A Cross-Comparison of Advanced TCP Protocols in High Speed and Satellite Environments

Cesar Marcondes, Jerrid Matthews, Robert Chen, M.Y. Sanadidi, Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, USA 90050
Email: {cesar, matth122, rchen}@cs.ucla.edu

Abstract—In today’s ever-changing network, new challenges arise with the presence of various types of physical links, such as high speed, satellite, and wireless networks. The common solution is to create a new TCP algorithm that is optimized for each challenge encountered. One example of this is TCP Hybla, which was designed to tackle satellite paths. On the other hand, there are other protocols that claim to be more adaptive to different scenarios, like TCP Adaptive Westwood. In this study, we explore the tradeoffs between utilization and coexistence of both these protocols in two networks: a high-speed and a satellite with identical (and large) Bandwidth-Delay Product. Our goal is to make a comparison of the issues solved by both protocols in both of these scenarios.

High-speed networks; satellite networks; congestion control; coexistence; simulation.

I. INTRODUCTION

In the past decade, the Transmission Control Protocol (TCP) has evolved into many different flavors to address various challenging network path conditions. TCP was originally designed to operate in point to point wired and low-speed networks. As the Internet evolved, gigabit speeds and wireless mobile connectivity became the latest trend; however new complexities and issues were introduced to TCP. Therefore innovative schemes were designed to provide a solution to certain issues. For example TCP Hybla [1] which was created to enable RTT fairness in a satellite environment, TCP VenoReno [3], which cope with wireless networks and High Speed TCP (HS-TCP) [4], which maintain TCP scalability over high-speed networks, such as fiber optic. There are also a few proposals that claim to be adaptive to many scenarios, like TCP Adaptive Westwood [2]. In this paper, we put to the test the above TCP Adaptive Westwood claim. At the same time, we also verify issues of co-existence involved in a satellite protocol, like TCP Hybla, when used over a terrestrial high-speed path. Our study allows us to cross-examine issues solved by both protocols in both scenarios.

The remainder of the paper is organized as follows. Section II describes TCP Hybla and TCP-Adaptive Westwood. Section III describes the simulation setup, based on a sophisticated parking lot topology from which we replace the last segment by either a high-speed or a satellite network with exactly the same bandwidth delay product (maximum number of packets unacknowledged in-flight). Following this, in Section IV, we present simulation results that cross-compare the protocols in both scenarios. The goal is to shed light on issues such as co-existence with TCP NewReno. Moreover, we also analyze fading effects on satellite environments to check robustness of the protocol. We conclude and provide final remarks in Section V.

II. RELATED WORK

This section explains TCP Hybla [1], TCP-Adaptive Westwood [2], and the TCP evaluation suite used in details.

A. TCP Hybla

TCP Hybla [1] aims to improve the performance of network connections with long round trip times (RTTs). Satellite links and some wireless links experience long propagation delays, which increase the RTTs experienced by TCP sources. Some of the more important negative effects of long RTTs are reduction of congestion window growth rate, and multiple losses in one congestion window. TCP Hybla contains several solutions which address these problems. To overcome the problem of slow congestion window increase, TCP Hybla attempts to remove the reliance on RTT from the congestion window update algorithm. This is accomplished by adjusting the congestion window size to a normalized ratio of the previous congestion window size. Equations (1) and (2) define the congestion window update mechanism in TCP Hybla.

\[
\rho = \frac{RTT}{RTT_0} \quad (1)
\]

\[
W^H(t) = \begin{cases} 
\rho^{\frac{\rho}{RTT}}, & 0 < t < t_{\gamma,0}, SS \\
\rho \left( \frac{t - t_{\gamma,0}}{RTT} + \gamma \right), & t \geq t_{\gamma,0}, CA 
\end{cases} \quad (2)
\]
The result of TCP Hybla’s new congestion window update mechanism is a larger average cwnd.

TCP Hybla uses the SACK option to help alleviate possible packet losses resulting in the same window. By using the SACK option, the sender knows exactly which packets it has sent successfully, and allows the sender to retransmit more than one packet per RTT. Furthermore, TCP Hybla uses timestamps to track packets, rather than using the traditional approach of waiting for the ACK.

Another interesting enhancement is packet spacing during transmission. TCP Hybla’s typical large congestion window size may result in bursty transmissions. This burstiness could be “smoothed out” by more intermittent transmissions and spacing each transmission out over a period of time. TCP Hybla’s enhancements make it an ideal choice for a satellite path. The protocol function well with long RTTs experienced in a typical satellite network’s path and is aggressive enough with its congestion window update algorithm to fill the link’s bandwidth capacity quickly.

