TCP Kikolo for Performance Improvement in High-speed Networks in Presence of Packet Losses

Paul Kiryowa and Fabrizio Granelli
DISI - University of Trento
Via Sommarive 14, I-38050 Trento, Italy
E-mail: paul.kiryowa@studenti.unitn.it, granelli@disi.unitn.it

Dzmitry Kliazovich
University of Luxembourg
6, rue Coudenhove Kalergi, L-1359 Luxembourg
E-mail: dzmitry.kliazovich@uni.lu

***Abstract*—TCP shows poor performance in high speed networks with large latencies as well as in wireless networks where link error rates are high. The main two factors degrading TCP performance are related to its conservative policies for sending rate increase and treating all packet losses as congestion-related. In order to address these limitations a number of solutions have been proposed. Most of them either adapt to the large bandwidth-delay product when the loss rate is low (below 10^{-7}) or face the problem of high error rate (above 10^{-3}) not accounting for high-speed connections. In this paper we propose a novel approach Kikolo TCP. It is based on Markov Modulated Poisson Process (MMPP) to control the congestion window evolution. Kikolo TCP offers improved performance in environments with high error rates. At the same time it ensures bandwidth scalability in low loss environments, TCP friendliness and fairness.**

Keywords- TCP congestion control, high-speed networks, fairness

I. INTRODUCTION

The increased use of the Internet to transmit data ranging from simple text messages to heavy flows produced by the content distribution applications requires the development of new transmission systems able to support the ever increasing demand for online services. The Transmission Control Protocol (TCP) is the most widely used protocol by applications mainly due to the offered reliability and congestion control. Designed in the early days of the Internet, TCP treats all packet losses as congestion-related and triggers dramatic reduction in sending rate to avoid congesting the links. As a result, TCP fails to deliver the required performance level, when used in the high-speed networks with long propagation delays as reported in [6-11]. Not being aggressive enough, it under-utilizes the available end-to-end resources. In order to mitigate these limitations of standard TCP[1] a number of solutions have been proposed [1,6,7,9-11].

Standard TCP uses Time Out (TO) and Duplicate ACKnowledgement (DUPACK) events to determine the presence of congestion. Each time congestion is detected, the sending rate is reduced to half, which is one of the main reasons for poor performance in high-speed networks as underlined in [6,7,9,11]. The high speed solutions proposed still perform similar to the standard TCP under high loss environments leading to underutilization of network resources.

[1] Standard TCP refers to TCP Newreno modification

The need for a scalable, TCP friendly and a fair protocol motivates the proposal of a new algorithm able to meet all such requirements over a wide range of operational scenarios. The proposed TCP modification, called Kikolo[2] TCP, uses a Markov Modulated Poisson Process (MMPP) [3] to define the evolution of the congestion window. The MMPP is a subclass of the doubly stochastic Poisson processes. It is used to model time varying arrival rates and the correlations between inter-arrival times. MMPP modeling is shown to capture the dynamics of Internet traffic well [4], which is typically characterized as bursty, self-similar, and long-range dependant.

An MMPP is characterized by a state S_n, and the time until the next renewal T_n, where n = 0, 1, 2, …. The time T_n is exponentially distributed with a parameter dependent on the state S_n. The modeling of the congestion window evolution with an MMPP helps capturing the Internet traffic behavior naturally (where ACK and DUPACK arrivals are the two stochastic processes).

Kikolo TCP uses a modified Additive Increase Multiplicative Decrease (AIMD) algorithm to adjust the congestion window size. This provides scalability in high speed networks and increased resilience in high error rate environments, leading to the improved performance in scenarios where traditional[3] and high-speed[4] TCP modifications under perform.

The rest of the paper is organized as follows: Section II presents state-of-the-art and related works; Section III describes the core Kikolo TCP concepts; Section IV presents performance evaluation results; and Section V concludes the paper outlining future research directions on the topic.

II. RELATED WORK

A. TCP congestion control

TCP connection begins with the “slow start” phase, characterized by exponential increase of the transmission rate. The congestion window defines the amount of data outstanding in the network without an acknowledgement.

[2] “Kikolo” is a Luganda word, a language spoke in Uganda, that means plant propagation (grafting) - a natural or artificial process of distributing plants.

[3] Traditional TCP Variants include TCP Newreno, TCP Vegas, TCP Westwood+

[4] High-speed TCP variants include BIC TCP, Compound TCP, High-speed TCP and Scalable TCP

During slow start, TCP sender increases the congestion window size by one for every received acknowledgement resulting in exponential growth.

