periodica polytechnica

*Electrical Engineering*
*51/1-2 (2007) 57–64*
*doi: 10.3311/pp.ee.2007-1-2.06*
*web: http://www.pp.bme.hu/ee*


RESEARCH ARTICLE

# Throughput analysis of Scalable TCP congestion control

*Mohamed* Tekala / *Róbert* Szabó

Received 2006-05-19

***Abstract***

*Scalable TCP (STCP) has been proposed a congestion control algorithm for high speed networks. We present a mathematical analysis of STCP's congestion window through the slow start and the congestion avoidance phases. We analyse the evolution of congestion windows for single and multiple flows and for DropTail queues with and without random loss. We derive throughput formulas for the different setups and reveal the inherent unfairness between different round trip times flows. Our mathematical analysis is compared to state-of-the-art network simulator (ns) results, which verifies our model's accuracy. With our analysis we want to adaptively control STCP's fixed parameters in order to overcome the fairness problems. These experiments are work in progress and will be presented in a sequel paper.*

***Keywords***

*Scalable TCP analysis · throughput · TCP fairness*

**Mohamed Tekala**

Department of Telecommunications and Media Informatics, BME, H-1117 Budapest Magyar Tudósok krt. 2., Hungary
e-mail: tekala@qosip.tmit.bme.hu

**Róbert Szabó**

Department of Telecommunications and Media Informatics, BME, H-1117 Budapest Magyar Tudósok krt. 2., Hungary
e-mail: szabo@tmit.bme.hu

## 1 Introduction

Transmission Control Protocol (TCP) is a reliable and widely used internet transport protocol. Well known examples of its application include the Hyper Text Transfer Protocol (HTTP) and the File Transfer Protocol (FTP), both of which are based on TCP. Communication networks often experience periods where their offered traffic exceeds their available transmission capacity; during such periods, the network is said to be congested. Congestion control [6] algorithms have been introduced into TCP to relieve the problem of network collapse during these periods. Such algorithms attempt to share network resources between flows during periods of congestion, generally leading to similar throughputs for flows with similar round trip times.

The basic TCP flow control algorithm uses a sliding window and end-to-end acknowledgments to provide reliable data transfer across a network. TCP congestion management is generally implemented in two phases: $i$) slow-start and $ii$) congestion avoidance. This breakdown allows TCP to increase the overall data transmission rate rapidly without overwhelming the network. In particular, TCP uses the congestion window variable ($W$) to describe the maximum number of unacknowledged bytes present in the network. The receiver also advertises to the sender a receive window ($rwnd$), which is the size of the socket receive buffer available for this connection. The receive window allows the receiving application to exert some control over the flow.

The lesser of $W$ and $rwnd$ determines the maximum number of packets that can be present in the network waiting for acknowledgments. The congestion window $W$ provides flow control during periods in which the network is congested. Packet loss is detected either through the time-out of an unacknowledged packet, the receipt of several duplicate acknowledgments, or through selective acknowledgment (SACK) reports [12] sent by the receiver. Both packet loss and explicit congestion notification (ECN) are taken as evidence of congestion [13]. A TCP sender updates its congestion window whenever it receives an acknowledgment of received packets, as well as on detection of congestion. The performance of a TCP connection is dependent on the network bandwidth, the round trip time and the packet loss.

There has been much research on improving the performance of TCP in situations where the product of network bandwidth and delay is high (see related works in [5, 6, 8–10]).

Scalable TCP (STCP) [10] is one result of this work. STCP is a simple sender-side extension to TCP; specially designed for use in high-speed, wide-area links. It updates the congestion window using fixed increase/decrease parameters. Unfortunately, under this protocol heterogeneous flows (i.e., flows with different round trip times) sharing the same bottleneck link will not receive equal portions of the available bandwidth [14].

The goal of this work is to obtain a detailed understanding of the performance of Scalable TCP over asymmetric networks. We conducted a mathematical analysis of the behaviour of Scalable TCP for a single flow and multiple flows competing on bottleneck link, where both synchronous as well as asynchronous losses are considered. Our analysis of Scalable TCP shows that the throughput of connections is inversely proportional to its round trip time (RTT). This behaviour is the source of the Scalable TCP unfairness problem. Finally, a set of network simulations is implemented to verify our expressions.

The rest of the paper is organized as follows: Sec. 2 summarizes the related works, and an introduction to STCP is provided in Sec. 2.1. In Sec. 3 we present analytical expressions for the performance of Scalable TCP in different environments. Simulation results of Sec. 4 verify our analytical analysis. Finally, Sec. 5 concludes our work.

## 2 Related Works

Several TCP variants have recently been proposed. Analytical studies of these protocols can be used to improve existing congestion control algorithms and TCP implementations, as well as to evaluate new techniques. One analytical study and simulation of TCP (Tahoe and Reno) with no reverse path congestion has been presented by Lakshman and Madhow [11]. These authors studied the RTT unfairness of standard TCP in high bandwidth-delay product networks, and reported that under a DropTail queue system throughput is inversely proportional to RTT. In [4], Firoiu and Borden also investigate the interaction between TCP and active queue management (AQM) algorithms.

