Theoretical Analysis of High-speed Multiple TCP Connections through Multiple Routers

Sudheer Poojary and Vinod Sharma
Department of ECE, Indian Institute of Science, Bangalore, India
Email: {sudheer,vinod}@ece.iisc.ernet.in

Abstract—We study a system of multiple routers traversed by multiple TCP connections using TCP New Reno, CUBIC and Compound. More than one router may be congested. To analyze this system we will use earlier theoretical models of TCP New Reno and CUBIC but develop a new model of TCP Compound experiencing random packet losses and queuing delays. We model the router queues as M/GI/1 queues with arrival rates controlled by TCP window flow control. We also look at an alternate approach assuming proportional fairness of TCP to find the throughputs of the multiple TCP connections. These approximations are validated through comparison with ns-2 simulations.

Index Terms—TCP Compound, TCP CUBIC, Window flow control, High speed Internet.

I. INTRODUCTION

TCP is a dominant transport layer protocol in the Internet. The goals of reliable data transfer and preventing congestion collapse on the Internet have been met successfully by TCP for more than two decades. However traditional TCP viz., TCP Reno and TCP New Reno have been found to be inadequate in wireless medium and high-speed, large delay networks. The wireless medium is prone to random packet drops and as TCP treats losses as an indication of congestion, it reduces the sending rate leading to inefficient use of network capacity. Furthermore, TCP Reno and TCP New Reno use AIMD algorithm [1],[2] for congestion control. In high-speed, large delay networks, AIMD is inefficient as it under-utilizes the links for prolonged periods of time [3],[4]. In the past few years a number of alternatives to AIMD have been proposed which address the above issues. H-TCP [5], Fast TCP [6], BIC [7], CUBIC [8], Compound [9] are a few examples of recent high-speed TCP algorithms. AIMD TCP has been extensively studied and analyzed ([10]–[14]). See [15] for an extensive survey.

High-speed TCP algorithms mentioned above use a sliding window congestion control mechanism. These algorithms increase the window size gradually till congestion is detected. Then the window size is reduced. Packet losses and increase in queuing delay are used as indicators of congestion. The algorithms can be classified as either delay-based or loss-based depending on the technique used for congestion detection.

H-TCP, BIC and CUBIC are examples of loss-based algorithms. BIC [7] uses a binary search mechanism to search for the ‘optimum’ sending rate. In H-TCP [5] and CUBIC [8] as time elapsed since last congestion increases, the sending rate increases. In [16], the authors investigate the inter-protocol fairness and performance of different high speed TCP algorithms.

Delay-based congestion control algorithms infer flow congestion from increase in queuing delay experienced by the flow. These algorithms have better RTT fairness, lower buffer occupancy and higher efficiency when compared to loss-based algorithms [6], [17]. However in a mixed environment where the different variants of TCP coexist, one sees an inequitable allocation of resources with flows using loss-based algorithms receiving higher throughput than the flows using delay-based flows. This is because of the less aggressive behaviour of delay-based algorithms.

TCP Compound ([9]) combines the loss-based approach with the delay-based approach due to which its flows are able to get their fair share of throughput in a mixed environment. In [18] and [19], the authors study performance of a single TCP Compound with random losses. Queuing delays have not been modelled.

In this paper we will consider a multi-hop network with connections using different variants of TCP, the common scenario in the current Internet. In particular we will focus on TCP New Reno, CUBIC and Compound. This is because TCP Compound is an option in Windows systems, TCP CUBIC is the default protocol in Linux based systems and TCP New Reno is the most common legacy protocol. In [20], it is shown that among the top 5000 web servers, a majority use TCP CUBIC and that TCP Compound and TCP New Reno are also commonly used.