As soon as the first packet loss is detected or a timeout is reached, the congestion avoidance phase replaces the slow start phase. During congestion avoidance, the window is increased linearly by 1/(Round Trip time (RTT)) for every acknowledgement received in sequence.

When a packet loss occurs, a new parameter called slow start threshold (*ssthresh*), initially initialized to the maximum allowable window size, is set to a half of the congestion window size. If the loss is detected via timeout mechanism, TCP triggers the slow start again until *ssthresh* is reached. Then, the congestion avoidance phase follows [5].

TCP Vegas [1] employs a different window evolution mechanism, which is delay-based. TCP Vegas predicts growing network congestion and, hence, it reacts before congestion losses occur. To do so, it observes the difference between the expected and the actual throughput [1]. Any increase in this difference corresponds to the increased queuing in bottleneck buffers. Based on such measures, TCP Vegas reduces its sending rate before congestion builds up.

TCP Westwood [2,10] performs bandwidth estimation available to a flow by filtering ACK inter-arrival times at the sender node.

Compound TCP [7], is a synergy between delay- and loss-based approaches. Its delay-based component is used to increase the sending rate when the network is under-utilized. It prevents self-induced congestion and provides RTT fairness during congestion avoidance phase. The loss-based component adjusts the congestion window in a way similar to the standard TCP and mainly operates when network becomes heavily congested.

BIC TCP [9] is based on the binary search technique, where the sender tries to estimate optimal operating point (path capacity) and receives a binary feedback from the network. The congestion window evolution is linear for small window sizes and becomes logarithmic as the sender approaches available end-to-end capacity.

Scalable TCP [6] exponentially increases its congestion window both in the slow start and congestion avoidance phases. As a result, it shows constant time for recovering from any window reduction back to the path capacity. On the other hand, as the resulting rate approaches the end-to-end capacity may become the reason for multiple packet losses.

High-speed TCP [11] is based on AIMD but using varying increase and decrease.

III. Kikolo TCP Algorithm

Kikolo TCP is a loss-based protocol whose window evolution is based on MMPP model with two levels L and C, and four states S0, S1, S2, S3. The state diagram describing the model is provided in Fig. 1.

Kikolo TCP uses two schemes for window increase: a coarse scheme used to perform an aggressive increase, and a refined scheme used to increase the congestion window in a controlled way when approaching the bottleneck capacity. Congestion window reduction is done according to the mechanism presented in Table I.

The congestion window scale is subdivided into three portions (see Table I): the first level applies to the initial slow start algorithm, the second level applies to the modified additive increase technique and the third level applies to the additive increase technique proposed in [5]. The congestion window evolution is controlled by a number of parameters (default values are indicated in parentheses):

ρ (= 4) is used to ramp up the gradient of the Additive Increase phase. γ (= 0.005) determines a "*safe*" region over which the ramp up of the additive slope is applied, the ramp-up phase. $\beta3$ (= 0.95) represents a mild reduction rate for window size lower than the *ssthresh*. $\beta1$ (= 0.85) is an intermediate reduction rate for window sizes close to *ssthresh*. $\beta2$ (= 0.75) is a strong reduction rate for window sizes that are almost twice *ssthresh*.

Kikolo TCP modifies only the congestion avoidance mechanism of standard TCP, leaving the initial slow start mechanism unchanged. During congestion avoidance the following actions are taken.

1) When an in-sequence ACK is received

The evolution of the congestion window starts from the initial state S0. In this state, a transmission is attempted and, when an ACK is received, Kikolo TCP moves from state S0 to state S1 increasing the congestion window size by an amount proportional to the size of the current congestion window. As more in-sequence ACKs are received, Kikolo TCP moves to the state S2 with increased congestion window size. While in the state S2, Kikolo TCP switches between *coarse* and *refined* increment schemes in order to control the amount of self-induced congestion and, at the same time, ensure scalability. The *coarse* scheme increases the congestion window by (1/cwnd or $\boldsymbol{\rho}$/cwnd), while the *refined* scheme increases the window size by 85% of (1/cwnd or $\boldsymbol{\rho}$/cwnd). States S0 and S1 apply only the coarse increment scheme.

2) When a DUPACK is received

Kikolo TCP does not react to the first received DUPACK. When a second DUPACK is received, Kikolo TCP returns to the state S0 from the state S1 with a congestion window reduced by an amount proportional on the size of the window prior packet loss detection. Fast retransmit is then performed.

