Comparison of TCP Congestion Control Performance over a Satellite Network

Saahil Claypool[1], Jae Chung[2], and Mark Claypool[1]

[1] Worcester Polytechnic Institute, Worcester, MA, USA
{smclaypool,claypool}@wpi.edu
[2] Viasat, Marlborough, MA, USA
jaewon.chung@viasat.com

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 satellite Internet bitrates have increased, latency can still degrade TCP performance. Realistic 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 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 consistent, relatively low 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 both cases.

1 Introduction

Satellites are an essential part of modern networking, providing ubiquitous connectivity even in times of disaster. The are 2100+ satellites in orbit, a 67% increase from 2014 to 2019 [2]. Improvements in satellite technology have produced spot beam frequency reuse to increase transmission capacities more than 20x with the total capacity of planned Geosynchronous orbit satellites over 5 Tb/s.

Geosynchronous orbit satellites have about 300 milliseconds of latency to bounce a signal up and down [9], a hurdle for TCP protocols that use round-trip time communication to advance their data windows. TCP congestion control algorithms play a critical role determining throughput in the presence of network latency and loss. TCP Cubic [16] is the default TCP congestion control algorithm in Linux and Microsoft Windows, but BBR [17] has been widely deployed for TCP by Google on Linux servers and is a congestion control option available in the QUIC transport protocol, as well [7]. A better understanding of TCP congestion control algorithm performance over satellite networks is needed in

order to assess challenges and opportunities that satellites have to better support TCP moving forward.

However, there are few published studies measuring network performance over actual satellite networks [18], with most studies either using just simulations [3] or emulations with satellite parameters [1, 12, 21, 22].

This paper presents results from experiments that measure the performance of TCP over a commercial Internet satellite network. We compare four TCP congestion control algorithms, chosen based on their representative approaches to congestion control: default loss-based Cubic [16], bandwidth-delay product-based BBR [17], utility function-based PCC [12], and satellite-optimized Hybla [5]. Our network testbed and experiments are done on the Internet, but are designed to be as comparable 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 for a baseline satellite network with no other traffic.

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 TCP congestion control, BBR, Cubic and Hybla have slight higher median and 90th percentile steady-state throughput than 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. At steady-state, PCC has the lowest 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 2 presents related work, Section 3 describes our methodology, Section 4 analyzes the data, and Section 5 summarizes our conclusions and future work.

2 Related Work

This section describes work related to our paper, including TCP congestion control (Section 2.1), comparisons of TCP congestion control algorithms (Section 2.2), and TCP performance over satellite networks (Section 2.3).

2.1 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 Firrinielli [5] propose TCP Hybla to overcome the limitations TCP NewReno flows have when running over high-latency links (e.g., a Satellite). TCP Hybla modifies the standard congestion window increase with an extension based on the round-trip time. In Hybla slow-start, $cwnd = cwnd + 2^{\rho} - 1$ and in congestion avoidance $cwnd = cwnd + \frac{\rho^2}{cwnd}$, where $\rho = RTT/RTT_0$. RTT_0 is 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. [16] 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. 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. [17] provide TCP Bottleneck Bandwidth and Round-trip time (BBR) as an alternative to Cubic's (and Hybla's) loss-based congestion control. BBR uses the maximum bandwidth and minimum round-trip time observed to 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 as of kernel 4.9 (end of 2016).

Dong et al. [12] propose TCP PCC that observes performance based on small measurement "experiments". During these experiments, throughput, loss, and round-trip times are assessed with a utility function, adopting the rate that has the best utility. PCC is not generally available for Linux, but Compira Labs[3] provided us a Linux-based implementation.

2.2 Comparison of CC Algorithms

Cao et al. [6] 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. [23] 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. [20] 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 [4]) and hybrid (TCP BBR) – using 2 flows

[3] `https://www.compiralabs.com/`

with combinations of protocols competing with each other. They also do some evaluation of QUIC [19] 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.

2.3 TCP over Satellite Networks

Obata et al. [18] 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 have better throughputs over the satellite links than TCP NewReno, but while we evaluate TCP Hybla, there is no public Linux implementation of TCP STAR available.