To study the above scenario analytically, we need theoretical models of TCP New Reno, CUBIC and Compound which work with multiple TCP connections, (random) packet losses and variable round trip time (RTT) to accommodate random queuing delays at various routers on the way. TCP New Reno has been well studied. TCP CUBIC with random losses has been studied in [21]. We studied TCP CUBIC with random packet losses and queuing delays in [22]. In this paper we first form a model of TCP Compound (with multiple connections, random packet losses and queuing delays) and then use it with the previous models of TCP New Reno and TCP CUBIC to study the overall system. To the best of our knowledge, such an investigation in mixed TCP environment has not been carried out before. Also such an analysis is important and relevant as the congestion control over the Internet is heterogeneous [20].

The paper is organized as follows. In Section II, we present the system model. In section III, we briefly describe TCP
Compound protocol. In Section IV, we consider a model for TCP Compound when there is non-negligible queuing delay. In Section V, we consider an approximation of the TCP flow throughput by assuming proportionally fair allocation. In Section VI, we use the above approximations along with results for TCP CUBIC and New Reno to obtain throughput and mean window sizes when multiple TCP Compound, CUBIC and New Reno connections share a link. This models a realistic network scenario. In Section VII, we consider the case when there are multiple bottleneck queues. Section VIII concludes the paper.

II. SYSTEM MODEL

The system model studied in this paper is shown in Figure 1. We consider a network with multiple TCP connections (denoted by \( \mathcal{R} \)) going through a set of links (denoted by \( \mathcal{L} \)). The link speed of link \( l \in \mathcal{L} \) is \( C_l \) bps. TCP connection \( r \) connects TCP source \( S_r \) to TCP sink \( D_r \). We consider long-lived connections, i.e., the connections always have data to send. Packets of connection \( r \) have independent identically distributed (i.i.d.) length with a generic length denoted by \( s_r \). These packets are lost independently with probability \( p_r \geq 0 \). Such random losses are likely if the connections use wireless links. This is one likely scenario and a commonly made assumption ([18], [23]). We denote by \( \Delta_r \) the constant component of the RTT of connection \( r \), which includes the processing delays, transmission delays and link propagation delays. If flow \( r \in \mathcal{R} \) goes through link \( l \in \mathcal{L} \), we denote this by \( l \in r \). Thus each TCP flow can be associated with a set of links through which it passes. The network in this case is denoted by \( (\mathcal{R}, \mathcal{L}) \).

Different TCP flows may use TCP New Reno, CUBIC or Compound. We theoretically analyze this system. For TCP New Reno and CUBIC we use previous models. But we develop a model for a single TCP Compound connection in sections III and IV taking into account random losses and queuing delays.

III. TCP COMPOUND

The TCP Compound window has two components, an aggressive delay-based component which makes it efficient in high speed networks and a loss-based component which guarantees the flow a fair share of link capacities in a mixed environment. We denote the TCP Compound window size at the end of the \( n^{th} \) round trip time (RTT) by \( W_n \). We denote the loss-based component of \( W_n \) by \( L_n \) and the delay-based component by \( D_n \). The window size \( W_n \) is equal to \( D_n + L_n \). Since the slow start phase contributes insignificantly to the system performance under steady state for reasonable packet loss probabilities, we ignore this phase. This is common practice in TCP performance analysis ([10], [18]). The window size evolution, in congestion avoidance phase, of a TCP Compound flow is given as [9],

\[
D_{n+1} = \begin{cases} 
D_n + (\alpha(W_n)^k - 1)^+, & \text{if no loss and } Q_{n+1} < \gamma; \\
(D_n - (Q_{n+1})^+)^+, & \text{if no loss and } Q_{n+1} \geq \gamma; \\
D_n, & \text{if loss is detected};
\end{cases}
\]

\[
L_{n+1} = \begin{cases} 
L_n + 1, & \text{if no loss}; \\
L_n, & \text{if a loss is detected}.
\end{cases}
\]

The terms \( k \) and \( \zeta \) are constants and \( \gamma \) denotes the queuing threshold. The behaviour of \( L_n \) is same as that of an AIMD TCP. Hence a TCP Compound flow has rates at least as much as that of an AIMD TCP. The variable \( Q_n \) is the estimate of queuing size at the end of \( n^{th} \) RTT and is given by

\[
Q_n = \left( \frac{W_{n-1}}{RTT_{min}} - \frac{W_{n-1}}{RTT_n} \right) RTT_{min},
\]

where \( RTT_n \) is the measured RTT at the end of \( n^{th} \) round and \( RTT_{min} \) is the lowest RTT observed till \( n^{th} \) round.