Scalable TCP [10], which is specifically designed for use in networks with a high bandwidth-delay product, updates the congestion window using fixed parameters.

Altman et al. have studied the window behaviour of Scalable TCP in the presence of Markovian window-independent losses [1]. Another important research direction related to congestion control protocol is the fairness issue, which arises when multiple sources must share a common link. Chiu and Jain, for example, have shown that multiplicative increase - multiplicative decrease (MIMD) algorithms (e.g., Scalable TCP) are unfair in the presence of synchronous losses [3]. On the other hand, Altman et al. have also studied MIMD congestion control algorithms and have shown that fairness can be achieved by introducing asynchronous losses [2]. Furthermore, they have also investigated the difference in achievable throughput between additive increase-multiplicative decrease (AIMD) sessions (e.g., NewReno) and MIMD sessions in the presence of synchronous losses.

In a previous work [14], we studied the performance of Scalable TCP under different network environments and its fairness in allocating bandwidth to connections with different RTTs. The studies mentioned above have motivated us to model the behaviour of Scalable TCP mathematically to understand the reasons behind the weakness of Scalable TCP.

### 2.1 Scalable TCP

Scalable TCP (STCP) involves a simple sender-side alteration to the standard TCP congestion window update algorithm. It robustly improves performance in high-speed, wide-area networks using traditional TCP receivers. Scalable TCP updates its congestion window using fixed increase and decrease parameters.

The Scalable TCP window update algorithm, as defined in [10], is divided into two phases.

**Slow-start phase:** in which the congestion window is increased by one packet for each acknowledgment received:

$$W = W + 1 \qquad \text{Ack}; \tag{1}$$

**Congestion avoidance phase:** If congestion has not been detected in at least one round trip time, the window responds to each acknowledgment received with the update

$$W = W + \alpha, \tag{2}$$

where $\alpha \in (0, 1)$ is a constant parameter. In the event of congestion, the congestion window is multiplicatively decreased as follows:

$$W = \beta \cdot W, \tag{3}$$

where $\beta \in (0, 1)$ is also constant. Typical values of these parameters are $\alpha = 0.01$ and $\beta = 0.875$. Further details on the Scalable TCP algorithm are available in [10].

## 3 Analytical Analysis

Our goal in analysing Scalable TCP is to later derive the functions $\alpha(\cdot)$ and $\beta(\cdot)$ that optimize throughput and fairness, in contrast to the approach of heuristically setting $\alpha$ and $\beta$ to constant values. In order to accomplish this, we first introduce our system model, and then investigate single and multiple Scalable TCP connection(s) with and without random losses.

### 3.1 System model

In our analysis we follow the system model presented by Lakshman and Madhow [11], but apply it to the multiplicative increase - multiplicative decrease (MIMD) type of Scalable TCP just described. We briefly list the assumptions of the system model:

- Data sources are infinite; there are always sufficient data to send packets of the maximum segment size (MSS);

- There is a single bottleneck, with a capacity of $\mu$ packets per second and buffer size $B$ (in packets). The bottleneck uses FIFO queuing (DropTail) unless otherwise specified;
- Random loss is introduced in addition to buffer overflow;
- The round trip time (RTT) is defined as the sum of two parts. First is the fixed (propagation) delay $T$, which includes all fixed delays of processing, transmission, and propagation. A variable delay denoted $D(t)$ is also introduced to represent the random queuing delay (service time). The total RTT is thus $T + D(t)$, or $T_i + D(t)$ for the $i^{th}$ connection when the distinction is necessary;
- Each connection uses a window-based flow control protocol. $W_i(t)$ denotes the window size of the $i^{\text{th}}$ connection at time $t$, i.e., the maximum number of unacknowledged packets. The maximum window size is $W_i^{max} = \max_t W_i(t)$. This gives the result $W_i^{max} = \mu T_i + B_i$ when the bottleneck buffer is fully occupied and there are $\mu T_i$ packets in flight and $B_i$ packets in the buffer.

Scalable TCP, just like most other TCPs, evolves in two distinct phases: i) a "slow start" phase, which begins at $W = 1$ and continues either until $W$ exceeds an initially chosen value of the threshold $ssthresh$ or until packet loss occurs; and ii) a congestion avoidance phase where the window increases more slowly until packet loss occurs. If packets are lost, then a recovery method is invoked where the congestion window is reduced to $\beta W^{\max}$ and the threshold is reset to $ssthresh = W^{\max}/2$. We analyse the throughput and fairness of such Scalable TCP connections both with and without random packet loss.

### 3.2 STCP without random loss

In this scenario the only source of loss is buffer overflow, which happens when the bottleneck link has been fully utilized. The scenario analysed incorporates both of the phases described above (see Fig. 1).

