Models of TCP in High-BDP Environments and Their Experimental Validation

Gayane Vardoyan  
University of Massachusetts Amherst  
Email: gvardoyan@cs.umass.edu

Nageswara S. V. Rao  
Oak Ridge National Laboratory  
Email: raons@ornl.gov

Don Towsley  
University of Massachusetts Amherst  
Email: towsley@cs.umass.edu

Abstract—In recent years, there has been a steady growth in network bandwidths. This is especially true in scientific and big data environments, where high bandwidth-delay products (BDPs) are common. It is well-understood that legacy TCP (e.g., TCP Reno) is not appropriate for such environments, and several TCP variants were developed to address this shortcoming. These variants, including CUBIC, STCP, and H-TCP, have been studied in some empirical contexts, and some analytical models exist for CUBIC and STCP. However, since these studies were conducted, BDPs further increased, and new bulk data transfer methods have emerged that utilize parallel TCP streams. In view of these new developments, it is imperative to revisit the question: ‘Which congestion control algorithms are best adapted to current networking environments?’ In order to answer this question, (i) we create a general theoretical framework within which to develop mathematical models of TCP variants that account for finite buffer sizes, maximum window constraints, and parallel TCP streams; (ii) we validate the models using measurements collected over a high-bandwidth testbed and achieve low prediction errors; (iii) we find that CUBIC and H-TCP outperform STCP, especially when multiple streams are used.

I. INTRODUCTION

The congestion collapse in the ARPANET during the late 1980s prompted the adoption of what is now referred to as legacy Transmission Control Protocol (TCP). Later, the advent of high-BDP networks stimulated the development of several TCP variants to overcome the inefficiencies of legacy TCP. These include HSTCP (HighSpeed TCP) [1], FAST (FAST AQM Scalable TCP) [2], BIC (Binary Increase Congestion control) TCP [3], STCP (Scalable TCP) [4], CUBIC TCP [5], and H-TCP (Hamilton TCP) [6]. To an extent, all of these Congestion Avoidance (CA) algorithms are successful in improving bandwidth utilization, although some exhibit better fairness, friendliness, and convergence properties than others. Several empirical studies of these algorithms exist ([7], [8], [9], [10], [11], among others), and although CUBIC is currently the default in Linux kernels, as of yet, there is no definitive consensus on which CA algorithm is best.

Adding to this uncertainty, many other studies rely either on simulations or networks with relatively low bandwidths compared to current High-Speed Network (HSN) environments. Today, 10 Gbps Wide Area Network (WAN) links are not uncommon between large-scale datacenters and computing facilities. An example is the Extreme Science and Engineering Discovery Environment (XSEDE) [12], a collection of supercomputing resources spanning several universities, national labs, and other research institutions, all interconnected by 10 Gbps (and sometimes faster) links. XSEDE offers tools like Globus GridFTP [13] and sftp for data transfers (note that FTP relies on TCP for reliable data movement), and such bulk data transfer mechanisms are becoming increasingly popular in HSNs.

Since BDPs have grown significantly and new bulk data transfer protocols that use parallel TCP streams have been introduced, it is important to reevaluate TCP both analytically and empirically. In this paper, we present a framework within which we derive analytical models for TCP variants. Unlike prior work, where models are studied in isolation for each TCP variant, our framework is not variant-specific and can be used to model a range of loss-based TCP variants. We use measurements on an experimental testbed to motivate and validate these models.

In some environments, researchers share WAN links for data transfers. The drawback is that TCP streams belonging to different users may compete for bandwidth and other resources. An alternate method of transferring data is through the use of virtual circuits (VCs) dedicated to a single user or application. One example is the On-Demand Secure Circuits and Advance Reservation System (OSCARS) [14], which allows users to reserve high-bandwidth VCs for guaranteed performance. According to the Energy Sciences Network, more than 50 research networks deploy OSCARS, including the US LHC Network; and OSCARS VCs carry half of the Department of Energy’s (DOE) science traffic. The increasing popularity of VC-based data transfer options serves as excellent motivation to study TCP behavior in a controlled setting. Moreover, it provides an incentive to examine scenarios with only a fixed, rather than a dynamically changing, number of TCP connections.