In [24] the authors modify the TCP Compound algorithm incorporating a variable queuing threshold in place of a fixed \( \gamma \). We do not consider this modification.

IV. TCP COMPOUND WINDOW DYNAMICS AND THROUGHPUT WITH VARIABLE RTT

When queuing is negligible, the TCP window dynamics is approximately given by equations (1) and (2) with \( Q_{n+1} = 0 \). In this case, we see that \((D_n, L_n)\) form a finite state Markov chain which is aperiodic and irreducible and hence has a unique stationary distribution. We then obtain the average window size and throughput using the stationary distribution of \((D_n, L_n)\).

In this section, we develop a model for a single TCP Compound flow when the queuing delays are non-negligible and obtain its steady state throughput and average window size. We denote the time period to send a window of packets and to receive all the corresponding ACKs from the receiver as a round. The time period of a round is approximately the RTT of connection \( r \). The terms \( k \) and \( \zeta \) are constants and \( \gamma \) denotes the queuing threshold. The behaviour of \( L_n \) is same as that of an AIMD TCP. Hence a TCP Compound flow has rates at least as much as that of an AIMD TCP. The variable \( Q_n \) is the estimate of queuing size at the end of \( n^{th} \) RTT and is given by

\[
Q_n = \left( \frac{W_{n-1}}{RTT_{min}} - \frac{W_{n-1}}{RTT_n} \right) RTT_{min},
\]
round trip time by $RTT_n$. The quantities $R_n$ and $RTT_n$ can be approximated by

$$R_n = \Delta + \frac{W_{n-1}}{C} + \frac{Q_{n-1}}{C},$$

$$RTT_n = \Delta + \frac{Q_{n-1}}{C} + \frac{1}{C},$$

where $C$ is the bottleneck link capacity (in packets/sec) and $\Delta$ denotes the propagation delay. We see from equations (1) and (2) that $W_{n+1}$ is a function of $L_n$, $D_n$ and $Q_n$. The queue size, $Q_n$, at the end of the $n^{th}$ round can not be computed exactly. Hence we approximate $Q_n$ by $\max(0, W_{n-1} - C\Delta)$. The window size component $D_n$ takes values in $\{0, 1, \ldots, W_{max}\}$ and $L_n$ in $\{1, 2, \ldots, W_{max}\}$ and $L_n + D_n \leq W_{max}$. Since $W_n$ is finite and $C\Delta$ is a constant, $Q_n$ can only take a finite number of values in $[0, W_{max} - C\Delta]$. Since we also have an i.i.d. Bernoulli random loss model with packet drop probability $p > 0$, we obtain a Markov chain, $(L_n, D_n, Q_n)$, with a finite number of states. As $p > 0$, state $(1, 0, 0)$ can be reached from any state and hence the Markov chain $\{ (L_n, D_n, Q_n) \}$ is irreducible. We denote the unique steady state distribution of the Markov chain by $\pi(l, d, q)$.

The steady state mean window size $EW$ at the end of a round is

$$EW = \sum_{l,d,q:1 \leq l+d \leq W_{max}} (l+d)\pi(l,d,q).$$

Let the steady state mean RTT at the end of a round be $E[RTT]$. Let $E[W]$ and $E[RTT]$ be the time average mean window size and mean RTT respectively. Using Palm calculus [25],

$$E[W] = \frac{E[\bar{R}]}{E[R]}, \quad E[RTT] = \frac{E[\bar{W}RTT]}{E[W]}.$$

The steady state throughput for the connection is then given by $\lambda = (1 - p)E[W]/E[RTT]$ packets/sec. The average window size, $E[W]$ is dependent on the average queue size, $E[Q]$ and $p$. This is unlike TCP Reno where $E[W]$ is only dependent on $p$ and TCP CUBIC where $E[W]$ is a function of $E[RTT]$ and $p$ even when there is no queuing delay [22].