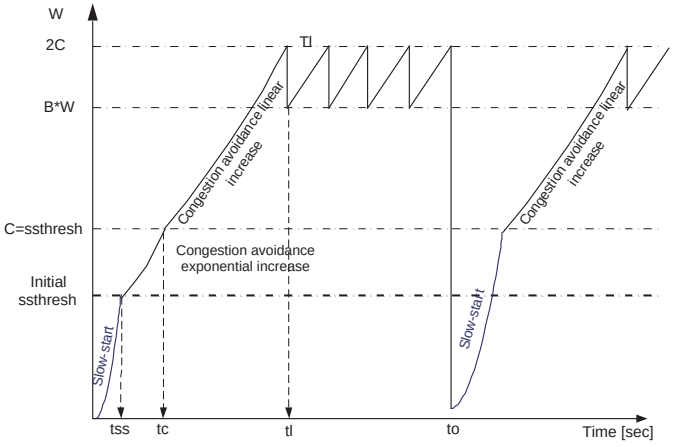


**Fig. 1.** The growth of $W$ as a function of time

#### 3.2.1 Slow-start phase

In the traditional TCP slow-start phase, the congestion window ($W$) is increased by one maximum size segment (MSS) for every received acknowledgment. This method basically doubles $W$ every RTT cycle ($T$), resulting in very rapid increases for large bandwidth-delay product (BDP) networks. On the one hand this approach is favourable, as it quickly utilizes the available bandwidth. If not stopped before overflow, however, then thousands of packets can be dropped in a single RTT – resulting in severe congestion, serious retransmissions, and timeouts. In a recent work by Sally Floyd, a limited slow-start (LSS) phase was proposed that places an upper bound on the window growth [5]. Let $dW/dt$ denote the rate of window growth with respect to time, $dW/da$ the window growth per arriving acknowledgment, and $da/dt$ the rate at which acknowledgments are arriving. A multiplicative increase phase, where $W = W + \alpha$ for each acknowledgment, can be described by the rates

$$dW/da = \alpha \tag{4}$$

and

$$\frac{da}{dt} = \begin{cases} W/T & \text{when } W \leq C \\ \mu & \text{when } W > C \end{cases}, \tag{5}$$

where $C = \mu T$ is the pipe size (capacity) of the connection.

In slow-start phase, for reasonable buffers sizes $ssthresh$ is less than $C$. In general $B = \gamma \mu T$, where $\gamma = B/(\mu T)$ is the normalized buffer size. If $\gamma = 1$ then the buffer is set to the BDP of the network. In large BDP networks, we have $\gamma \leq 1$ in most cases. When $\gamma \leq 1$, setting $ssthresh$ to half the congestion window ensures that $da/dt = W/T$. Combining Eq. (4) with Eq. (5) and integrating the result from $t = 0$ to $t$ yields

$$W(t) = \exp(\alpha \frac{t}{T}), \tag{6}$$

where $\alpha = 1$ (for traditional TCP slow-start phase). If we denote the initially chosen value of $ssthresh$ by $W_{ss}$, then we can calculate the time $t_{ss}$ required for the congestion window to attain this level:

$$t_{ss} = T \ln W_{ss}. \tag{7}$$

To calculate the number of sent packets $n(t)$, one must integrate the instantaneous rate – which is best approximated by the rate of acknowledgments given in Eq. (5). In our case $\alpha = 1$, so where $W \leq C$ this yields

$$n_{ss} = n(t_{ss}) = W(t_{ss}), \tag{8}$$

and the TCP throughput is thus $n_{ss}/t_{ss}$.

#### 3.2.2 Congestion avoidance phase

In Scalable TCP the congestion avoidance phase is very similar to the slow-start phase; in Kelly's work [10], however, the $\alpha$ parameter is fixed to 0.01. For the purposes of our analysis, this phase is further divided into an non-congested phase ($W \leq C$) and an accumulating backlog phase ($W > C$).

**Non-congested phase ($W \leq C$):**

For the non-congested phase $RTT \equiv T$. Following Eqs. (4) and (5), defined in the slow-start phase, the window function evolves as follows:

$$W(t) = W(t_{ss})e^{\alpha \frac{t-t_{ss}}{T}} \quad \forall t : t_C > t \geq t_{ss}, \tag{9}$$

where $t_c$ is defined as $W(t_c) = C$ (see Fig. 1).