Wang et al. [22] provide preliminary performance evaluation of QUIC with BBR on an emulated 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 throughput improvements compared with TCP Cubic for their emulated satellite network.

Utsumi et al. [21] develop an analytic model for TCP Hybla for steady state throughput and round-trip time 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.

3 Methodology

In order to evaluate TCP congestion control over satellite links, we: setup a testbed (Section 3.1), measure network baseline loss and round-trip times (Section 3.2), bulk-download data using each congestion control algorithm serially (Section 3.3), and analyze the results (Section 4).

3.1 Testbed

We setup a Viasat[4] satellite Internet link so as to represent a client with a "last mile" satellite connection. Our servers are configured to allow for repeated tests and comparative performance by serial runs with all conditions the same, except for the change in TCP congestion control algorithm.

Our testbed is depicted in Figure 1. The client is a Linux PC with an Intel i7-1065G7 CPU @ 1.30GHz and 32 GB RAM. There are four servers, each with

[4] `https://www.viasat.com/internet`

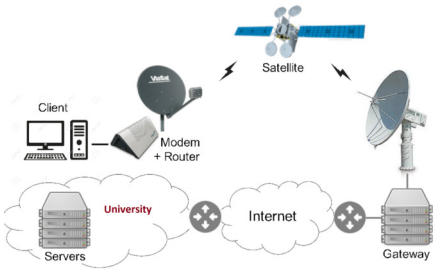


Fig. 1. Satellite measurement testbed.

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 our University LAN via Gb/s Ethernet. The campus network is connected to the Internet via several 10 Gb/s links, all throttled to 1 Gb/s. Wireshark captures all packet header data on each server and the client.

The client connects to a Viasat satellite terminal (with a modem and router) via a Gb/s Ethernet connection. The client's downstream Viasat service plan provides a peak data rate of 144 Mb/s.

The terminal communicates through a Ka-band outdoor antenna (RF amplifier, up/down converter, reflector and feed) through the Viasat 2 satellite[5] to the larger Ka-band gateway antenna. The satellite is capable of aggregate throughput of up to 260 Gb/s. 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 Viasat gateway performs per-client queue management, where the queue can grow up to 36 MBytes, allowing a maximum queuing delay of about 2 seconds at the peak data rate. Queue lengths are controlled by Active Queue Management (AQM) that randomly drops incoming packets when the queue is over a half of the limit (i.e., 18 MBytes).

The performance enhancing proxy (PEP) that Viasat deploys by default is disabled for all experiments in order to assess congestion control performance independent of the specific PEP implementation, and to represent cases where the PEP could not be used (e.g., for encrypted flows).

3.2 Baseline

For the network baseline, we run UDP Ping[6], from a server to the client continuously for 1 week. This sends one 20-byte UDP packet every 200 milliseconds (5 packets/s) from the server to the client and back, recording the round-trip time for each packet returned and the number of packets lost. Doing round-trip

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

time measurements via UDP avoids any potential special treatments routers may have for traditional ICMP packets.

3.3 Downloads

We compare the performance of four congestion control algorithms, chosen as representatives of different congestion control approaches: loss-based Cubic, bandwidth-delay product-based BBR (version 1), satellite-optimized loss-based Hybla and utility function-based PCC. The four servers are configured to provide for bulk-downloads via `iperf3`[7] (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 Vivace-Latency utility function [13], with throughput, loss, and round-trip time coefficients set to 1, 10, and 2, respectively.

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 include setting `tcp_mem`, `tcp_wmem` and `tcp_rmem` to 60 MBytes.

The client initiates a connection to one server via iperf, downloading 1 GByte, then immediately 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.

4 Analysis

This section presents network baseline metrics, followed by TCP steady state and start-up performance considering throughput, delay (round-trip time) and loss (retransmissions) [15].

4.1 Network Baseline

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. Table 3 provides summary statistics.