Most protocols that strive to achieve efficient bulk data transfers do so by providing features that allow users to tune them as they see fit. For example, GridFTP supports both parallelism and concurrency (parallel streams use one socket and concurrent streams use separate sockets; henceforth, we use these terms interchangeably). It has become clear that the reason for the emergence of tools like GridFTP is the fact that TCP is not able to keep up with the demands of today’s high-BDP networks. In this paper, we develop a framework that enables us to study the performance of protocols like GridFTP in dedicated high-BDP networks. To do so, we concentrate on
a possible underlying root cause of poor bandwidth utilization: the congestion control (CC) algorithms used by all TCP-based data transfer tools.

We take two approaches in an attempt to understand the dynamics of TCP in modern environments: (i) first-principled modelling and (ii) measurement-based. We create robust, detailed analytical models of variants of the protocol. At the same time, we evaluate these variants on a dedicated network link, in a controlled setting that allows us to emulate diverse experimental configurations while removing any interference (e.g. I/O, background traffic) that could obscure TCP’s intended behavior. We collect detailed measurements of memory-to-memory transfers on two different testbed configurations using iperf and tcpprobe, for three different TCP variants.

The main contributions of this work are:

- A general and comprehensive framework for modelling a diverse set of congestion control algorithms. The framework encompasses not only congestion avoidance, but also the slow-start mechanism. The latter takes into consideration the heuristic guidelines imposed by Hybrid slow-start (HyStart) [15], which is the implementation of slow-start used in current Linux kernels.
- A validation of these models using an extensive set of measurements.
- Last, we observe from our measurements that (i) CUBIC and H-TCP are comparable in terms of average throughput, while they both outperform STCP, and (ii) TCP performance benefits from the presence of a well-designed physical layer (e.g. SONET).

We use only first principles to model the performance of each TCP variant in terms of its average throughput as a function of round trip time (RTT). The models also accept link capacity, buffer size, transfer size, number of parallel streams, and maximum congestion window (cwnd) size as parameters. Using measurements, we validate the models and compare the performance of the TCP variants. To our knowledge, this work constitutes the first careful measurement-based modelling study of TCP congestion control algorithms in a high-BDP/HSN setting.

The remainder of the paper is organized as follows. We describe the three variants and related work in Section II. In Section III, we describe the testbed, measurement collection, and first-hand observations from the collected data. In Section IV, we delve into the analytical framework for slow-start, congestion avoidance, and each of the variants separately, presenting closed-form expressions for sending rate where possible. In Section V, we present and validate our results. Finally, we conclude the paper in Section VI.

II. BACKGROUND

A. TCP VARIANTS

We study CUBIC because it is the most commonly used variant in current HSN networks and the default CA algorithm in the Linux kernel. Unlike most TCP variants, CUBIC is not an acknowledgement (ACK)-based algorithm. Instead, CUBIC’s cwnd is a cubic function of time since the last congestion event such that the inflection point is the maximum window size immediately before the most recent loss occurred.

We also study STCP because it was developed within the optimization-based framework proposed in [16]. STCP is a multiplicative increase, multiplicative decrease (MIMD) \(^1\) algorithm with the following response functions:

\[
cwnd \leftarrow cwnd + a
\]

for every ACK received, where the increase factor \(a\) is usually set to 0.01. Upon loss detection,

\[
cwnd \leftarrow b \ast cwnd
\]

Usually, \(b = 0.875\) for STCP.