The transition from the state S2 to the state S3 is triggered by the reception of the second DUPACK, given that the first DUPACK was received in the state S2. This makes the transition to the C level possible only when *two consecutive DUPACK* messages are received, which may be an indication of growing congestion levels. On the other hand, the transition from the state S0 to the state S1 or from the state S3 to the state S2 is triggered by the reception of a *single ACK message*. However no window increase is performed as it may signal just a possible ease in the congestion levels or triggered by packet reordering.

3) When a timeout occurs

When the first timeout (TO) occurs, Kikolo TCP triggers the slow start phase without changing its state. The congestion window is exponentially increased until *ssthresh* is reached.

TABLE I. CONGESTION WINDOW EVOLUTION PHASES

	Slow Start	**Congestion Avoidance**	
	cwnd< ssthresh	**< (1 + γ)∗ssthresh**	**>=(1 + γ)*ssthresh**
Coarse Window Increase	cwnd+1	cwnd+1+ρ/cwnd	cwnd + 1/cwnd
Refined Window Increase	cwnd+1	cwnd+1+ β1*ρ/cwnd	cwnd+ β1*1/cwnd
Window Decrease	β3*cwnd	β1*cwnd	β2*cwnd

Then, between *ssthresh* and (1 + γ)**ssthresh* the congestion window is increased using additive increase scaled with the ramp-up factor. Then, the congestion window is increased following the *coarse* and *refined* schemes described in Table I.

In case of no two successive DUPACK messages received Kikolo TCP increases the window proportionally to the current size of the window.

The state probabilities associated with the state described above are defined as P(L= i, C = i); i = 0, 1. For simplicity, the following notation is used: P(C,0) = P(C = 0, L = 0), P(C,1) = P(C = 1, L = 0), P(L,0) = P(C = 0, L = 1), P(L,1) = P(C = 1, L = 1).

B. Equations for the state diagram

The equilibrium equations for the state diagram with four states and two levels are given below.

$$\lambda P(C,0) = (\mu + \lambda)P(L,0) \quad (1)$$

$$\mu P(C,0) = (\mu + \lambda)P(C,1) \quad (2)$$

$$\lambda P(C,0) = \mu P(L,0) + \mu P(C,1) \quad (3)$$

$$\mu P(L,1) = \lambda P(L,0) + \lambda P(C,1) \quad (4)$$

$$\lambda,\ \mu \geq 1 \quad (5)$$

These equations are solved, based on the following assumptions $\lambda = \mu$, $\lambda = 10\mu$, $\lambda = 20\mu$, $\lambda = 30\mu$, where λ is the arrival rate of ACKs and μ is the arrival rate of DUPACKs. The obtained analytical state probabilities are presented in Fig. 5. They demonstrate the stability of the model and can be used to determine the increase and decrease rates of the AIMD or MIMD schemes.

C. State diagram

The state diagram is shown in Fig. 1. The MMPP model has two levels and four states. The arrow with ACK annotation indicates the arrival rate λ of ACKs, which activates the transition from C to L level, thus triggering the additive increase. The arrow with DUPACK indicates the arrival rate μ of DUPACKs which triggers a transition from L to C level upon the reception of two successive *DUPACKs* followed by a reduction of the congestion window (this is different from standard TCP which reduces congestion window after reception of three DUPACKs*)*. The transition from the state S3 to the state S1 is triggered by reception of a third DUPACK message.

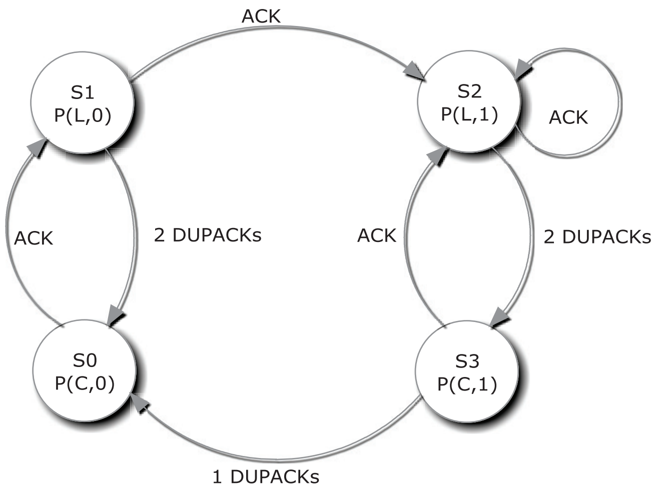

Figure 1. MMPP model.