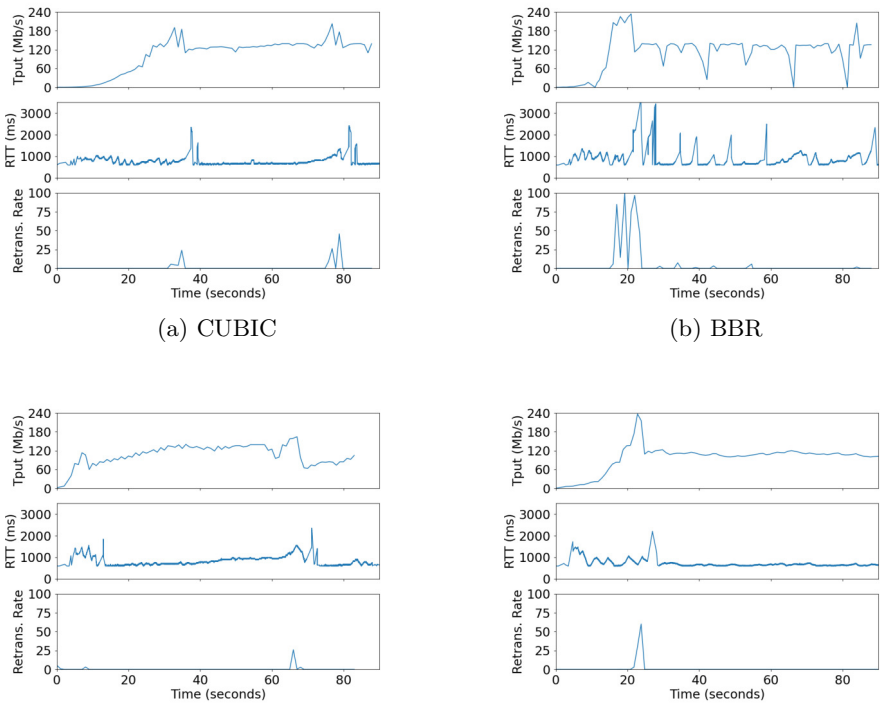
The vast majority (99%) of the round-trip times are between 560 and 625 milliseconds (median 597 ms, mean 597.5 ms, std dev 16.9 ms). However, the round-trip times have a heavy-tailed tendency, with 0.1% from 625 ms to 1500 ms and 0.001% from 1700 to 2200 ms. These high values show multi-second round-trip times can be observed on a satellite network even without any self-induced queuing. There are no visual time of day patterns to the round-trip times.

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

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 about 15x lower than the reported WINDS satellite loss of 0.7% [18].

4.2 Representative Behavior

We begin by examining the TCP congestion control performance over time for a single download representative of typical behavior for each protocol for our satellite connection. Figure 2 depicts the throughput, round-trip time and retransmission rate where each value is computed per second from Wireshark traces on the server.



(a) CUBIC (b) BBR

(c) Hybla (d) PCC

Fig. 2. 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 Cubic illustrates typical exponential growth in throughput during start-up, but exits slow start relatively early, about 15 seconds in where throughput is far lower than the link capacity. Thus, it takes Cubic about 30 seconds to reach the 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.

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 BBR 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 since the 2x bandwidth-delay product BBR keeps in flight is below the AQM queue 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 since it adjusts congestion window growth based on latency, causing queuing at the bottleneck, evidenced by the high early round-trip times. However, there are few retransmissions. 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 retransmissions, albeit fewer than BBR. At steady state, throughput and round-trip times are consistent, near the minimum round-trip time (around 600 milliseconds), and the expected maximum throughput (about 140 Mb/s). The lower round-trip times are expected since round-trip time is used by the PCC utility function.

4.3 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.

For each protocol, we compute steady state throughput in 1 second intervals, extracting the 10th, 50th and 90th percentiles (and means) across all flows. Figure 3 shows the boxplot distributions. The top left is the distribution for the 10th percentiles, the top right the 50th (or median), the bottom left the 90th 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 1 shows the corresponding summary statistics.