Finally, we study H-TCP because of its favorable fairness and convergence properties [8]. H-TCP is an ACK-based generalized additive increase multiplicative decrease (AIMD) algorithm whose additive increase factor \(a\) is a function of the time \(t\) since the last congestion event. Specifically, \(a\) is defined as:

\[
a \leftarrow 2(1-b)a(t)
\]

where \(a(t)\) is

\[
a(t) = \begin{cases} 
1 & t \leq \Delta_L \\
1 + 10(t - \Delta_L) + \left(\frac{t - \Delta_L}{2}\right)^2 & t > \Delta_L
\end{cases}
\]

\(\Delta_L\) is usually set to one second so that for small congestion epochs, H-TCP behaves like standard TCP. H-TCP’s decrease factor \(b\) is defined as

\[
b \leftarrow \frac{RTT_{\text{min}}}{RTT_{\text{max}}}, \quad b \in [0.5, 0.8]
\]

where \(RTT_{\text{min}}\) and \(RTT_{\text{max}}\) are the minimum and maximum measured RTTs of a flow. Upon loss detection, cwnd is updated as follows:

\[
cwnd \leftarrow b \ast cwnd
\]

B. Related Work

There exist a number of analytical studies for modelling TCP. Kelly proposed an optimization-based framework for studying and designing CA algorithms in [16], where STCP was an output. In [17], Srikant presented a simple analysis of Jacobson’s TCP CC algorithm. The derivation sets the maximum window constraint to the sum of the BDP and the size of the buffer. Our analysis is more refined in that it takes into consideration two different maximum window constraints, as discussed in Section IV. A model for slow-start is also presented in [17]. We extend this model by considering the latest version of slow-start currently in use by Linux kernels.

In [18], Misra et al. model TCP throughput using stochastic differential equations. El Khoury et al. [19] present a model for STCP that includes buffer size as a parameter, but only in the

\(^1\)Note that although the per-ACK update rule for STCP is additive, this CA algorithm is MIMD at the RTT level.
case of a very small buffer. In addition, they rely only on ns-2 simulations for validation. Bao et al. propose Markov chain models for average CUBIC throughput, but for a wireless environment [20]. Moreover, they do not directly account for buffer constraints. Leith et al. present empirical evidence that H-TCP fares well in bandwidth utilization compared to other TCP variants [21], but the protocol’s CC dynamics have not been analyzed in-depth.

There are some empirical studies that explore the behavior of TCP with multiple concurrent flows. Morris looks at a number of performance metrics using simulations and real packet traces, but does not explore different TCP variants [22]. Yu et al. compare the performance of three open-source big data transfer protocols in [23] using memory-to-memory transfers on a 10 Gbps international HSN. Bateman et al. compare different TCP variants for fairness at high speeds using ns-2 and Linux [24]. As far as we know, no previous work attempts to model TCP with multiple flows using first principles.

III. MEASUREMENTS

A. Emulation Testbed

Our testbed consists of two types of Linux hosts: 32-core and 48-core HP ProLiant servers, each with Broadcom 10 GigE NICs, running Linux 2.6 kernel (CentOS release 6.6). It also consists of ANUE OC192 and IXIA 10 GigE hardware connection emulators, and a 10 Gbps Force10 E300 WAN-LAN switch. Two separate configurations are utilized for 10 GigE and SONET measurements. These hardware connection emulators transport the physical packets between hosts, delaying them during transit by an amount specified at configuration. This process closely matches the effects of physical connections, particularly, the TCP dynamics of hosts connected to them, which in turn determine the throughput rates achieved. They more closely capture the real-time TCP dynamics compared to packet-level simulators (such as ns-3 and OPnet) that are typically driven by discrete “packet delivery events”.

![Fig. 1: Two testbed configurations for dedicated 10 Gbps connections.](image)

C. Empirical Observations

It is interesting to note the differences in measurements produced by the SONET and 10 GigE links. In general, we observe that the data obtained from SONET is well-behaved and more deterministic than that collected from 10 GigE. Figure 2 illustrates the stark contrast in TCP behavior between these two testbeds. The evolution of cwnd over time exhibits a consistent sawtooth pattern for SONET, whereas the 10 GigE transfer experiences non-uniformly-spaced losses and seemingly flat regions for smaller RTTs. The data shown is for STCP, for RTTs of 91.6 ms and 183 ms, although the results are consistent across different variants and other RTT values. These differences are important: they mean that for accurate predictions, the models must account for both the frequency of losses and the shapes of the cwnd curves. Section IV goes into specifics on how this can be accomplished.

