A Novel Approach to Model TCP Traffic

Anh-Tuan Trinh, Sándor Molnár

High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, H-1117, Magyar tudósok körútja 2., Budapest, Hungary, E-mail: {trinh,molnar}@tmit.bme.hu

***Abstract*— In this paper, a state-based modelling of TCP traffic is presented. During a connection, TCP stays in either of the following states: Slow Start, Congestion Avoidance, Loss Recovery (Fast Recovery and/or Fast Retransmit) and Time Out. We consider the states of a TCP connection as the *phases* of a stochastic process. We propose the use of the discrete-time batch Markovian arrival process (D-BMAP) to model the traffic generated by a TCP connection. The main contributions of the paper are the followings. Firstly, we provide *a simple unified model* for some well-known versions of TCP based on the D-BMAP process. Secondly, we introduce a new concept, namely the *TCP characterization matrix* for a TCP connection that characterizes the transition probabilities between the states of TCP. This matrix is crucial in our state-based analysis. Thirdly, we present a technique to detect the states of TCP. We have developed our technique into a tool called TCP-ASD that automates state detection of a TCP connection. Our tool can automatically detect the beginning and the end of the states of TCP and thus the sojourn time distributions as well as other statistics that we use in our analysis. We also discuss the trade-offs between simplicity and accuracy in the state-based approach. Finally, we use simulation and numerical analysis to validate our proposed model.**

I. Introduction

TCP modelling can be found in two main levels: packet level and flow (fluid) level. One of the motivations for the packet level approach is the possibility of applying existing discrete-time models [3],[9]. Respectively, the motivation for fluid level model is the possibility of applying existing continuous-time (control-theoretic) models, [4],[5],[8], to name a few. In both approaches, good points have been addressed and important, subtle results have been achieved. In [8], T. Ott *et al* used stochastic differential equations to model TCP behavior and first suggested the well-known *square-root* formula. J. Padhye *et al* in [9] extend the model in [8] to capture Time Out. This model is widely accepted as one of the most accurate models for TCP Reno (in the case of bulk data transfer). We can also mention here the chaotic nature of TCP as suggested and examined in [10]. However, as TCP modelling is application-sensitive, a general purposed TCP model that is precise, yet simple, is still unavailable. This makes the TCP modelling task still very challenging. Another issue of TCP modelling is the type of modelling: black-box modelling and white-box modelling. Black-box modelling approaches usually start from a theoretical model while white-box modelling approaches try to mimic inherent operations of TCP based on some statistics. An example of white-box modelling is the well-known ON/OFF model for voice traffic. It has two states: SILENCE and SPEAK. If the speaker speaks, then it is in the SPEAK state, and it is in the SILENCE state otherwise. With some probability the process jumps from SILENCE state to SPEAK state and respectively, with some probability the process jumps from SPEAK state to SILENCE state. So if the sojourn time distributions at the state are exponential (continuous time) or geometrical (discrete time), then the background process can be modelled by a two-state Markov chain and the traffic generated by voice sources can be well modelled by a Markov Modulated Poisson Process (MMPP), [2]. A natural question arises then: How about TCP traffic? We model TCP by its states. During a connection, TCP stays in any of the following states: Slow-Start, Congestion Avoidance, Fast Recovery, Exponential Back-off. TCP can jump from one state to another state in response to external events such as packet loss or Time Out. We consider how much time TCP stays in each state and the distribution of time elapsed at each state. We then consider the jumping probability from one state to another state. From the statistics, we can build a model to estimate TCP throughput. Last but not least, on the one hand, we agree that square-root-style models of TCP based on important metrics like packet loss probability and average round-trip time are significant steps forward. We argue, on the other hand, that these metrics are not necessarily the *only* metrics that TCP models should be based on. We could just mention other metrics of TCP as the sojourn time distributions of TCP at different states, the probabilities that TCP jumps from a state to another state during a connection, the distributions (as well as the expectations) of the number of packets sent in one round-trip time in different states, etc. In this paper, we not only present a state-based model to estimate the long term throughput of TCP based on new metrics of TCP but also try to build the bridges between our model and the existing models.

The remainder of the paper is organized as follows. In Section II we present our state-based model for TCP. A tool for validation is briefly described in Section III. Section IV provides validation results of the proposed model. Finally, Section V concludes the paper.

II. A D-BMAP model for TCP stationary throughput

A. The general case

The D-BMAP process was originally introduced and examined in detail in [1]. The idea of D-BMAP can be traced back

This work is supported by the Inter-University Center for Telecommunications and Informatics (ETIK).

to M. Neuts' work in [6]. Here, we will discuss how to apply the D-BMAP process to model the traffic generated by a TCP connection in a slightly different manner as in [1]. We propose a discrete-time model for TCP. The states of the background process (modulating process) are the states of TCP itself (i.e. Slow Start, Congestion Avoidance, Loss Recovery and Time Out).