We denote the relation between $E[W]$ and $E[Q]$ and $p$ for TCP Compound obtained above by

$$E[W] = fp(E[Q]).$$

This expression is not available in closed form. As against this, there is a closed form expression for $E[W]$ in [9], but that is valid for only constant RTT. For variable RTT, considered in this and later sections, equation (8) provides a better approximation.

We compare the above results for average window size and mean throughput with those obtained by ns-2 simulations in Figure 2. Link speed of 1 Mbps is taken to have significant queuing delay. The propagation delay is set to 100 m sec. The packet size is 1050 bytes. Theoretical results match well with simulations.

![Figure 2: Average window size (EW) and throughput (packets/sec) for single TCP Compound connection vs packet error rate, C = 1 Mbps and $\Delta$ = 100 ms.](image-url)

We use these results, in particular function $fp$ in (8) in Sections VI and VII to obtain results for multiple TCP connections.

V. OPTIMIZATION APPROACH TO TCP PERFORMANCE ANALYSIS

In this section we present an optimization approach to model multiple TCP connections through multiple routers, initially developed in [26]. In [26], it was proposed for constant window size protocols. We extend it to our setup where different TCP protocols with variable window size may be employed. In Sections VI and VII we will use it to analyze our overall system. We will also consider an M/GI/1 queue based approximation which was developed in [22] and [27].

Consider the network $(R, L)$ as described in Section II. For flow $r \in R$, let $\lambda_r$ be the average output rate, $E[W_r]$ be the average window size, $\Delta_r$ be the constant delay (propagation, processing and transmission delays), $p_r$ be the probability of loss, $E[A_r]$ be the sending rate and $E[RTT_r]$ be the average RTT.

For TCP flow $r$ we have,

$$E[A_r] = \frac{E[W_r]}{E[RTT_r]}.$$ (9)

The average output rate of flow $r$ is given by $\lambda_r \approx E[A_r](1 - p_r)$. Let $E[Q_l]$ denote the average queue size at link $l$. The waiting time for flow $r$ through link $l$ is given by $E[Q_l]$. The average RTT for flow $r$ is

$$E[RTT_r] = \Delta_r + \left( \sum_{l \in r} \frac{E[Q_l]}{\sum_{r' \in R : l \in r'} \lambda_r} \right).$$ (10)

Therefore,

$$E[W_r] = \lambda_r \left( \Delta_r + \left( \sum_{l \in r} \frac{E[Q_l]}{\sum_{r' \in R : l \in r'} \lambda_r} \right) \right).$$ (11)

Next we assume that $\sum_{r \in R : l \in r} \lambda_r < C_l$ implies that $E[Q_l] = 0$. This assumption although inaccurate is motivated
by the fluid flow model in [26]. Then we have
\[
(1 - p_r)E[W_r] = \lambda_r \Delta_r + \left( \sum_{l \in r} \frac{E[Q_l]}{C_l} \right) \lambda_r.
\]
(12)

Now consider the following optimization problem,
\[
\max_{r \in R} \left( B_r \log(\lambda_r) - \lambda_r \bar{T}_r \right)
\]
(13)
s.t.
\[
\lambda_r \geq 0, \forall r \in R \text{ and } \sum_{r \in L} \lambda_r = C_l, \forall l \in L,
\]

where \(B_r\) are weights associated with flow \(r\).

The above optimization problem has concave objective function and affine constraints. Therefore we have a unique solution. Let \(\lambda^*\) be the optimal solution and let \(\{\mu_l\}_{l \in L}\) be the Lagrangian multipliers for the inequality constraints. Then by the KKT conditions for optimality, we have
\[
B_r = \lambda^*_r \Delta_r + \left( \sum_{l \in r} \mu_l \lambda^*_l \right)
\]
and either \(\mu_l = 0\) or \(\sum_{r \in r \subseteq l} \lambda^*_r = C_l\).