Figures 2(c) and 2(d) reveal another difference between these two modalities: 10 GigE has more aggregate buffer space than SONET. Visually, buffering occurs when cwnd grows as
throughput remains relatively constant. The size of the buffer
corresponds to the white space between the two curves, and
since it is more prevalent in 10 GigE data, we know that this
configuration has the larger buffer size. Figure 3 presents a
comparison of the three variants in terms of their average
throughput. The dataset was collected over the 10 GigE link.
Each memory-to-memory flow was active until it transferred
10 GB of data. It is evident that for a small number of streams,
the three variants perform almost equally well. However, as the
number of flows grows, the differences in performance become
more notable: CUBIC and H-TCP significantly outperform
STCP with ten parallel flows.

IV. ANALYSIS

We first present a model for slow-start and discuss how it
can accommodate an important component of HyStart. Then,
we present single-stream analytical models for two different
types of loss-based congestion avoidance mechanisms: (i)
those that are ACK-based and (ii) those that grow cwnd as a
function of time since last loss (referred to as TSL-based
variants). STCP and H-TCP are ACK-based, while BIC and
CUBIC are examples of TSL-based algorithms. Finally, we
demonstrate how single-stream models can easily be extended
to incorporate multiple parallel TCP streams.

Table I contains definitions of variables used in this section.
We assume that we know the link capacity $C$, aggregate buffer
size $\beta$, round-trip-time $\tau$, size of the transfer $F$, and increase
or decrease parameters. The minimum RTT, $\tau$, is the delay
measured before the buffer of size $\beta$ begins to fill up. This
can be measured by running a simple ping command between
the source and receiver. $\beta$ represents the aggregate size of
all buffers present on the link (e.g. on routers and NICs). An
illustration is shown in Figure 4. $W_{max}$ is a constraint on cwnd
typically imposed by the receiver (receive-window).

A. Slow-Start

In traditional slow-start, cwnd approximately doubles every
round trip time of length $\tau$ in which loss is not detected:

$$w(t) = 2^{t/\tau}$$
The amount of data transferred during one such CA epoch, and TCP variant). The goal is to derive expressions for

\[ S = \frac{N_s + kN_a}{T_s + kT_a}. \]

Then, throughput \( \Theta \) can be estimated as

\[ \Theta = S(1 - p) \]

where \( p \) is the packet loss probability for a TCP flow. For small values of \( p \),

\[ \Theta \approx S. \]

Next, we discuss how to obtain \( N_a \) and \( T_a \) for various cases.

**Case 1:** \( W_{max} \leq C\tau \). In Phase I, \( cwnd \) grows until either the link capacity or \( W_{max} \) is reached (if \( W_{max} \leq C\tau \)). In the latter case, \( cwnd \) remains flat until the transfer ends, buffer overflow never occurs, and there is no second phase. The amount of data transferred in this case is \( N_a = F - N_s \) (file size minus the amount of data transferred in slow-start), and since the sending rate is capped at \( W_{max}\/\tau \), it takes \( T_a = N_a\tau/W_{max} \) time until the transfer ends.

**Case 2:** \( W_{max} > C\tau \). In this case, each congestion epoch will have two phases.

**Phase I:** The methodology for obtaining \( N_0 \) and \( T_0 \) for Phase I is the same for ACK-based and TSL-based variants. Given a congestion window function \( w(t) \), we know that:

\[ w(T_0) = C\tau. \]

Using this relation, we can solve for \( T_0 \): either directly (if \( C\tau < W_{max} < W_m \) as in Figure 5b) or in terms of \( W_m \), if \( W_{max} \geq W_m \) as in Figure 5a). In order to distinguish between 5b and 5a, we must first solve for \( W_m \) (discussed below). The amount of data transferred in Phase I is:

\[ N_0 = \frac{1}{\tau} \int_0^{T_0} w(t)dt. \]

**Phase II:** In Phase II, the sending rate is capped at \( C \), and the buffer begins to fill up. We first solve for \( W_m \) with the hypothesis that 5a is the correct representation of our congestion epoch. Once a value for \( W_m \) is obtained, this hypothesis can be rejected or accepted by comparing \( W_m \) with \( W_{max} \). However, how we solve for \( W_m \) depends on the type of CA mechanism.