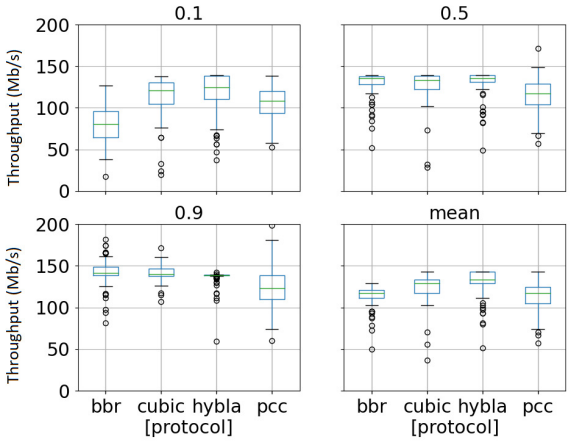


Fig. 3. Steady state throughput distributions for 10%, 50%, 90% and mean.

Table 1. 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

Table 2. 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

From the graphs, at the 10th percentile BBR has lowest distribution of steady state throughput. This is attributed to the minimal throughput during the round-trip time probing phase, which, if there is no change to the minimum round-trip time, triggers every 10 seconds and lasts for about 1 second. PCC's throughput at the 10th percentile is also a bit lower than Cubic's or Hybla's, possibly because PCC's reward for a low round-trip time can result in occasional under-utilization.

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 90th percentile, followed by Cubic, Hybla and PCC. BBR's estimation of the link bandwidth may yield more intervals of high throughput than the other protocols. Hybla's 90th percentile distribution is the most consistent (as seen by the small box), while PCC's is the least, maybe due to fuller queues and emptier queues, respectively (see Table 4).

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, probably since the consistent link quality provides for a steady delivery rate and round-trip time.

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 for Cubic by independent, 2-tailed t tests ($\alpha = 0.05$) with a Bonferroni correction and compute the effect sizes. An effect size provides a quantitative measure of the magnitude of difference – in our case, the difference of the means for two protocols. In short, effect size quantifies how much the difference in congestion control protocols matters. The Cohen's d effect size quantifies the differences in means in relation to the pooled 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 2. Statistical significance is highlighted in bold.

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

Figure 4 shows the round-trip times during steady state. 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 4 shows the summary statistics.

Table 3. Baseline round-trip time summary statistics.

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

Table 4. 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

During steady state, Hybla typically has round-trip times about 200 milliseconds higher than any other protocol, likely because its aggressive congestion window growth with high round-trip time yields more queueing delay. PCC has the lowest and steadiest round-trip times, near the link minimum, likely because its utility function rewards low round-trip times. BBR and Cubic are in-between, 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 5 shows the retransmissions during steady state. The axes and data groups are as for Figure 4, but the y-axis is the percentage of retransmitted packets computed over the second half of each flow.

From the figure, Cubic has the highest retransmission distribution and Hybla the lowest. BBR and PCC are in-between, with BBR moderately higher but PCC having a much heavier tail. Hybla and PCC are consistently low (0%) for about 75% of all runs, compared to only about 20% for BBR and Cubic.

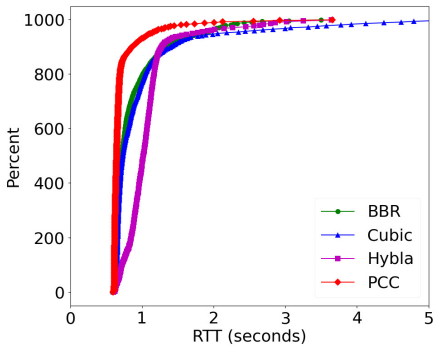


Fig. 4. Steady state round-trip time distributions.

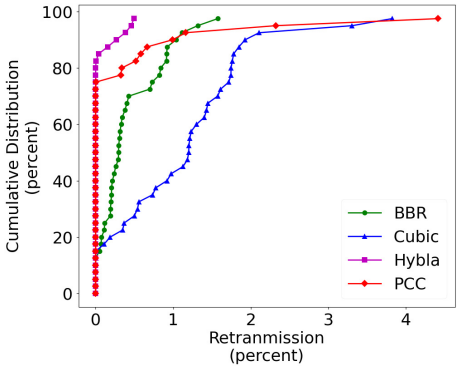


Fig. 5. Steady state retransmission distributions.