We see that if we replace \(B_r\) and \(\mu_l\) in equation (14) by \((1 - p_r)E[W_r]\) and \(\frac{E[Q_l]}{C_l}\) in equation (12), respectively then the equations are identical. This suggests that we can obtain (12) as a solution to the optimization problem (13) after making the above mentioned variable changes. It can be shown that the optimal allocation provides proportional fairness [26], with weights proportional to \(E[W_r]\).

We use this approach to estimate the throughput and average window size of multiple TCP connections for the system in Figure 1.

VI. MULTIPLE TCP CONNECTIONS

In this section we consider the case where multiple TCP connections share a single bottleneck link. The connections may be using TCP Compound, TCP CUBIC or New Reno.

We have developed an analytical model for TCP CUBIC behaviour in [22] which is used here for evaluating the TCP CUBIC throughput and window size. As in [22], we use the single connection model as a building block. In [22], we have modeled the router queue as an M/GI/1 queue with arrival rates obtained from the window dynamics. We have found it to be a good approximation. This motivates us to use the same approximations for TCP Compound connections. Alternately, we will also use the optimization framework of Section V.

First we consider the case where the link is shared by multiple TCP Compound flows. There are \(N\) TCP flows sharing a bottleneck link of capacity \(C\) bps. Suppose \(E[s_i]\) and \(E[s^2_i]\) are the mean and second moments of packet lengths of connection \(i\). If \(\lambda_i\) is the throughput of TCP connection \(i\), then
\[
E[s] = \sum_{i} \frac{\lambda_i}{\lambda} E[s_i], \quad E[s^2] = \sum_{i} \frac{\lambda_i}{\lambda} E[s^2_i],
\]
(15)

where \(E[s]\) and \(E[s^2]\) are the overall mean packet length and its second moment and \(\lambda\) packets/sec is the overall throughput.

TABLE I: Average window sizes and throughputs for three C-TCP connections with bottleneck link capacity = 10 Mbps

| Flow (ns-2) | \(p_i\) | \(\Delta_i\) | \(E[|W_i|]\) (M/GI/1) | \(E[|W_i|]\) (PP) | \(\lambda_i\) (M/GI/1) | \(\lambda_i\) (PP) |
|------------|--------|--------------|---------------------|-----------------|----------------|----------------|
| 1          | 0.005  | 0.1          | 19.0               | 19.5            | 186.8          | 191.6          | 192.2 |
| 2          | 0.005  | 0.1          | 19.0               | 19.5            | 186.4          | 191.6          | 192.2 |
| 3          | 0.005  | 0.1          | 19.5               | 19.5            | 189.0          | 191.6          | 192.2 |
| 4          | 0.005  | 0.1          | 19.2               | 19.5            | 186.9          | 191.4          | 192.2 |
| 5          | 0.005  | 0.5          | 19.2               | 19.5            | 38.0           | 38.7           | 38.8 |
| 6          | 0.005  | 0.4          | 19.1               | 19.5            | 451.5          | 467.5          | 474.5 |
| 7          | 0.005  | 0.1          | 19.0               | 19.5            | 183.0          | 190.1          | 192.2 |
| 8          | 0.010  | 0.1          | 13.0               | 12.9            | 126.8          | 126.1          | 126.5 |
| 9          | 0.003  | 0.1          | 25.8               | 27.6            | 253.6          | 271.2          | 272.5 |
| 10         | 0.003  | 0.04         | 25.9               | 27.6            | 590.6          | 653.9          | 672.6 |

Assuming the bottleneck queue to be an M/GI/1, the mean queue size is given by
\[
E[Q] = \frac{\lambda^2 E[s^2]}{2C^2 (1 - \rho)}.
\]
(16)

Then, the mean RTT of connection \(i\) is given by
\[
E[RTT_i] = \frac{\lambda E[s^2]}{2C^2 (1 - \rho)} + \frac{E[s]}{C} + \Delta_i,
\]
(17)

where \(\rho = \lambda E[s]/C\).