1) **ACK-Based CA:** The following ordinary differential equation (ODE) describes the behavior for ACK-based TCP variants:

\[ \frac{\partial w}{\partial t} = \frac{\partial w}{\partial A} \frac{\partial A}{\partial t}. \]  

where \( A \) represents an acknowledgement. Since the sending rate is \( C \), this is also the rate at which ACKs are being received:

\[ \frac{\partial A}{\partial t} = C. \]
For every ACK, $cwnd$ increases by $a$, so
\[ \frac{\partial w}{\partial A} = a. \]
Hence,\[ \frac{\partial w}{\partial t} = C a. \] (5)
The initial condition for this ODE is $w(0) = C \tau$. The ODE can now be solved to obtain a window function for Phase II, $w_2(t)$.

2) TSL-Based CA: In a TSL-Based variant, $cwnd$ grows identically in Phase II as it does in Phase I. Hence,\[ w_2(t) = w(t). \]
Now that we have $w_2(t)$, we make the following useful observations:
\begin{align*}
  w_2(T_m) &= W_m, \quad (6) \\
  N_m &= \frac{1}{\tau} \int_0^{T_m} w_2(t) dt, \quad (7) \\
  N_m - C T_m - \beta &= 0. \quad (8)
\end{align*}
Using (6), we can solve for the duration of Phase II, $T_m$, in terms of $W_m$. The amount of data transferred in Phase II, $N_m$, is given by (7), also in terms of $W_m$. Finally, we obtain a value for $W_m$ by finding the roots of (8) and selecting the appropriate root (subject to the constraints that $W_m \in \mathcal{R}$ and $W_m \geq C \tau$).

Subcase 1: $W_m \leq W_{max}$. In this case, we simply substitute the value of $W_m$ into the expressions for $T_m$, $N_m$, $T_0$ and $N_0$. Then,
\begin{align*}
  N_a &= N_0 + N_m, \\
  T_a &= T_0 + T_m.
\end{align*}
Subcase 2: $W_m > W_{max}$. In this case, 5b correctly depicts the shape of the congestion epoch. Since $W_{max}$ is known, we can solve for $T_{max}$, the duration of sub-phase I of Phase II, directly:
\[ w_2(T_{max}) = W_{max}. \]
The amount of data transferred during this sub-phase is
\[ N_{max} = \frac{1}{\tau} \int_0^{T_{max}} w_2(t) dt. \]
The only unknown left is $T_{\beta}$, the duration of sub-phase II, and $N_{\beta}$, the amount of data transferred during that interval. However, we know that during sub-phase II, the sending rate is capped at $W_{max}/\tau$, so
\[ N_{\beta} = \frac{W_{max}}{\tau} T_{\beta}. \]
We can solve the following equation for $T_{\beta}$:
\[ N_{max} + N_{\beta} - C (T_{max} + T_{\beta}) - \beta &= 0. \]
Then,
\begin{align*}
  N_a &= N_0 + N_{max} + N_{\beta}, \\
  T_a &= T_0 + T_{max} + T_{\beta}.
\end{align*}

C. Examples
We demonstrate the versatility of the generalized modelling framework described in previous sections through three examples: STCP and H-TCP (both ACK-based variants) and CUBIC (a TSL-based variant).