The length of this period is therefore

$$\tau_c = t_c - t_{ss} = \frac{T}{\alpha} \ln\left(\frac{C}{ssthresh}\right). \tag{10}$$

**Backlog accumulation phase ($W > C$):**

When $W(t) > C$, Eq. (5) uses $\mu$ as the rate of acknowledgements. We thus have $\lambda \equiv \mu$, where $\lambda$ is the achieved throughput. This will lead to a linear increase of the congestion window until packet loss occurs (Fig. 1), after which

$$W(t) = C + \alpha\mu t, \quad \forall t : t_l > t \geq t_c. \tag{11}$$

We define $t_l$ as the value of the congestion window when the first loss happens; i.e., $W^{\max} = W(t_l)$. The backlog accumulation time, from the start of this phase to the first packet drop (recall that the bottleneck uses DropTail FIFO queuing), can be expressed as

$$\tau_l = \frac{W^{\max} - C}{\alpha\mu}. \tag{12}$$

#### 3.2.3 Buffer overflow

At $t_l$ in Fig. 1, some packet drops must occur. When the sender receives reports of packet loss, it decreases the current congestion window to $W = \beta W^{\max}$ and continues with the congestion avoidance phase. Let $W^{\max}$ now denote the value of the congestion window before this reduction takes place. If $B \geq (1-\beta)W^{\max}$, then the buffer is never empty. The flow continues to fully utilize the bottleneck, and thus maintains a high throughput.

#### 3.2.4 Multiple connections

When multiple flows are present, let the index $i$ denote a particular flow. Suppose that in steady-state operation synchronized losses occur for all flows with a period of $\tau_l$. That is, every $\tau_l$ seconds increase the congestion windows until the bottleneck's buffer is full. Since the losses are assumed to occur simultaneously (due to the DropTail mechanism), the period $\tau_l$ must be the same for all competing connections, even those with different RTTs (Fig. 2).

Let $W_i^{\max}$ denote the maximum congestion window of flow $i$. With synchronized losses each flow reduces its congestion window to $W_i = \beta W_i^{\max}$ at the same time. The bottleneck link, however, is still fully utilized at this point (see Sec. 3.2.3). During the steady state of a congestion avoidance phase (see Fig. 1), the congestion window of flow $i$ increases at the linear rate $\alpha$ resulting in the trigonometric formula:

$$\tan(\alpha) = \frac{W^{\max}(1-\beta)}{\tau_l}. \tag{13}$$

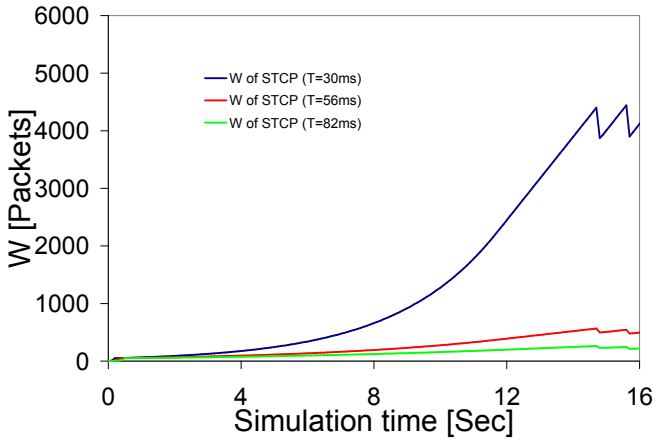


**Fig. 2.** Window growth for three competing STCP flows ($\mu$=1Gbps, $T_{base}$=30msec) experiencing synchronized losses with period $\tau_l$.

Turning this expression around so that $W^{\max}$ is the subject yields

$$W^{\max} = \frac{\tau_l \tan(\alpha)}{(1-\beta)}; \tag{14}$$

hence, the average congestion window of flow $i$ is

$$W^{\text{avg}} = \frac{\tau_l \cdot \tan(\alpha)(1+\beta)}{2(1-\beta)}. \tag{15}$$

The instantaneous throughput for flow $i$ can be approximated using Little's law:

$$\lambda_i(t) = \frac{W_i(t)}{T_i + D_i(t)}, \tag{16}$$

where $D_i(t)$ is the instantaneous waiting time in the queue for a packet from connection $i$. This delay has an initial value of $\frac{\beta \sum W_i^{\max} - C}{\mu}$, and attains a maximum value of $B/\mu$ when the buffer is full. At any given instant, the queuing delay $D_i(t)$ can be taken as a good measure of the average delay. It is also assumed that the average queuing delays $D_i$ are the same for all connections, where

$$D = \frac{\beta \sum W_i^{\max} - C + B}{2\mu}. \tag{17}$$

Combining Eqs. (15) and (16) yields the following formula for the average steady-state throughput ($\lambda_i$) of flow $i$:

$$\lambda_i = \frac{\tau_l \tan(\alpha)(1+\beta)}{2(T_i + D)(1-\beta)}. \tag{18}$$

It can be seen that the throughput is proportional to the congestion window increase parameter $\alpha$, monotonic in the decrease parameter $\beta$, and inversely proportional to the average round trip time.