While higher round-trip times generally mean larger router queues and more drops and retransmissions, the Viasat AQM does not drop packets until the queue is above about 1 second of delay. This means if a flow's round-trip times remain under about 1.6 seconds, it can avoid retransmissions.

4.4 Start-Up

We compare the start-up 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 and is about when we observed throughput growth over time "flattening" for most flows.

The average Web page size for the top 1000 sites worldwide was around 2 MBytes as of 2018 [11], including HTML payloads, and all linked resources (e.g., CSS files and images). The Web page size distribution's 95th percentile was about 6 MBytes and the maximum was about 29 MBytes. Today's average total Web page size is probably about 5 MBytes [14], dominated by images and video.

Many TCP flows stream video content and these may be capped by the 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 [8].

Figure 6 depicts the time on the y-axis (in seconds) that would have been required to download an object for the given size on the x-axis (in MBytes). The object size increment is 1 MByte. Each point is the average time required by a protocol 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 (2.5x to 5x larger than either BBR or Cubic). In general, this

larger initial window means 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, likely because BBR has reached its estimate maximum bottleneck bandwidth. 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.

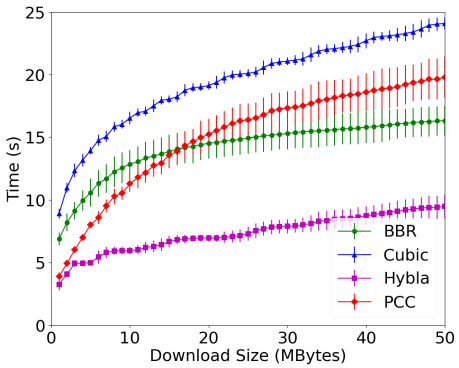


Fig. 6. Download time versus download object size.

Table 5 presents the summary statistics for the first 30 seconds of each flow for each protocol. During start-up, Cubic has a low round-trip time, mostly because it takes a long time to ramp up 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. The relatively higher average round-trip time for BBR may be because it keeps up to a bandwidth-delay product of packets in queue. PCC has average throughputs and round-trip times, but the steadiest round-trip times, possibly stabilized by the utility function rather than probing for increased data rates (and causing variable amounts of queuing) as do the other protocols.

Table 5. 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 6. 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

Table 6 is like Table 2, 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 comparing Cubic throughput to PCC, BBR and Hybla throughputs are all very large.

4.5 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 [15] – the idea is that the utility of higher throughput is offset by higher delay and vice-versa. Doing power analysis using the mean throughput (in Mb/s) and delay (in seconds) for each protocol for start-up and steady state yields the numbers in Table 7 (units are MBits). The protocol with the most power in each phase is indicated in bold.

Table 7. TCP Power – throughput ÷ delay

	Power (MBits)	
Protocol	Steady	Start-up
BBR	145	25
Cubic	150	22
Hybla	136	**51**
PCC	**164**	25

During steady state, PCC is 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, Cubic and PCC are similar at about half the power of Hybla.

5 Conclusion

Satellite Internet connections are important for providing reliable connectivity, but 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, and the satellite-tuned Hybla. These algorithms have different approaches to congestion control: loss-based (Cubic), bandwidth estimation-based (BBR), utility function-based (PCC), and satellite-optimized (Hybla). Results from 80 downloads for each protocol, interlaced so as to minimize temporal differences, provide for steady state and start-up performance. Baseline satellite network results are obtained by long-term round-trip analysis in the absence of our 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 happens every 10 seconds and lasts for about 1.5 seconds). During start-up, Hybla's higher throughputs allow it to complete small downloads (e.g., Web pages) about twice as fast as BBR (∼5 seconds versus ∼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 consistently higher round-trip times, an artifact of more packets in the bottleneck queue, while PCC has the lowest. 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, as may protocol-specific settings such as RTT_0 for TCP Hybla. Since prior work has shown TCP BBR does not always share a bottleneck network connection equitably with TCP Cubic [10], future work is to run multiple flow combinations over the satellite link, including QUIC [7]. When TCP BBR v2 is out of alpha/preview, we plan to evaluate it, as well. Other future work is to compare the protocols with and without a performance enhancing proxy (PEP), designed to mitigate the high-latencies on the satellite link.