Let's consider a general model of discrete MAP:

- The process is time-slotted: the slot length is the average round-trip time ($\overline{RTT}$)
- The probability of transition from state i to state j is denoted by p_{ij} and the transition probability matrix of the modulating Markov-chain is $\mathbf{P} = \{p_{ij}\}$
- When the chain is in state l, the TCP source transmits a random number of packets with probability generating function (p.g.f.) $B_l(z) = \sum_i b_i^{(l)} z^i$, where $b_i^{(l)}$ denotes the probability of i arrivals in a slot when the Markov chain is in state l.

Now, let's define $\mathbf{B}(z)$ matrix as follows:

$$\mathbf{B}(z) = \begin{pmatrix} p_{00}B_0(z) & p_{10}B_0(z) & ... & p_{N0}B_0(z) \\ p_{01}B_1(z) & p_{11}B_1(z) & ... & p_{N1}B_1(z) \\ . & . & ... & . \\ . & . & ... & . \\ p_{0N}B_N(z) & p_{1N}B_N(z) & ... & p_{NN}B_N(z) \end{pmatrix}$$

Let Π denote the stationary (limit) distribution of the modulating Markov chain. Then we can estimate the long term average throughput ($\overline{BW}$) of a TCP connection as follows:

$$\overline{BW} = \Pi(\mathbf{B}'(1))^T \overline{e}[MSS/\overline{RTT}]$$

where $\overline{e}$ is the unit column matrix defined by $\overline{e} = [1, 1, ..., 1]^T$ and $\mathbf{B}'(1) = d\mathbf{B}(z)/dz|_{z=1}$.

B. Numerical analysis

The main purpose of this section is to give the formula for stationary throughput of TCP in *closed form* with some *assumptions* to ease the analysis. In the following discussion, state 0 stands for Slow Start, state 1 stands for Loss Recovery, state 2 stands for Time Out and state 3 stands for Congestion Avoidance, respectively. We analyze TCP Reno in detail. The analysis of other versions of TCP are similar to TCP Reno analysis. In TCP Reno we assume that the duration of Loss Recovery is typically one RTT. It is because at the end of an RTT, the sender can decide to get out of Fast Recovery and continue in Congestion Avoidance or Time Out will occur. In other words, if TCP is in Fast Recovery then the probability of staying in Fast Recovery in the next round-trip time is assumed to be 0. We experience from most of our simulations that Time Out occurred (if any) only in one RTO and no Exponential Back-off. Although our general model can deal with Exponential Back-off, for the sake of simplicity, we deal mainly with Time Out that lasts for only one RTO and as a consequence, TCP jumps to Slow Start with probability 1. Denote p_{TD} the probability of triple ACK loss event and p_{TO} the probability of Time Out event. Let $p_{loss} = p_{TD} + p_{TO}$. In this way, we have the probability that TCP jumps from Loss Recovery to Congestion Avoidance (p_{13}) is $\frac{p_{TD}}{p_{loss}}$ and the probability that TCP jumps from Loss Recovery to Time Out (p_{12}) is $\frac{p_{TD}}{p_{loss}}$. Now let's determine p_{01} and p_{31}. The fact that TCP jumps from Slow Start to Loss Recovery reveals to us that a loss has occurred and TCP was in Slow Start before the loss has been detected. As a result, we have p_{01}=P[loss occurred | from Slow Start]. Similarly, the event that TCP jumps from Congestion Avoidance to Loss Recovery implies that a loss has occurred and TCP was in Congestion Avoidance before the loss has been detected. Consequently, p_{31}=P[loss occurred| from Congestion Avoidance]. The probability of the event that TCP jumps directly from Slow Start to Congestion Avoidance (p_{01}) is P[cwnd_=ssthresh_]. Our simulation shows that, except for the first Slow Start, *no* packet is lost in Slow Start phase. This is understandable because Slow Start can only happen following a Time Out and Slow Start ends when the congestion window equals to the Slow Start threshold and TCP gets to Congestion Avoidance. After Time Out the pipe is already empty and the threshold value is sufficiently small so that it is easily reached by Slow Start phase and state change happens. That's why packet loss is very rarely detected in this period. Consequently, we assume that $p_{01} \approx 0$. From $p_{01}+p_{31} = p_{loss}$ we have $p_{31} \approx p_{loss}$ and consequently $p_{33} \approx (1-p_{loss})$. Now denote $p_{threhold}$=P[cwnd_=ssthresh_] then we have $p_{03} = p_{threshold}$ and consequently $p_{00} = 1 - p_{threshold}$.