### 3.3 STCP with random loss

#### 3.3.1 Single flow

TCP treats the loss of packets as a signal of network congestion and reduces its window when this occurs. Losses not due to congestion are mostly caused by transmission errors. If we assume that the probability for a packet to be lost due to link

errors is $q$, then on average $1/q$ packets are successfully transmitted before a loss occurs. More formally, we can write

$$\frac{1}{q} = \int_{t_l}^{t_l+\tau_l} \lambda(t)dt. \tag{19}$$

The limits of the integral are $t_l$, the last time that a loss occurred, and $t_l + \tau_l$, the expected time of the next loss. If we assume that $W(t_l) > C \; \forall \; t > t_l$, then according to Eq. (5) we have

$$\frac{1}{q} = \int_{t_l}^{t_l+\tau_l} \mu dt. \tag{20}$$

Hence, the expected duration of the next cycle can be expressed as Eq. (21),

$$\tau_l = \frac{1}{q\mu}. \tag{21}$$

According to Eq. (11), the congestion window increase over this interval will be given by $\Delta W_l = \alpha \; \mu \; \tau_l$. This allows us to note the equivalence

$$W_l = \beta W_l + \alpha \mu \tau_l, \tag{22}$$

assuming that $\beta W_l + \alpha \mu \tau_l - \mu T < B$. Combining Eq. (21) with Eq. (22) yields the relation

$$W_l = \frac{\alpha}{q(1-\beta)}. \tag{23}$$

The maximum of the congestion window is again proportional to the increase parameter $\alpha$ and monotonic in the decrease parameter $\beta$; it is also inversely proportional to the loss probability $q$. Hence, the average throughput of the flow is given by

$$\lambda = \frac{\alpha(1+\beta)}{2q(T+D)(1-\beta)}. \tag{24}$$

## 4 Numerical Results

For our numerical analysis we used a dumbbell network with one bottleneck link, as shown in Fig. 3. In all the scenarios described below, the capacity of the shared link was either 100 Mbps, 1 Gbps or 2.4 Gbps. Link delays vary from scenario to scenario. The two routers use FIFO queuing and DropTail buffer management. Their buffers were set to the bandwidth-delay product of the network. The bandwidth between hosts and their routers was 10 Gbps, and the link delay between hosts and their routers was 1 ms. The packet size was set to 1,500 bytes, and the maximum window size was large enough (83,000 packets) to saturate the bottleneck. We used TCP SACK [12] for both clients and servers. In addition to the TCP agents we used an FTP application to transmit large datasets. All FTPs begin transmitting simultaneously (at time zero), and the simulation ran for a total of 100 sec unless indicated otherwise. The individual traffic mixing scenarios are discussed in detail below.

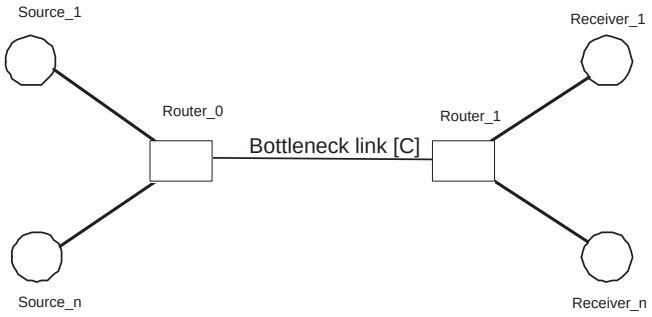


**Fig. 3.** Network topology with a single bottleneck link

### 4.1 Congestion window behaviour

To verify the two regimes of the window growth expressions presented in Sec. 3.2, a set of simulations were carried out for different RTT connections and 1 Gbps. Fig. 4 shows congestion window plot of 100 ms round trip time for both analytical and numerical analysis. The first regime of a congestion avoidance phase is exponentially increased followed by the linear increase regime. When loss occurs, the congestion window is reduced to $\beta W^{\max}$, then the window starts increase linearly until it reaches $W^{\max}$, this takes $\tau_l$ period, in each $\tau_l$ period the congestion windows increases until the bottleneck link's buffer is full. Due to this oscillation, steady state case is obtained.

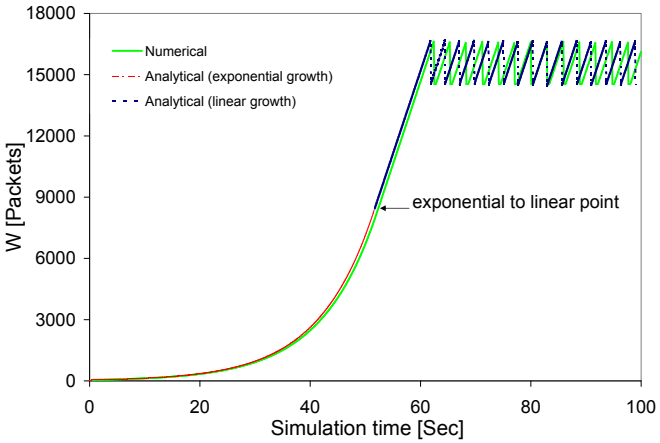


**Fig. 4.** $W$ of STCP flow with DropTail queue($\mu = 1Gbps$, $RTT = 100ms$)

### 4.2 Lossy link condition

According to our loss (link failure) model presented in Sec. 3.3, any packet served at the forward link may be lost with probability $q$, and such losses are independent from each other. To focus on the effect of random loss on the congestion window we execute a set of simulations with different RTTs and $q$ loss probabilities. We used the simulator's error model to simulate losses in the bottleneck link. This loss model was set to drop a packet with a defined average drop rate. To observe the behaviour of the performance of STCP flows with different round trip time when subjected to systemic losses, we simulated two STCP flows competing on bottleneck link, the bottleneck bandwidth was 1 Gbps and the buffer size was large enough not to have tail drops. Hence, the congestion window