B. TCP Adaptive Westwood

TCP-AW is a combination of the best features from both TCP Westwood ABSE (TCP-ABSE) [5] and TCP Adaptive Reno (TCP-AReno) [6]. TCP-ABSE’s best feature is its eligible rate estimation (ERE) mechanism, which helps predict imminent congestion. Essentially, the congestion window is adjusted according to the network congestion rate, and is not based on the traditional delay model. TCP-AReno’s best feature is its use of packet loss interval time to adjust the congestion window, instead of using bandwidth estimation. This approach improves RTT-fairness even when accurate bandwidth estimation is not available. The following explains the many pieces that compose the algorithm.

1) Safe Slow Start

In order to improve transmission times of relatively small communications, AW introduced a safe slow start mechanism that uses an estimate of buffer occupancy to moderate the sending rate increase during start-up (Algorithm 1). The idea is to introduce artificial pauses during slow start microbursts.

If (Slow Start AND (ERE * (RTT - RTTmin)) > 0) then

- If ((W mod 2) == 0) then
  \[ W = W + 1 \]
- else
  skip increment / create a pause

2) Incipient Congestion Detection

Leveraging the congestion detection via rate estimation in Westwood, it also proposes an extension to detect incipient congestion before RTT actually increases. The modified RTT value, \( RTT^{\text{Compensate}} \), is provided as follows:

\[
RTT^{\text{Compensate}} = RTT + \max \left( \left\{ \frac{W}{RTT} \right\} \frac{1}{EPE} - 1 \right) \cdot \gamma, 0
\]

In steady state equilibrium, \( W/RTT \) is expected to be equal to ERE, thus \( RTT^{\text{Compensate}} = RTT \). When a new congestion period starts, it would take longer time for RTT to increase large enough to detect congestion because it has to wait for the queue to grow. On the other hand, ACK arrival interval changes quickly after the beginning of congestion, thus ERE changes promptly. In this situation ERE becomes small before \( W/RTT \) becomes small, thus \( RTT^{\text{Compensate}} > RTT \) and the congestion control algorithm reacts to the congestion more promptly.

3) Light-Weight ERE calculation

Additionally, in order to make the combined protocol more effective in very high-speed environment, it proposed a simplified rate estimation scheme that resembles TCPW-ABSE but does not need a large ACK history and has a simplified filter.

\[
ERE_{\text{sample}} = \frac{\sum d_j}{T_k}, \quad \sigma = 4W + \frac{\sum d_j}{T_k}
\]

\[
ERE_k = \alpha ERE_{k-1} + (1 - \alpha) ERE_{\text{sample}}
\]

4) Control Equation

The control equation uses packet loss interval time, rather than ERE, because measurement error of ERE tends to be biased regarding RTT, which can result in biased RTT fairness.

\[
W_{\text{def}} = \left( \alpha L - W \frac{RTT - RTT_{\text{min}}}{RTT} \right) \frac{RTT_{\text{min}}}{RTT_{\text{ref}}}
\]

\[
W = W + W_{\text{def}}, \quad \text{if } W_{\text{def}} \leq 0
\]

\[
W = W + W_{\text{def}} e^{-c}, \quad \text{if } W_{\text{def}} > 0
\]

\[
c = \frac{RTT - RTT_{\text{min}}}{RTT_{\text{cong}} - RTT_{\text{min}}}
\]

In equation, (10), \( c (0 \leq c \leq 1) \) is an estimated congestion level and \( RTT_{\text{cong}} \) is an expected RTT value when a packet loss occurs. In addition, \( L \) is an average “non-congested” time period between two packet losses. It follows the idea that a packet loss interval reflects long-term congestion level, which is proposed by CUBIC. We extended the idea by using “non-congested” (\( c < 0.4 \)) time periods rather than using the entire time period to estimate congestion level more properly. The parameter alpha is the coefficient for congestion window increase and it is also the basic parameter that determines the aggressiveness of the protocol. The congestion window increase is mitigated exponentially as \( c \) increases. \( RTT_{\text{ref}} \) is a parameter that determines control gain of the function.