D. Kikolo TCP window evolution

Fig. 2 compares Kikolo TCP and TCP Newreno congestion window evolution algorithms. It is plotted for a bottleneck of 10 Mbps and 60 ms one way delay with queue size equal to 10% of the Bandwidth Delay Product (BDP). The Packet Error Rate (PER) is 10^{-5}. The slow start, ramp-up, coarse and refined phases are highlighted.

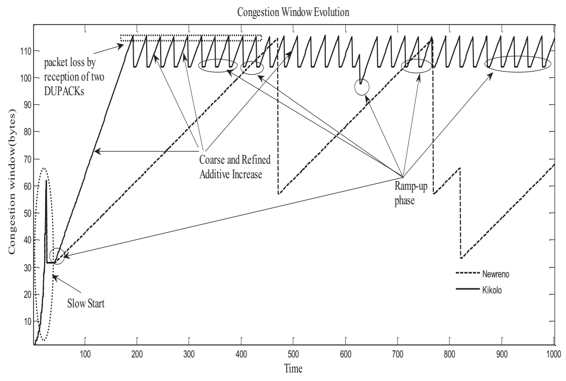

Figure 2. Congestion window evolution.

IV. PERFORMANCE EVALUATION

Performance evaluation is carried using the network simulator Ns2 [12]. The simulated dumbbell topology is presented in Fig. 3. The source nodes S are connected to router N1, while destination nodes D are connected to router N2 with 10 Gb/s, 1ms links. All flows share a single bottleneck link between the routers N1 and N2. The duration of each simulation is 1000 seconds.

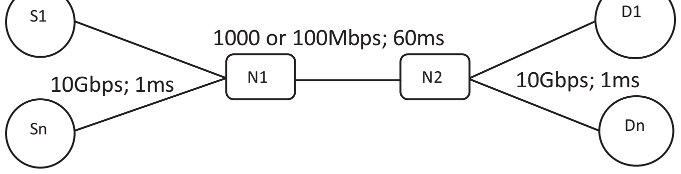

Figure 3. Simulation topology.

A. Homogeneous TCP connections

In this section we analyze the RTT fairness (of 4 flows) and throughput (of 15 flows) homogeneous TCP connections of traditional TCP variants, high-speed TCP variants and Kikolo TCP sharing a bottleneck in under random packet losses. This experiment aims at comparing the efficiency of TCP protocols under varying bottleneck link capacity and RTT fairness under varying PER.

1) TCP states

The analytical and simulation state distribution probabilities of Kikolo TCP are shown in Fig. 4. Simulation results demonstrate the ability of the model to converge to a steady state, spending most of the time in state S2, for PERs smaller that 1%. In case error rates or congestion increase, Kikolo TCP changes states trying to find an optimal window size that will reduce the presumed congestion. The analytical state distribution probabilities shown with line graphs are in the agreement with the simulation results, indicating that Kikolo TCP attains a steady state in state S2.

2) TCP Throughput

This section compares the throughput performance of standard TCP, high-speed TCP variants and Kikolo TCP. Fig. 5 shows that Scalable TCP, Kikolo TCP, BIC TCP and Compound TCP achieve higher throughput than High-speed TCP, and standard TCP variants for PERs lower than 10^{-4}. However, as PERs increase and become greater than 10^{-4} Kikolo TCP outperforms all traditional and high-speed TCP variants.

The simulation results, presented in Fig. 6, indicate that BIC TCP and Kikolo TCP achieve almost 100% of the link utilization for PERs lower than 10^{-4}, followed by Scalable TCP, TCP Vegas, TCP Westwood+, TCP Newreno, High-speed TCP and Compound TCP. However, as PERs increase above 10^{-3} Kikolo TCP outperforms all the considered TCP variants.

3) RTT fairness

This section compares RTT fairness of standard TCP, high-speed TCP variants and Kikolo TCP as a function of PER. The metric chosen for performance evaluation is the ratio between the throughput connections with short RTT to the throughput of the connections with longer RTT. We simulated two flows with RTT = 40ms competing for bottleneck resources with two flows with RTT = 240ms, over 100Mbps, 10ms delay link. Fig. 9 shows that Kikolo TCP's RTT fairness is comparable to that of traditional TCP variants and that of Compound TCP, BIC TCP and High-speed TCP follow in fairness. Scalable TCP is the most unfair protocol when the PER is less than 10^{-4}, this is due to its excessive scalability. However, as PERs increase and becomes greater than 10^{-4} all TCP variants perform the same, this is because high-speed and traditional TCP variants perform similar to standard TCP.