was controlled by the introduced random loss instead of buffer overflow. Fig. 5 shows the congestion window plot for the two STCP flows when the round trip times were 30 ms and 56 ms for flow_0 and flow_1 respectively. The probability of loss was 1.5e-4. The Fig. 5 nicely shows that the numerical result coincides with Sec 3.3's analysis. Fig. 6 shows comparison between the numerical throughput achieved by the simulation and the analytical analysis presented in Sec 3.3. One can see that the two results are almost identical.

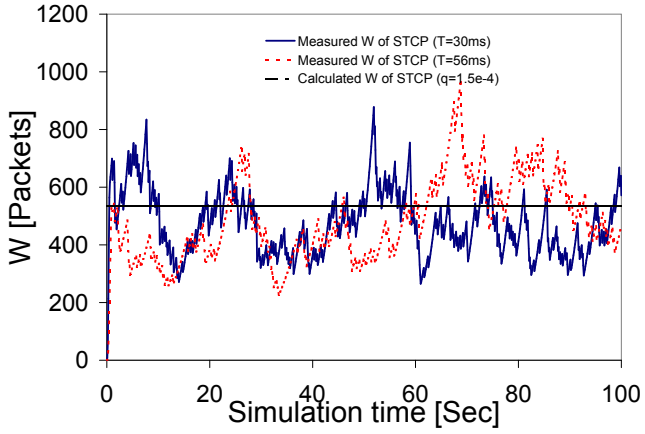


**Fig. 5.** $W$ of two STCP flows under random loss ($\mu = 1$Gbps, $q = 1.5e-4$)

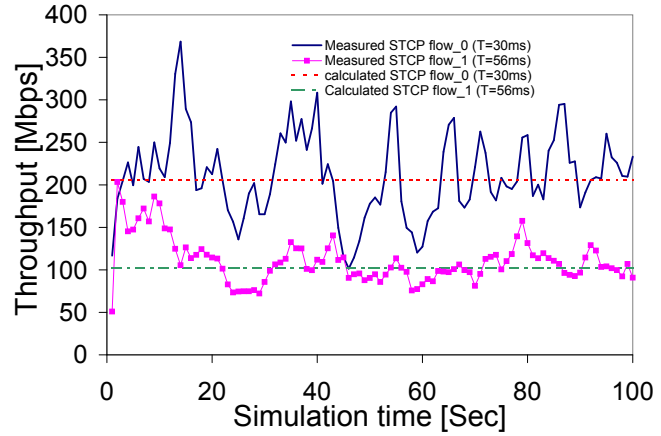


**Fig. 6.** Throughput of two STCP flow under random loss ($\mu = 1$Gbps, $q = 1.5e-4$)

### 4.3 Fairness

Fairness refers to the relative performance of several connections under the same TCP variant. To provide a single numerical measure for the criterion, we use Jain's Fairness index [7]:

$$\text{FairnessIndex} = \frac{(\sum_{i=1}^{n} \lambda_i)^2}{n \sum_{i=1}^{n} \lambda_i^2}, \qquad (25)$$

where $\lambda_i$ is the throughput of the $i^{th}$ flow and $n$ is the total number of flows.

We simulated the activity of multiple flows with varying RTT values on the network. The RTTs were chosen according to the following formula[1]: for flow $i$, $\mathrm{T}_i = (i+1)\mathrm{T}_{\text{base}} - 4i$ ms ($i = 0...n-1$, where $n$ is the number of flows). Both $n$ and $T_{base}$ were varied to provide 9 different flow sets. Finally, we tested these scenarios under two types of queue management: DropTail and adaptive RED.

**DropTail queue** simply drops every packet that arrives at the buffer as long as the buffer is full. Fig. 7 shows the congestion windows as a function of time for three competing Scalable TCP flows. One can easily see that the connection with the highest round trip time gets the smallest congestion window in this case. If one calculates the congestion window for the DropTail case using the Eqs. (9) and (11) this coincides with numerical results also shown in Fig. 7. The window size and the round trip time together determine the achieved throughput.

Fig. 8 reveals that in Scalable TCP when a bottleneck link is shared between different (RTT) connections, the shortest RTT connection out-placed the others. It is important to notice that the Fig. 7 nicely shows the numerical result coincides with Sec. 3.2.4's analysis.