5) Final tunings

Several final touches were added in the algorithm, i) the congestion window reduction is similar to TCP-AdaptiveReno, thus it is based on the congestion measurement with a lower bound equal to NewReno, i.e. \( \max(W/(1+c), W/2) \), this provides a natural loss...
discriminator as well. ii) A variable of the parameter that counts time intervals (alpha = 0.2) based on “non-congestion”, can be fine tuned as a friendliness parameter to obtain more network utilization at the cost of increased aggressiveness, or decreased friendliness.

C. TCP Evaluation

Our evaluation work is based on a set of simulation tools collectively called “TCP Evaluation Suite” [7]. The tool allows the creation of complex network topologies and traffic characteristics. It is the proper tool for studying co-existence problems in TCP Standard NewReno because it supports a mode where half of the flows running on the network can be replaced by advanced TCP schemes while keeping the same generated statistical distribution of (1) file start-up time and (2) file sizes and same network topology.

The output of the tools can be organized to extract automatically the aggregate throughput utilization per-flow and relate it to the flow RTT, number of hops, and link utilization among other features. It is also possible to obtain a so-called “efficiency-friendliness” tradeoff graph [6] – a 2D graph that provides a clear picture of performance improvement of the advanced TCP variant and performance degradation in relationship to standard TCP NewReno.

III. NS2 SIMULATION SETUP

A. Simulation Setup

In our simulations, we used ns-2.31 [8], the TCP Linux patch [9] – where all protocols have native support for SACK, the TCP Evaluation Suite described above and several statistical seeds.

Regarding the topologies, both the ns-2 simulation scenarios (Figures 1 and 2) have a “common” parking lot “wired” network segment, that provides cross traffic mimicking a real world environment with elephant and mice TCP flows. Each wired link in the parking lot topology has a 1 Gbps wired connection to the core and every link has exponentially distributed 15 msec propagation delays.

This “common” network has 130 flows where 30 are long lived transferring DVD sized files with thinking time of 2 min average, and 100 short lived flows of 100KB with thinking time of 1 sec. All these flows use the standard TCP NewReno. The sources and destinations are uniformly distributed, thus, the central link in the network (between R2 and R3) tends to have more flows. It has an average of 10 flows passing through in each direction.

There are a total of five routers that coordinate traffic in the network; each labeled respectively (RT1 to RT5). RT5 is the router, which we connect a separate network segment that we vary (either a high-speed or a satellite network) in our study. All buffer queues in the setup are the same 8 MB large, following normal router recommendations.

This segment contains several mice flows and 2 special groups of long-lived flows, the experimental group and the control group. The experimental group is a set of flows (half of the total flows passing through RT5) that host an advanced variant, either TCP-AW or TCP Hybla. The control flows will be all the other flows, which use TCP NewReno.

We perform a qualitative comparison of the protocols by first generating a simulation execution where the experimental group of flows uses TCP NewReno. This forms the baseline and afterwards, in a separate simulation, we replace these with the new variant.

![Figure 1. Satellite network simulation topology.](image)

![Figure 2. High speed network simulation topology.](image)

The access links of the nodes in the high-speed network segment connected to RT5 are 600Mbps and average 50ms each (this small variability avoids phase effects). Meanwhile, the simulated satellite network has clients using dedicated satellite slots and thus no contention among the nodes. The satellite downlink for each client has an assured capacity of 60Mbps and 500ms round trip time delay (RTT), plus a small variance due to different geographical locations. For each scenario, the links maintain an equal bandwidth delay product (BDP) to ensure a comparable test environment.

B. Testing of Protocols

For each network topology scenario, we will test the following TCP protocols: TCP NewReno, TCP-AW, and TCP-Hybla. Thus, a combination of 6 simulations sets. From the topology described in figure 1 above, the experimental group will vary to allow comparison (by using the same exact file and start-up demand). The three protocol comparisons will be: Reno vs. Reno (baseline), Reno vs. TCP-AW, and Reno vs. TCP-Hybla. The
“common” wired network will be always running legacy TCP Reno protocol (130 flows), while the high-speed and satellite network connected to RT5 will be running the various protocols depending on the group they belong (4 flows – 2 flows from the experimental group and 2 from control group). We summarize the set up in the following Table 1. We will present next the analysis of the performance of each protocol for each scenario.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Traffic Mix</th>
<th>Source/Dest. Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 – R4</td>
<td>Standard Reno</td>
<td>Uniform</td>
</tr>
<tr>
<td></td>
<td>30 long lived</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 short lived</td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td>Study Group</td>
<td>Each studied flows “exit”</td>
</tr>
<tr>
<td></td>
<td>Advanced TCP</td>
<td>on each router R1 to R4</td>
</tr>
<tr>
<td></td>
<td>2 Standard Reno</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 short lived</td>
<td>Uniform</td>
</tr>
</tbody>
</table>