1) STCP: Since the additive increase factor for STCP is a constant, the solution to the ODE in (5) is simply
\[ w_2(t) = C(at + \tau). \] (9)
Using this equation, we obtain:
\begin{align*}
  T_m &= \frac{W_m - C \tau}{C a}, \\
  N_m &= \frac{W_m^2 - (C \tau)^2}{2 C a \tau}, \\
  W_m &= C \tau + \sqrt{2 C a \tau \beta}.
\end{align*}
If $W_m \leq W_{\text{max}}$, we have everything we need for Phase II. Otherwise, we use $W_{\text{max}}$ instead of $W_m$ in the expressions for $T_m$ and $N_m$; this gives us $T_{\text{max}}$ and $N_{\text{max}}$:

$$T_{\text{max}} = \frac{W_{\text{max}} - C\tau}{Ca},$$

$$N_{\text{max}} = \frac{W_{\text{max}}^2 - (C\tau)^2}{2Ca\tau}.$$ 

Further,

$$T_{\beta} = \frac{\beta\tau}{W_{\text{max}} - C\tau} - \frac{W_{\text{max}} - C\tau}{2Ca\tau}.$$ 

This completes the analysis for Phase II. We now analyze Phase I by first constructing an ODE that describes $\text{cwnd}$ growth before the BDP is reached:

$$\frac{\partial w}{\partial t} = a$$ as in Phase II, but since $S$ is below $C$, $\frac{\partial A}{\partial t} = \frac{w}{\tau}$.

The final ODE and its solution are:

$$\frac{\partial w}{\partial t} = \frac{w}{\tau}a,$$

$$w(t) = ge^{at/\tau}$$ where $g$ is a constant,

$$w(0) = bW_m = g,$$

$$w(t) = bW_m e^{at/\tau}.$$  

(10)

Next, we can solve for $T_0$ and $N_0$:

$$T_0 = \frac{\tau}{a} \ln \left( \frac{C\tau}{bW_s} \right),$$

$$N_0 = \frac{C\tau - bW_s}{a}.$$ 

where

$$W_s = \min (W_m, W_{\text{max}}).$$  

(11)

This completes the analysis for STCP.

2) CUBIC: Because CUBIC is not ACK-based, there is no need to derive two different window functions for its analysis: $w(t)$ grows as follows for both phases:

$$w(t) = c \left( t - \sqrt{\frac{bW_s}{c}} \right)^3 + W_s$$  

(12)

where $W_s$ is defined in (11), $t$ is the time since the last congestion event in unit of RTT, $c$ is a scaling factor (usually equal to 0.4), and $b$ is a multiplicative decrease factor usually equal to 0.2.

We present the closed-form solutions for the necessary variables. Since CUBIC is TSL-based, it helps to think of $T_0$ and $T_m$ as points in time that delimit the phases, rather than durations of phases (let $T_{\beta}$ remain as the duration of sub-phase II of Phase II).

$$T_m = \sqrt{\frac{bW_m}{c}},$$

$$N_m = \frac{W_m}{\tau} \sqrt{\frac{bW_m}{c}} \left( 1 - \frac{b}{4} \right),$$

$$T_0 = \sqrt{\frac{C\tau - bW_m}{c}} + \sqrt{\frac{bW_m}{c}}.$$ 

Above, $N_m$ is the amount of data transferred in the interval $[0, T_m]$. Let $N_1$ be the amount of data transferred in the interval $[T_0, T_m]$.

$$N_1 = \frac{1}{\tau} \int_{T_0}^{T_m} w(t)dt = -\frac{1}{\tau} \sqrt{\frac{C\tau - W_m}{c}} \left( \frac{C\tau + 3W_m}{4} \right),$$

$$N_1 - C(T_m - T_0) - \beta = 0.$$ 

The last equation above can be solved for $W_m$:

$$W_m = C\tau - \sqrt{\frac{4\beta\tau}{3}}.$$ 

Since we know that $W_m > C\tau$, we must take the negative root of the fourth-root term above. If $W_m$ is indeed less than or equal to $W_{\text{max}}$, then the sending rate is

$$S = \frac{N_s + kN_m}{T_s + kT_m}.$$ 

Otherwise, we use $W_{\text{max}}$ in the expressions for $T_m$ and $N_m$ above to obtain $T_{\text{max}}$ and $N_{\text{max}}$, respectively. Also,

$$T_{\beta} = \frac{\beta - N_{\text{max}} - C\sqrt{\frac{C\tau - W_{\text{max}}}{c}}}{W_{\text{max}}/\tau - C}.$$ 