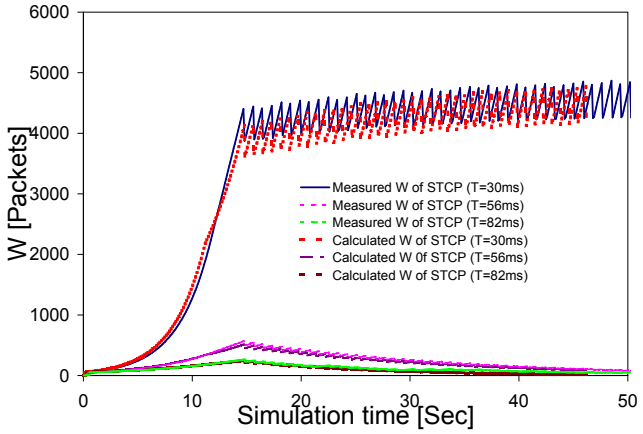


**Fig. 7.** $W$ vs. $t$ for three STCP flows under DropTail queuing ($\mu = 1$Gbps, $T_{base} = 30$msec)

Table 1 shows the Jain's Fairness Index for all the flow sets and $\mathrm{T}_{\text{base}}$ values simulated using DropTail queuing. It can be seen that in any cases the Fairness Index in Scalable TCP degrades when DropTail system is used whereas RED queue system (see below) can acheive better share of bottleneck link (see Table 2).

**Tab. 1.** Jain's Fairness Index for STCP under DropTail queuing ($\mu$ =1Gbps)

| **RTT** vs. | **Number of flows** | | |
|---|---|---|---|
| **Bandwidth** | **3** | **5** | **8** |
| 30ms | 0.34 | 0.21 | 0.13 |
| 60ms | 0.34 | 0.21 | 0.13 |
| 100ms | 0.35 | 0.21 | 0.14 |

[1] without any specific reasons

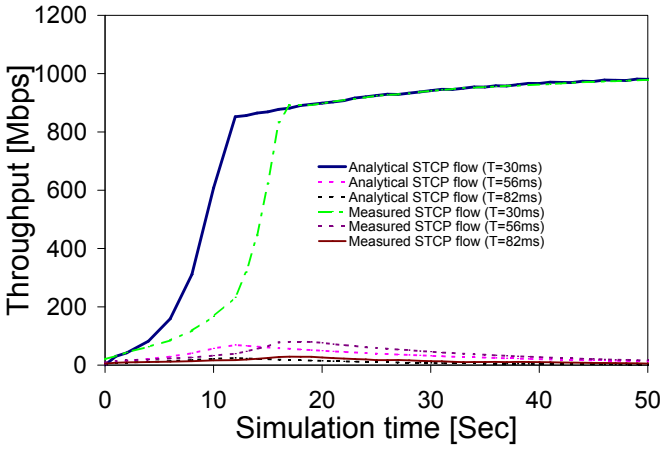


**Fig. 8.** Throughput of three STCP flows under DropTail queuing ($\mu = 1Gbps$, $T_{base} = 30msec$)

**Adaptive RED queue** is used to mitigate unfairness in high-speed networks. This mechanism allows a router to control congestion and keep network utilization high by dynamically updating the random loss probability in the buffers. This keeps the average queue size close to some target queue size. In this scenario we use adaptive RED queue management on the bottleneck link buffer. The scenarios simulated for this method are identical to those described for DropTail queuing. Fig. 9 shows typical behaviour of the congestion windows for five competing flows. One can see that under adaptive RED, connections with different RTTs achieve similar congestion windows. Flows with identical congestion windows, however, will have throughputs proportional to $\frac{1}{RTT}$. Under this scenario very different rates of throughput are achieved (Fig.10).

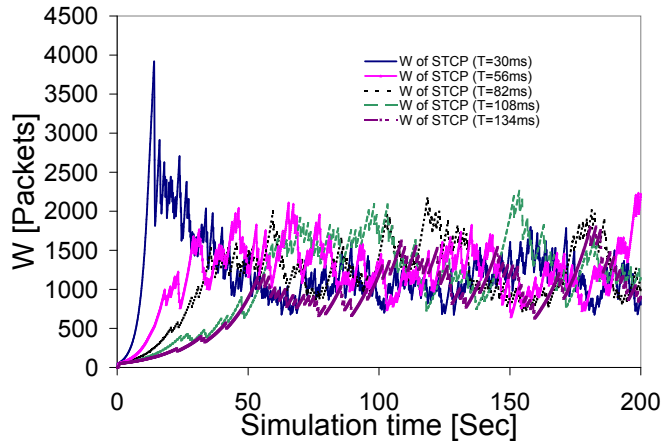


**Fig. 9.** $W$ vs. $t$ for five STCP flows under adaptive RED queuing ($\mu = 1$Gbps, $T_{\text{base}} = 30$msec)

To further quantify the fairness of Scalable TCP flows Table 2 shows the Jain's Fairness Index for different number of flows and $T_{\text{base}}$ values using adaptive RED queue. It can be seen that in any cases the Fairness Index of different RTT connections is inversely proportional to round trip time of the flows.