Acknowledgments

Thanks to Amit Cohen, Lev Gloukhenki and Michael Schapira of Compira Labs for providing the implementation of PCC. Also, thanks to the anonymous reviewers for their thoughtful feedback and suggestions on improving our paper.

References

1. Arun, V., Balakrishnan, H.: Copa: Practical Delay-Based Congestion Control for the Internet. In: Proceedings of the Applied Networking Research Workshop. Montreal, QC, Canada (Jul 2018)
2. Association, S.I.: Introduction to the Satellite Industry. Online presentation: `https://tinyurl.com/y5m7z77e` (2020)
3. Barakat, C., Chaher, N., Dabbous, W., Altman, E.: Improving TCP/IP over Geostationary Satellite Links. In: Proceedings of GLOBECOM. Rio de Janeireo, Brazil, (Dec 1999)
4. Brakmo, L., O'Malley, S., Peterson, L.: TCP Vegas: New Techniques for Congestion Detection and Avoidance. ACM SIGCOMM Computer Communication Review **24**(4) (1994)

5. Caini, C., Firrincieli, R.: TCP Hybla: a TCP Enhancement for Heterogeneous Networks. International Journal of Satellite Communications and Networking **22**(5), 547–566 (Sep 2004)
6. Cao, Y., Jain, A., Sharma, K., Balasubramanian, A., Gandhi, A.: 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)
7. Cardwell, N., Cheng, Y., Yeganeh, S.H., Jacobson, V.: BBR Congestion Control. IETF Draft draft-cardwell-iccrg-bbr-congestion-control-00 (Jul 2017)
8. Che, X., Ip, B., Lin, L.: A Survey of Current YouTube Video Characteristics. IEEE Multimedia **22**(2) (April - June 2015)
9. Cisco: Interface and Hardware Component Configuration Guide, Cisco IOS Release 15M&T. Cisco Systems, Inc. (2015), chapter: Rate Based Satellite Control Protocol
10. Claypool, S., Claypool, M., Chung, J., Li, F.: 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)
11. 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)
12. Dong, M., Li, Q., Zarchy, D., Godfrey, P.B., Schapira, M.: 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. Dong, M., Meng, T., Zarchy, D., Arslan, E., Gilad, Y., Godfrey, B., Schapira, M.: 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)
14. Everts, T.: 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)
15. Floyd, S.: Metrics for the Evaluation of Congestion Control Mechanisms. RFC 5166 (March 2008)
16. Ha, S., Rhee, I., Xu, L.: CUBIC: A New TCP-Friendly High-Speed TCP Variant. ACM SIGOPS Operating Systems Review **42**(5) (2008)
17. 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 **60**(2), 58–66 (Jan 2017)
18. Obata, H., Tamehiro, K., Ishida, K.: Experimental Evaluation of TCP-STAR for Satellite Internet over WINDS. In: Proceedings of the International Symposium on Autonomous Decentralized Systems. Tokyo, Japan (Mar 2011)
19. Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., , Yang, F.: The QUIC Transport Protocol: Design and Internet-Scale Deployment. In: Proceedings of the ACM SIGCOMM Conference. Los Angeles, CA, USA (Aug 2017)
20. Turkovic, B., Kuipers, F.A., Uhlig, S.: Interactions Between Congestion Control Algorithms. In: Proceedings of the Network Traffic Measurement and Analysis Conference (TMA). Paris, France (2019)
21. Utsumi, S., Muhammad, S., Zabir, S., Usuki, Y., Takeda, S., Shiratori, N., Katod, Y., Kimb, J.: A New Analytical Model of TCP Hybla for Satellite IP Networks. Journal of Network and Computer Applications **124** (Dec 2018)

22. Wang, Y., Zhao, K., Li, W., Fraire, J., Sun, Z., Fang, Y.: 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)
23. Ware, R., Mukerjee, M.K., Seshan, S., Sherry, J.: Modeling BBR's Interactions with Loss-Based Congestion Control. In: Proceedings of the Internet Measurement Conference (IMC). Amsterdam, Netherlands (Oct 2019)