From (8), the average window size for flow \(i\) is
\[
E[|W_i|] = p_i (E[Q]).
\]
(18)

For connection \(i\), using Little’s law,
\[
\lambda_i = \frac{E[|W_i|]}{E[RTT_i]},
\]
(19)

We can now solve equations (17), (18) and (19) to compute \(E[|W_i|]\), \(E[|W_i|]\) and \(\lambda_i\) for \(i = 1, 2, \ldots, n\). For the optimization based approach discussed in Section V, we solve the optimization problem (13) replacing \(B_r\) by \((1 - p_r)E[W_r]\) along with (18) and (19).

In the ‘variable RTT’ model developed for Compound TCP in Section IV, we approximate the queue size \(Q\) by \(\max(0, W_{n-1} - C - D)\). This approximation overestimates the queue size when the TCP throughput is far lesser than the link capacity \(C\). In such a case it seems better to ignore queuing and use the TCP Compound model with constant RTT (discussed in the beginning of Section IV).

Table I compares \(E[|W_i|]\) and \(\lambda_i\) obtained for three TCP Compound connections (denoted as C-TCP) via the M/GI/1 approximation (denoted by M/GI/1) and via the weighted proportional fair allocation approximation discussed in Section V (denoted by PP) with the results via ns-2 simulations. The link capacity is 10 Mbps. We observe that most of the errors are within 10%. For both cases if \(E[Q] \leq 5\) we use the constant RTT model of Compound. We will be using the same approximation henceforth.

We now consider the case where the bottleneck link is shared by TCP Compound, CUBIC and New Reno connections. This is the likely scenario in the practical network. For a connection using TCP CUBIC (say connection \(j\)), the average window size is a function of \(p_j\) and \(E[RTT_j]\), (as against TCP Compound (18)). We denote this relation by
TABLE II: Average Window Size for Compound, CUBIC and New Reno connections with packet error rate, $p_i = 0.005$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>19.1 19.3 18.7</td>
<td>19.5 17.7 18.5</td>
<td>19.5 17.6 18.5</td>
</tr>
<tr>
<td>0.1</td>
<td>19.1 19.3 18.2</td>
<td>19.5 17.7 18.5</td>
<td>19.5 17.6 18.5</td>
</tr>
<tr>
<td>0.2</td>
<td>19.1 19.4 18.5</td>
<td>19.5 24.0 18.5</td>
<td>19.5 24.0 18.5</td>
</tr>
<tr>
<td>0.4</td>
<td>18.9 19.2 18.4</td>
<td>19.5 30.1 18.5</td>
<td>19.5 30.3 18.5</td>
</tr>
<tr>
<td>0.5</td>
<td>18.8 17.7 18.4</td>
<td>19.5 46.3 18.5</td>
<td>19.5 46.3 18.5</td>
</tr>
<tr>
<td>1</td>
<td>19.1 18.6 17.7</td>
<td>19.5 73.6 18.5</td>
<td>19.5 73.6 18.5</td>
</tr>
</tbody>
</table>

$C = 10$ Mbps

TABLE III: Throughput for Compound, CUBIC and New Reno connections with packet error rate, $p_i = 0.005$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>19.1 18.8 18.3</td>
<td>19.5 18.9 18.5</td>
<td>19.5 18.9 18.5</td>
</tr>
<tr>
<td>0.1</td>
<td>19.2 19.0 18.5</td>
<td>19.5 17.6 18.5</td>
<td>19.5 17.6 18.5</td>
</tr>
<tr>
<td>0.2</td>
<td>19.0 24.5 18.4</td>
<td>19.5 23.9 18.5</td>
<td>19.5 23.9 18.5</td>
</tr>
<tr>
<td>0.4</td>
<td>18.7 40.2 18.5</td>
<td>19.5 39.3 18.5</td>
<td>19.5 39.3 18.5</td>
</tr>
<tr>
<td>0.5</td>
<td>18.6 46.5 18.5</td>
<td>19.5 46.2 18.5</td>
<td>19.5 46.2 18.5</td>
</tr>
<tr>
<td>1</td>
<td>18.6 76.7 18.3</td>
<td>19.5 74.5 18.5</td>
<td>19.5 74.5 18.5</td>
</tr>
</tbody>
</table>

$C = 1$ Gbps

$E[W_j] = g_{ij} \cdot [E[RRTT_j]]$, \hspace{1cm} (20)

which is provided in [22]. Thus $E[RRTT_j]$, $\lambda_j$ and $E[W_j]$ can be computed using equations (17), (19) and (20). For a connection using TCP New Reno (say connection $k$), the average window size is a function of $p_k$ and is independent of RTT. We use [10] for computing the average window size of TCP New Reno connections.