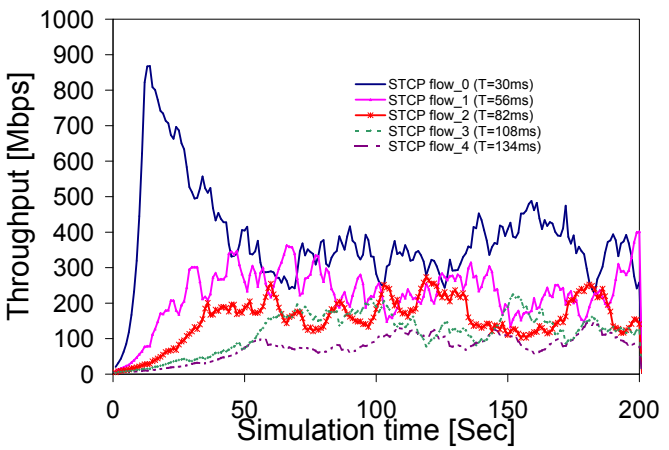


**Fig. 10.** Throughput of five STCP flows under adaptive RED queuing ($\mu = 1Gbps$, $T_{\text{base}} = 30$msec)

**Tab. 2.** Jain's Fairness Index for STCP flows under adaptive RED ($\mu$ =1Gbps)

| **bandwidth** vs. | **Number of flows** | | |
|---|---|---|---|
| **RTT** | **3** | **5** | **8** |
| 30ms | 0.86 | 0.7 | 0.6 |
| 60ms | 0.77 | 0.58 | 0.42 |
| 100ms | 0.78 | 0.44 | 0.4 |

## 5 Conclusion

In this work we have analysed the behaviour of the Scalable TCP (STCP) protocol, which is a variant of the TCP standard, and is designed for high speed wide area networks. Our goal was to understand and model the exact behaviour of STCP in order for a later work to derive some adaptive parameter setting mechanisms for the parameters $\alpha$ and $\beta$. In order to accomplish this task, we have analytically analysed the STCP for slow start and congestion avoidance regimes and for DropTail queues and random loss links. Our analysis verifies that STCP throughput is inversely proportional to the connection's round trip time (RTT) and when multiple flows with different RTTs compete for a single bottleneck link, the set up results in a highly unfair situation where basically the flow with the shortest RTT suppresses all the others. This is also true with random link losses. Our analytical results are compared against some numerical results, where very exact matches can be found between the results.

Besides, we have also shown that even thought different RTT based STCP flows can operate with Random Early Drop (RED) buffer management regime, their overall fairness in throughput remains poor. This encourages us to investigate the adaptive parameter settings for STCP, which work is in progress now.

## References


1 **Altman E, Avrachenkov KE, Kherani AA, Prabhu BJ**, *Analysis of Scalable TCP in the presence of Markovian Losses*, PFLDnet 2005 workshop, 2005 February.

2 ———, *Fairness in MIMD Congestion Control Algorithms*, IEEE Infocom 2005, Miami (2005 March).

3 **Chiu D, Jain R**, *Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks*, Computer Networks **17** (1989), no. 1, 1-14.

4 **Firoiu V, Borden M**, *A study of active queue management for congestion control*, Proc. INFOCOM, 2000, pp. 1435-1444.

5 **Floyd S**, *Limited slow-start for TCP with large congestion windows*, Technical Report (Mar 2004). RFC 3742, IETF.

6 **Jacobson V**, *Congestion Avoidance and Control*, SIGCOMM 1988, conference (1988 aug).

7 **Jain R**, *The art of computer systems performance analysis*, New York, QA76.9.E94J32 (1991).

8 **Jin C, Wei DX, Steven H. Low**, *FAST TCP: motivation, architecture, algorithms, performance*, IEEE INFOCOM (2004).

9 **Katabi D, Handley M, Rohrs C**, *Internet congestion control for high bandwidth-delay product networks*, ACM Sigcomm 2002 (August, 2002).

10 **Kelly T**, *Scalable TCP: Improving Performance in Highspeed Wide Area Networks*, 2002, available at `http://www-lce.eng.cam.ac.uk/~ctk21/scalable/`. submitted for publication.

11 **Lakshman TV, Madhow**, *The performance of TCP/IP for networks with high bandwidth-delay products and random loss*, IEEE/ACM Transactions on Networking **5** (July 1997), no. 3, 336-350.

12 **Mathis M, Mahdavi J, Floyd S, Romanow A**, *TCP Selective Acknowledgment Options*, Technical Report (October 1996). IETF.

13 **Ramakrishnan KG, Floyd S**, *A Proposal to add Explicit Congestion Notification (ECN) to IP* (15 jan 1999), no. 2481, 25.

14 **Tekala M, Szabó R**, *Evaluation of Scalable TCP*, AICCSA-05, 3rd IEEE conference in Egypt (2005 jan).