B. Heterogeneous TCP connections

This section compares the performance of Kikolo TCP when competing for bottleneck resources with other TCP variants. The focus of this simulation setup is devoted to underline the aggressiveness and friendliness properties of Kikolo TCP towards other TCP variants. In this setup, five Kikolo TCP flows share the same bottleneck with five flows of the concurrent TCP version. The main metric chosen for

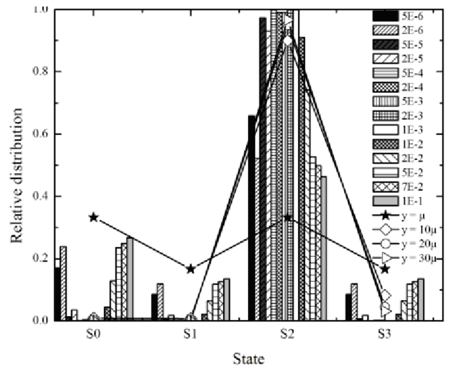

Figure 4. State probability distribution under different error rates.

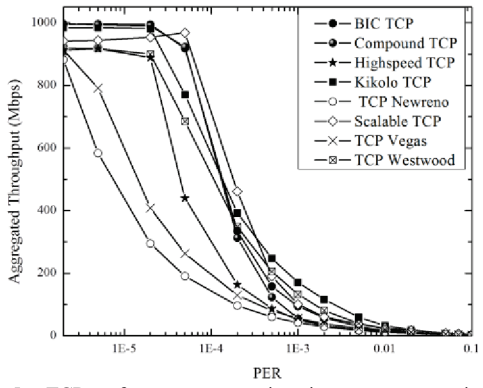

Figure 5. TCP performance comparison in error-prone environment (1Gb/s bottleneck link).

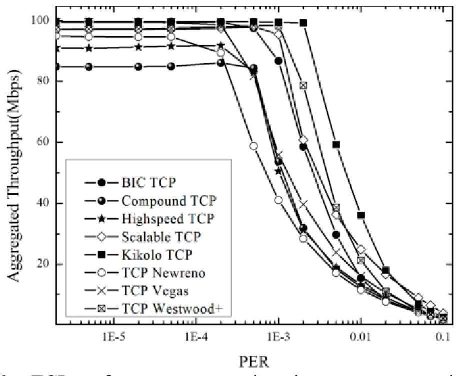

Figure 6. TCP performance comparison in error-prone environment (100Mb/s bottleneck link).

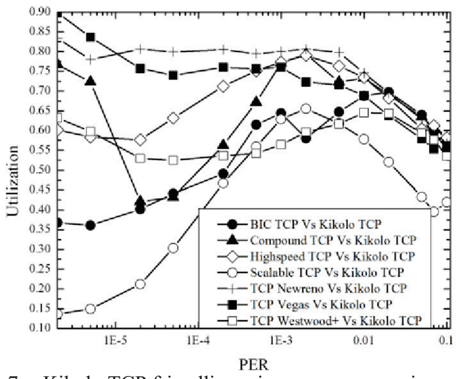

Figure 7. Kikolo TCP friendliness in error prone environment (over 1Gb/s bottleneck link)

performance evaluation is the ratio between the throughput achieved by Kikolo TCP connections and the total aggregated throughput traversing the bottleneck link.

1) Link Utilization

Fig. 7 shows that Kikolo TCP is friendly towards traditional TCP variants for all error rates. This is due to the fact that Kikolo approaches the available end-to-end capacity gently and avoids monopolizing the bottleneck buffer space. Kikolo TCP demonstrates excellent friendliness for PERs above 10^{-4}. However, when PERs are below 10^{-4}, it suffers considerably while competing with Scalable TCP and BIC TCP because of their higher aggressiveness. Comparing with Compound TCP, Kikolo TCP remains friendly for PERs above 10^{-5}. However, as PER is reduced Compound TCP behaves as a standard TCP. The delay-based component reduces its aggressiveness leaving the weak loss-based component that can't properly scale in high-speed networks.

Fig. 8 compares the friendliness of Kikolo TCP towards high-speed and traditional TCP variants. The simulation results indicate that Kikolo TCP is friendly towards all TCP variants for PERs above 10^{-4}, because all the considered variants perform poorly in high loss environments. As PER is reduced Kikolo TCP obtains more than 50% of its share. The reason for such unfriendliness is due to the fact that for small window sizes the other TCPs do not scale very well, remaining similar to standard TCP. On the other hand, Kikolo TCP can scale well under a wide range of window sizes and is resilient to high bit error rates.