$E[W_k] = 1.31/E[RRTT_k]$. \hspace{1cm} (21)

Tables II and III show $E[W_j]$ and $\lambda_i$ for three TCP connections using Compound (denoted as C-TCP), CUBIC and New Reno respectively. We compare these results with ns-2 simulations. For all the simulations $p_i$ and $\Delta_i$ are kept same for all $i$. Theory and simulations are within 10%. We observe from Tables II and III that when $\Delta \geq 0.2$ sec, TCP CUBIC provides higher throughput than TCP Compound and TCP New Reno.

We have observed that the average window size of TCP Compound is a function of $p$ and the average queue size but is not dependent on $\Delta$ whereas the TCP CUBIC average window size grows with RTT. This explains the difference in throughput.

VII. THROUGHPUT AND AVERAGE WINDOW SIZES FOR MULTIPLE QUEUES

In this section we consider the case when TCP flows go through multiple bottleneck links. They may be using TCP Compound, CUBIC or New Reno. We will use the M/GI/1 based approximation as well as the optimization based approach.

As in section V, we denote the set of flows by $R$ and set of links by $L$. The average RTT for flow $r \in R$ is given by equation (10). We use (16) to compute the average queue size, $E[q_r]$, at link $l$. We replace $p$ by $p_i = (\sum_{r \in R} \lambda_r)E[s^i]/C_l$, where $E[s^i] = \sum_{r' \in r'} \lambda_{r'}$ and $C_l$ is the capacity of link $l$ (in bps).

For a TCP Compound connection $i$, we now solve equations (18), (19) and (10) to compute $E[W_i]$ and $E[RRTT_i]$. For a TCP CUBIC connection (say $j$) we use equations (19), (20) and (10) to compute $\lambda_j$, $E[W_j]$ and $E[RRTT_j]$. For a TCP New Reno connection (say $k$) we use equations (19), (21) and (10) to compute $\lambda_k$, $E[W_k]$ and $E[RRTT_k]$. For the optimization based approach discussed in Section V, instead of (10) we use (13) along with (18), (19), (20) and (21) to get throughput and average window sizes of the different TCP connections.

To verify the above approximate model, we set up ns-2 simulations for a network with twelve TCP flows (four Compound, four New Reno, four CUBIC) and three queues as shown in Figure 3. Each group $F_i$ consists of one Compound, one CUBIC and one New Reno connection and have the same propagation delay $\Delta_i$ and experience the same PER $p_i$.

In Figure 3, the queues $Q_1$, $Q_2$ and $Q_3$ have capacity $C_1 = 10$ Mbps, $C_2 = 1$ Gbps and $C_3 = 10$ Mbps respectively. The link speed 10 Mbs are kept to ensure queuing delays. Without queuing delays the analytical models can be simpler and can match simulations better. However one does come across lower bandwidths at least at access links. The flows labelled $F_1$ go through all three queues, flows labelled $F_2$ go through only $Q_1$, flows labelled $F_3$ go through only $Q_2$ and flows labelled $F_4$ go through only $Q_3$. The packet error rate for all connections is 0.005. The propagation delays are given by $\Delta_1 = 500$ msec, $\Delta_2 = 150$ msec, $\Delta_3 = 300$ msec, $\Delta_4 = 100$ msec for the four classes of flows. Results for this setup are provided in Table IV.

In Table V, we provide results for the topology of Figure 3 with the following modifications: $C_3 = 5$ Mbps, flows $F_2$ go through $Q_1$ and $Q_2$ and flows $F_3$ go through $Q_2$ and $Q_3$. The packet error rates for flows $F_2$ and $F_4$ are 0.003. The propagation delays for $F_2$ and $F_3$ are 300msec. We call this case 2.