Finally, $S$ is

$$S = \frac{N_s + k(N_{\text{max}} + N_{\beta})}{T_s + k(T_{\text{max}} + T_{\beta}).}$$

3) H-TCP: We do not present closed-form solutions for H-TCP, since they are too complex; the majority of the computations for this variant were performed using Matlab. Since $\Delta^{L}$ is usually set to one second, for simplicity we use it in the derivations below. We present the non-trivial case, in which the transfer lasts for $T_s + 1$ seconds. For H-TCP, let $T_0$, $T_m$, $T_{\text{max}}$, and $T_{\beta}$ be points in time, rather than phase durations. For the first second of CA, H-TCP uses the increase function shown in (1a). The $\text{cwnd}$ function is then

$$w_1(t) = bW_m + \frac{2(1-b)t}{\tau}.$$ 

The amount of data transferred during this time is $N_{\Delta^{L}}$,

$$N_{\Delta^{L}} = \frac{1}{\tau} \int_{0}^{T_{\Delta^{L}}} w_1(t)dt.$$ 

where

$$T_{\Delta^{L}} = \min \left( 1 + \frac{\tau(C\tau - bW_m)}{2(1-b)} \right).$$ 

This constraint on $T_{\Delta^{L}}$ is required because it may take less than one second for $w_1(t)$ to reach $C\tau$.

After the first second of CA, if $w_1(1) < C\tau$, H-TCP’s increase function changes as shown in (1b). The $\text{cwnd}$ function changes to

$$w_2(t) = w_1(1) + \frac{2(1-b)t}{\tau} \left( 1 + 10(t-1) + \left( \frac{t-1}{2} \right)^2 \right)$$.
until cwnd reaches $C\tau$ at time $T_0$. $N_0$, the amount of data transferred in the interval $[0, T_0]$, is:

$$N_0 = \begin{cases} N_{\Delta t}, & \text{if } T_{\Delta t} \leq 1s, \\ N_{\Delta t} + \frac{1}{\tau} \int_{T_0}^{T_{\Delta t}} w_2(t) dt, & \text{otherwise.} \end{cases}$$

$T_0$ can be isolated from the relation $w_2(T_0) = C\tau$. After $T_0$ seconds have passed, the cwnd function changes again because the transfer transitions into Phase II. This new function is described by the ODE in (5), which uses the increase function shown in (1a) or (1b) depending on the value of $T_{\Delta t}$:

$$\frac{\partial w}{\partial t} = aC\partial t,$$

$$w_3(t) = 2(1-b)C \int a(t) dt,$$

$$w_3(T_0) = C\tau \text{ is the initial condition.}$$

H-TCP then uses $w_3(t)$ for the rest of the congestion epoch, which ends at $T_m$. It is possible to solve for $T_m$ in terms of $W_m$ using the relation $w_3(T_m) = W_m$. The amount of data transferred during Phase II is $N_m$:

$$N_m = \frac{1}{\tau} \int_{T_0}^{T_m} w_3(t) dt.$$ 

It is now possible to solve for $W_m$ using the following equation:

$$N_m - C(T_m - T_0) = \beta.$$ 

If $W_{max} \geq W_m$, then $S$ is:

$$S = \frac{N_s + k(N_0 + N_m)}{T_s + kT_m}.$$ 

Otherwise, $T_\beta$ must be determined using a similar technique used for STCP and $S$ changed to:

$$S = \frac{N_s + k \left( N_0 + \frac{1}{\tau} \int_{T_0}^{T_m} w_3(t) dt + \frac{W_{max}}{\tau} (T_\beta - T_{max}) \right)}{T_s + kT_\beta}.$$ 

D. Parallel Flows

We now present a simple, yet effective modification to the single-stream modeling framework presented above to accommodate multiple flows. The main idea is to model $n$ parallel flows as a single, more aggressive flow. The modifications are as follows:

1) For any single-flow TSL-based CA cwnd function $w(t)$, the multiple-flow cwnd function is $w(nt)$, where $n$ is the number of flows. For any single-flow ACK-based CA cwnd function, the multiple-flow cwnd function is obtained by multiplying the increase factor (as in the case of STCP) or the increase function (as in the case of H-TCP) by $n$. The only exception to this rule are cwnd functions in Phase II of ACK-based CA variants, which must be left unchanged. For example, STCP’s (10) becomes

$$w(t) = bW_m e^{ant/\tau}$$

while (9) remains the same. The reason for this is that once an ACK-based variant transitions into Phase II, ACKs continue to arrive at a constant rate, so that there is no added advantage of using multiple flows past the point where cwnd $> BDP$. TSL-based variants, on the other hand, continue to grow the aggregate cwnd at a rate approximately $n$ times faster than with a single flow, for as long as congestion is not detected.

2) $W_{max}$ gets scaled up by a factor of $n$. However, there must be a hard constraint, $W_{maxH}$, on the aggregate cwnd, imposed by memory and buffer limitations. Therefore,

$$W_{max} = \min(nW_{max}, W_{maxH}).$$

3) In slow-start, we keep the duration ($T_s$) the same as for single flows, but multiply $N_s$ by $n$. In addition, instead of ssthresh, we use

$$\text{thresh} = \min(ssthresh, C\tau/n, W_{max}).$$

Interestingly, Crowcroft et al. have previously explored a related idea, but as a means of delegating some flows a higher fraction of the bandwidth on a network [25]. They propose a controller, MulTCP, that attempts to increase the throughput of a single flow by a factor of $n$ by scaling the flow’s additive increase parameter by the same amount. Simulations showed that MulTCP does indeed achieve a sending rate of approximately $nS$ as long as $n$ is not too large.

V. Model Performance

Figure 6 shows throughput predictions for one, five, and ten H-TCP streams (we also did this for two-four and six-nine streams, not shown here). Measurement data obtained from the 10 GigE testbed is also presented in the figure and serves as validation for the models. The predictions were obtained using models presented in Section IV. Figure 7 shows the result of using the multiple-stream models to predict the throughput of one, two, four, six, eight, and ten parallel STCP flows (we also did this for three, five, seven, and nine streams, not shown here). Figure 8 shows the same for CUBIC.

We use different $\beta$ and $W_{max}$ parameter values for prediction in each dataset (although $\beta$ and $W_{max}$ are kept constant within a dataset). Note the odd ‘dip’ in throughput in Figure 7a for RTTs 11.8ms and 22.6ms: this is believed to be an anomaly. Let average error be defined as follows:

$$E = \frac{100}{|\text{RTTs}|} \sum_{r \in \text{RTTs}} \left| \frac{M_r - P_r}{M_r} \right|$$

where $M_r$ and $P_r$ are the measured and predicted throughputs, respectively, for $\text{RTT} = r$. The mean errors (averaged across all RTTs and $n \in \{1, \ldots, 10\}$) for multiple-stream predictions are as follows: 5.2% for H-TCP, 9.3% for STCP, and 4.5% for CUBIC.

VI. Conclusion

In this work, we have derived a unifying scheme for analyzing single-stream and multiple-stream memory-to-memory TCP transfers. We performed a detailed analysis for a diverse set of TCP variants: STCP – a MIMD algorithm; CUBIC – a non-ACK-based algorithm; and H-TCP – an adaptive AIMD algorithm. The models that emerged from this analysis were
validated using an extensive set of measurements. The results show that our models can be used to achieve accurate and reliable throughput predictions. The measurements independently show that CUBIC and H-TCP consistently outperform STCP, and the difference in performance becomes more pronounced as the number of parallel flows grows. In future work, we plan to expand the analysis to include (a) exogenous and time-variant loss, (b) I/O constraints, and (d) considerations of fairness and convergence.

VII. ACKNOWLEDGEMENT

This work was supported in part by the US Department of Energy under Contract DE-AC02-06CH11357.

REFERENCES

Figure 8: Measurement averages and medians vs multiple-stream model predictions for CUBIC.