Table 1. Experimental Setup Description

IV. RESULTS AND ANALYSIS

A. RTT-Fairness versus Co-Existence Issues

One classical problem that users behind a satellite link face is the dependency of their rate on the RTT pertinent to a flow. Due to this unfair nature of the Standard TCP NewReno, satellite users feel the need to use specialized protocols like TCP Hybla. In Figure 3, we present results of the groups of flows that we replace with the new protocol (the 2 flows of experimental group – R5). The first set of bars present the aggregate goodput when the flows are high-speed, low-RTT. The second set of bars represents the aggregate when the flows are from the satellite network.

Figure 3. Aggregate Throughput (Experimental Group)

One can verify that by changing the final network-leg R5 to satellite (from topology 2 to 1, same BDP) and keeping the same traffic load, all TCP flows suffer to keep the same rate. The aggregate rate drops from 487Mbps to 38Mbps in the baseline (RENO+RENO). The problem is two-fold: 1) some long-lived flows in R3 and R4 experience a capture effect hurting the flows that experience the huge delays of satellite, they are able to reach almost 940 Mbps in some cases. On the other hand, whenever the flows are replaced by advanced congestion control like TCP-AW and TCP-Hybla, the situation ameliorates: 2) the case of Standard Reno, the algorithm should not able to reach good throughput even if it were alone because the simulation duration of 600 secs is not long enough to reach the maximum, as we will see in the next section.

In the case of the high-speed network-leg, the goodput boost was from 487Mbps to approximately 820Mbps in both cases of TCP-AW and TCP-Hybla, which represents an improvement of 166%. On the other hand, considering the satellite scenario, we obtained improvements in the order of 307% using TCP Hybla and 234% using TCP-AW. This was expected due to Hybla running in its natural satellite environment.

We also study possible shortcomings in other aspects like co-existence for the advanced protocols. In particular, if we compare the control group, the set of flows that we have not changed the protocol in the mixed scenario (AW+RENO or HYBLA+RENO) and that share the same long-delay path characteristic, we can verify some level of degradation (Figure 4) compared to the baseline RENO + RENO. It is important here to differentiate fairness from co-existence, since we are not trying to establish what is fair, but just measure the amount of impact of a new algorithm on the performance of legacy traffic that passes along with the modified flows. In this regard, the use of the exact same demand allows us to comparatively analyze this issue.

Figure 4. Relative Degradation (Control Group)

For the high-speed scenario, TCP-AW remarkably improved the aggregate throughput of the control group by 26%. This is due to the delay-based nature of the protocol. The protocol feels the presence of incoming TCP standard Reno flows, and then reacts accordingly.
while being agile enough to grab the extra bandwidth whenever the bottleneck is less loaded. On the other hand, TCP Hybla strives to co-exist with TCP NewReno by reducing the rate of the control group by 30% (First set of bars). It is also possible to explain this issue if we consider that the TCP Hybla algorithm scales by being more aggressive like Scalable TCP without getting feedback prior to the aggressiveness. The second set of bars is the satellite environment where TCP Hybla had performed better; the reduction was from the baseline from 100% to 84% in the Hybla case (16% degradation). On the other hand, TCP-AW improved again from 100% by 112% (12% improvement). This shows the adaptability and refined co-existence of Adaptive Westwood in both high speed and satellite environments.

By quantifying the amount of degradation on each scenario side-by-side, we can verify in Figure 4, that the aggregate throughput of the control group flows drop by half for the TCP-Hybla protocol on the high-speed network scenario. This result shows unexpected co-existence fairness that a protocol optimized for satellite, used on a high-speed network, can bring.

Finally, we analyze the impact of the study group on the rest of the network (all the remaining 130 flows). These results are just indicators of the overall performance of the network in both cases. Figure 5 shows the performance of the 30 elephant long lived flows that pass uniformly through each group of core segments of the network (R1-R4). Consider that since all links are 1Gbps, it is expected that the aggregate of all long lived flows remain 6Gbps on average. Also, as it happens in the Internet, the elephants are responsible for delivering the majority of the data transferred: 6 Gbps aggregate goodput in comparison to 150 Mbps of the mice flows (Figure 6).

