion Control Algorithms," in *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA)*, Paris, France, 2019.

[17] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, 1994.

[18] A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, , and F. Yang, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the ACM SIGCOMM Conference*, Los Angeles, CA, USA, Aug. 2017.

[19] A. Cohen, "How compira solves the last mile for streaming media," Online: https://tinyurl.com/yymxtubj, Jan. 2020.

[20] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "PCC Vivace: Online-Learning Congestion Control," in *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Renton, WA, USA, Apr. 2018.

[21] S. Floyd, "Metrics for the Evaluation of Congestion Control Mechanisms," Internet Requests for Comments, RFC 5166, March 2008.

[22] Data and Analysis, "Webpages Are Getting Larger Every Year, and Here's Why it Matters," Solar Winds Pingdom. Online at: https://tinyurl.com/y4pjrvhl, November 15 2018.

[23] T. Everts, "The Average Web Page is 3 MB. How Much Should We Care?" Speed Matters Blog. Online at: https://speedcurve.com/blog/web-performance-page-bloat/, August 9th 2017.

[24] M. Kaplan, M. Claypool, and C. Wills, "How's My Network? - Predicting Performance from within a Web Browser Sandbox," in *Proceedings of the 37th IEEE Conference Workshop on Local Computer Networks (LCN)*, Clearwater, FL, USA, Oct. 2012.

[25] X. Che, B. Ip, and L. Lin, "A Survey of Current YouTube Video Characteristics," *IEEE Multimedia*, vol. 22, no. 2, April - June 2015.

[26] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," Internet Requests for Comments, RFC 2414, September 1998.

[27] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," Internet Requests for Comments, RFC 2581, April 1999.

[28] CDN Planet, "Initcwnd Settings of Major CDN Providers," Online: https://www.cdnplanet.com/blog/initcwnd-settings-major-cdn-providers/, feb 2017.

[29] S. Claypool, M. Claypool, J. Chung, and F. Li, "Sharing but not Caring - Performance of TCP BBR and TCP CUBIC at the Network Bottleneck," in *Proceedings of the 15th IARIA Advanced International Conference on Telecommunications (AICT)*, Nice, France, Aug. 2019.