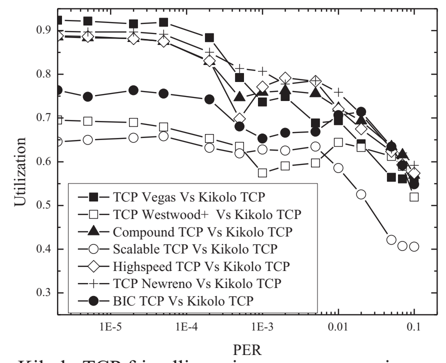

Figure 8. Kikolo TCP friendliness in error prone environment (100Mb/s bottleneck link)

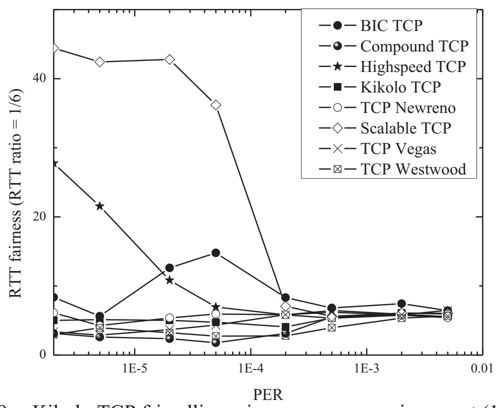

Figure 9. Kikolo TCP friendliness in error prone environment (100Mb/s bottleneck link)

V. CONCLUSION

This paper presents a novel TCP modification, called Kikolo TCP. Its window evaluation algorithm is based on a Markov Modulated Poisson Process (MMPP), whose states refine the behavior of window increase and decrease factors under changing network conditions. The obtained simulation results underline two main features of Kikolo TCP: scalability for high-speed networks and high performance in the presence of link errors.

Kikolo TCP outperforms most of the available protocols for high-speed networks. Kikolo TCP approaches the available end-to-end network capacity gently. This allows maintaining a good friendliness with traditional TCP variants, but greatly degrades the protocol performance in presence of highly aggressive BIC TCP and Scalable TCP.

Future work will focus on Kikolo TCP implementation in OS Linux and wide-area network experiments.

REFERENCES

[1] Brakmo, L., O'Malley, S., and Peterson, L., "TCP Vegas: New techniques for congestion detection and avoidance," Computer Communication Review, vol. vol.24, no.4 pp. 24-35, 1994.

[2] Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M. Y., and Wang, R., "TCP westwood: End-to-end congestion control for wired/wireless networks," Wireless Networks, vol. 8, no. 5, pp. 467-479, 2002.

[3] Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S.Trivedil, Queueing Networks and Markov Chains, second edition ed. 2006.

[4] Heyman, D. P. and Lucantoni, D., "Modeling multiple IP traffic streams with rate limits," Ieee-Acm Transactions on Networking, vol. 11, no. 6, pp. 948-958, 2003.

[5] Jacobson, V., "Congestion avoidance and control," Computer Communication Review, vol. vol.18, no.4 pp. 314-329, 1988.

[6] Kelly, T., "Scalable TCP: Improving performance in highspeed wide area networks," Computer Communication Review, vol. 33, no. 2, pp. 83-91, 2003.

[7] Kun, T., Jingmin, S., Qian, Z., and Sridharan, M., "A compound TCP approach for high-speed and long distance networks," 25Th Ieee Infocom Conference, pp. 1217-1228, 2006.

[8] Lijuan, W., Fei, P., and Leung, V., "Dynamic congestion control to improve performance of TCP split-connections over satellite links," Proceedings.13th International Conference on Computer Communications and Networks (IEEE Cat.No.04EX969), pp. 268-272, 2004.

[9] Lisong, X., Harfoush, K., and Injong, R., "Binary increase congestion control (BIC) for fast long-distance networks," Ieee Infocom 2004, pp. 2514-2524, 2004.

[10] Ren, W., Valla, M., Sanadidi, M., and Gerla, M., "Adaptive bandwidth share estimation in TCP Westwood," GLOBECOM'02 - IEEE Global Telecommunications Conference.Conference Record, pp. 2604-2608, 2002.

[11] S.Floyd, "HighSpeed TCP for Large Congestion Windows," The Internet Society, 2003.

[12] Teerawat Issariyakul and Ekram Hossain, Introduction to Network Simulator NS2, first edition ed. 2008.