In terms of how the study flows (the 4 long-lived flows of the experimental and control group) impact the rest of the network flows, we see small improvements in both cases. In the case of the long-lived flows (elephants), we see a slightly better performance of Hybla over TCP-AW. This may be due to the pacing feature that TCP-Hybla uses to affect its flows. Pacing helps in reducing congestion errors in general since it distributes the traffic evenly over the congestion epoch, and therefore, cross-traffic experiences less burstiness events from the study flows.

On the other hand, we see a different pattern in the mice short-lived flows (Figure 6). The pattern is complementary to the elephant flows pattern, which indicates that the improvement of elephant goodput pays a higher price on the mice flows. TCP-AW shows a good balance on both.

B. Robustness to the Challenging Satellite Environment

TCP Hybla has advantages in terms of providing ideal RTT-fairness and better goodput in satellite networks. TCP-AW, in the results above, provided a more balanced performance so far. However, the challenge to adapt to different environments, like the satellite, also includes robustness to special features. For example, satellite links, depending on weather (storms and solar flares) and antenna gain configuration, may experience losses not due to congestion. These types of losses were recently profiled in [10] and could be modeled by a uniform distribution with a small packet loss probability (between 0.05% and 0.25% PER).
In order to study this point, we set a simplified topology as shown in Figure 7. In this scenario, we run 4 flows simultaneously over a satellite repeater randomly dropping packets. The characteristics of the link are 20Mbps upstream and downstream, 700 msec propagation delays, and the simulation has duration of 2000 seconds.

Two of the flows were legacy TCP NewReno, and two of the other flows were the advanced protocol (either Hybla or AW). It is important to point out that TCP NewReno flows have the same bad performance under error by themselves (alone) because the errors cap their maximum transmission rate; this issue can be seen in the time-series goodput plot of Figure 8(a). The plot shows a cumulative throughput averaged over the time experienced so far and all four standard Reno flows obtain 1.2 Mbps approximately under 0.05% errors.

Whenever we replace two of the flows with the new congestion control algorithms we see a substantial improvement. In the case of TCP-Hybla, the replaced flows reached on average 8Mbps while TCP-AW reaches 5Mbps. On the other hand, if we pay attention to the performance of the legacy standard Reno flows that share the lossy path, we can see a different story. In the case of TCP-Hybla, the standard TCP Reno flows experience starvation due to the lack of a more agile feedback system, while with TCP-AW, the standard Reno flows experience good throughput as if they were alone.

C. Analysing the Loss Discrimination Feature

There are two interesting things to point out regarding the good performance of TCP-AW under errors: (1) agile response due to the delay feedback loop and, (2) embedded congestion loss discrimination. TCP-AW has this latter feature to cope with loss under non-congestion regimes. This is accomplished by probing the congestion level at the point of loss. We investigate all loss periods and the congestion levels recorded at those points, then we study the distribution obtained in that case.

Figure 9 shows the congestion level presented over an integer scale from 0 to 256. This scale can be easily converted to percentages by normalization. We use this due to Linux kernel code constraints, namely some lack of support for floating point numbers. As fading effects are increased (PER from 0.05% to 0.25%), the numbers of congestion epochs caused by traffic overflow decreases (numbers close to 256 or 100%) from 30% of the losses to less than 10%. This is due to reduced traffic intensity and the consequent inability to saturate the link under increasing PER. On the other hand, there is a large amount of congestion level events between 0 and < 256. These are the cases where the buffer is not completely full and a loss happens. The 0.25% case can be verified on Fig.10.
The results in the range > 0 (clearly, buffer partially full) and < 256 (buffer full) fit well on an Erlang distribution and we are investigating ways to obtain the amount of random errors from this distribution.

V. CONCLUSIONS

In this paper, we perform a cross-comparison of equal BDP scenarios and very different network topologies based on high-speed networks and satellite networks. We experimented with two advanced TCP algorithms aimed at optimizing the throughput. We then analyze issues related to improving RTT-fairness and co-existence with standard NewReno. Our results show that there are tradeoffs between utilization and co-existence. TCP Adaptive Westwood shows good throughput in High Speed networks and performs safely both on high-speed and satellite networks. On the other hand, TCP Hybla has better throughput on satellite networks. Moreover, if we delve into challenging characteristics of satellite networks such as fading effects, we can verify that TCP Adaptive Westwood obtains a substantial improvement in co-existence due to its embedded loss discriminator component.

REFERENCES