We compare results obtained using the M/GI/1 approximation (denoted by M/GI/1) and by the weighted proportional fair allocation approximation (denoted by PF) with the ns-2 simulations in Tables IV and V. We see that TCP CUBIC gets substantially higher throughput than others when $\Delta \geq 300$ msec. Furthermore, in Table V, we see that in $F_4$ with $\Delta = 100$ msec, TCP CUBIC gets the highest throughput. This
is due to the increase in RTT caused by substantial queuing at $Q_3$ in case 2. These conform to the idea that TCP CUBIC being only loss based is more aggressive than TCP Compound. We see that the theory and simulations match reasonably well.

### VIII. Summary

We have studied a system where multiple TCP connections pass through multiple (bottleneck) routers. The TCP connections may use TCP New Reno, CUBIC or Compound. To study this system we have first developed an analytical model for a single TCP Compound connection with non-negligible queuing and random packet loss. We obtained average window size and throughput. These results are used in a multiple TCP connection scenario which includes TCP CUBIC and TCP New Reno connections (for which we have used existing models). We have then analyzed the case of multiple bottleneck links. We use two approximate models for analysis: M/GI/1 based and optimization based. We have compared the results of our analysis with ns-2 simulations and seen that there is a close match between our analysis and simulations for both models. The comparison of the three protocols indicates when propagation delays are large, TCP CUBIC gets higher throughput than TCP Compound and TCP New Reno.

### TABLE V: Average window size and throughput for case 2

<table>
<thead>
<tr>
<th>Flow group $F_1$</th>
<th>TCP type</th>
<th>$E[\psi]$ (ns-2)</th>
<th>$E[\psi]$ (M/GI/1)</th>
<th>$E[\psi]$ (DFP)</th>
<th>$\lambda$ (ns-2)</th>
<th>$\lambda$ (M/GI/1)</th>
<th>$\lambda$ (DFP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$</td>
<td>CUBIC</td>
<td>10.5</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>New Reno</td>
<td>18.3</td>
<td>18.5</td>
<td>18.5</td>
<td>25.0</td>
<td>24.6</td>
<td>24.6</td>
</tr>
<tr>
<td></td>
<td>Compound</td>
<td>2.7</td>
<td>2.6</td>
<td>2.6</td>
<td>15.0</td>
<td>15.3</td>
<td>15.3</td>
</tr>
<tr>
<td>$F_2$</td>
<td>CUBIC</td>
<td>23.7</td>
<td>23.9</td>
<td>23.9</td>
<td>78.4</td>
<td>79.2</td>
<td>79.3</td>
</tr>
<tr>
<td></td>
<td>New Reno</td>
<td>23.7</td>
<td>23.9</td>
<td>23.9</td>
<td>78.4</td>
<td>79.2</td>
<td>79.3</td>
</tr>
<tr>
<td></td>
<td>Compound</td>
<td>23.7</td>
<td>23.9</td>
<td>23.9</td>
<td>78.4</td>
<td>79.2</td>
<td>79.3</td>
</tr>
<tr>
<td>$F_3$</td>
<td>CUBIC</td>
<td>23.7</td>
<td>23.9</td>
<td>23.9</td>
<td>78.4</td>
<td>79.2</td>
<td>79.3</td>
</tr>
<tr>
<td></td>
<td>New Reno</td>
<td>23.7</td>
<td>23.9</td>
<td>23.9</td>
<td>78.4</td>
<td>79.2</td>
<td>79.3</td>
</tr>
<tr>
<td></td>
<td>Compound</td>
<td>23.7</td>
<td>23.9</td>
<td>23.9</td>
<td>78.4</td>
<td>79.2</td>
<td>79.3</td>
</tr>
</tbody>
</table>

In this paper, we assume a wireless scenario where packets are dropped randomly. One future direction of work is to compare performance in the droptrail scenario.

### REFERENCES