To sum up, the TCP characterization matrix for TCP Reno case can be filled as follows:

$$\mathbf{P_{Reno}} = \begin{pmatrix} (1-p_{threshold}) & 0 & 0 & p_{threshold} \\ 0 & 0 & \frac{p_{TO}}{p_{loss}} & \frac{p_{TD}}{p_{loss}} \\ 1 & 0 & 0 & 0 \\ 0 & p_{loss} & 0 & 1-p_{loss} \end{pmatrix}$$

where $p_{loss} = p_{TD} + p_{TO}$.

Let Π be the stationary distribution of the modulating Markov chain, $\Pi = (\pi_0, \pi_1, \pi_2, \pi_3)$. We have $\Pi = \Pi\mathbf{P}$ and Π is a distribution vector, so the following equation system hold:

$$\begin{aligned} \pi_0 &= (1-p_{threshold})\pi_0 + \pi_2 \\ \pi_1 &= \pi_3 p_{loss} \\ \pi_2 &= \pi_1 \frac{p_{TO}}{p_{TD}} \\ \pi_3 &= \pi_0 p_{threshold} + \pi_1 \frac{p_{TD}}{p_{loss}} + \pi_3(1-p_{loss}) \end{aligned}$$

with the constraint $\pi_0 + \pi_1 + \pi_2 + \pi_3 = 1$.
Algebraic computation yields:

$$\begin{aligned} \pi_0 &= \frac{p_{TO}}{(1+p_{TD}+2p_{TO})p_{threshold} + p_{TO}} \\ \pi_1 &= \frac{p_{loss}p_{threshold}}{(1+p_{TD}+2p_{TO})p_{threshold} + p_{TO}} \\ \pi_2 &= \frac{p_{TO}p_{threshold}}{(1+p_{TD}+2p_{TO})p_{threshold} + p_{TO}} \\ \pi_3 &= \frac{p_{threshold}}{(1+p_{TD}+2p_{TO})p_{threshold} + p_{TO}} \end{aligned}$$

Finally, we deal with the case when the slot times at the states are different. Let T_i be the slot time at state i, then we

have the *corrected* stationary distribution $\alpha = (\alpha_0, \alpha_1, ..., \alpha_N)$ with $\alpha_i = \frac{\pi_i T_i}{\sum_j \pi_j T_j}$. Specifically, the time slot in Slow Start, Congestion Avoidance and Fast Recovery is roughly $\overline{RTT}$ whereas the time slot in Time Out (Exponential Back-off) is measured by $\overline{RTO}$. Denote $k = \frac{\overline{RTO}}{\overline{RTT}}$. Notice that in practice k is approximately equal to 4 ($k \approx 4$). Denote $\rho = \frac{1}{1+(k-1)\pi_2}$ be the multiplicative correction term. We have the *corrected* stationary distribution of the modulating Markov-chain can be expressed in *closed form* as follows

$$\pi_0 = \frac{\rho p_{TO}}{(1 + p_{TD} + 2p_{TO})p_{threshold} + p_{TO}}$$
$$\pi_1 = \frac{\rho p_{loss} p_{threshold}}{(1 + p_{TD} + 2p_{TO})p_{threshold} + p_{TO}}$$
$$\pi_2 = \frac{k\rho p_{TO} p_{threshold}}{(1 + p_{TD} + 2p_{TO})p_{threshold} + p_{TO}}$$
$$\pi_3 = \frac{\rho p_{threshold}}{(1 + p_{TD} + 2p_{TO})p_{threshold} + p_{TO}}$$

Notice that if $k > 1$, then $\rho < 1$ and $k\rho > 1$. The fact that $\rho < 1$ implies that the *corrected* fraction of time that TCP stays in Slow Start, Congestion Avoidance and Loss Recovery is *smaller* than before correction. Similarly, the fact that $k\rho > 1$ implies that the *corrected* fraction of time that TCP stays in Time Out (Exponential Back-off) is *longer* than before correction. Since in Time Out events significantly reduce performance, without correction we might have *overestimated* the performance that TCP does actually produce.
Finally, the distributions (as well as the expected values) of the number of packets sent in each time slot for every state are estimated by simulations.

III. A TOOL FOR VALIDATION

The most important part of our tool is the state detection of TCP. To collect the statistics needed for our model, we first need to detect the changes of the states. In this Section, we first describe the basic mechanism then we discuss the difficulties involved with the implementation and our proposed solutions.

A. The mechanism

To begin with, all TCP connections, after hand-shake phase, start with Slow Start phase to estimate the available bandwidth of the network. The TCP sender uses the congestion window variable (as well as the slow start threshold and some other variables) to control the number of packets sending to the network. The idea of our state detection algorithm is based on the dynamics of the congestion window and the slow start threshold process. With the congestion window, we can detect the *changes* of the states. Observe that if TCP is in some state and the congestion window is *increasing* then TCP *stays* in that state. If the congestion window is halved or decreased to 1, then a state change has happened. The slow start threshold provides us the details about the *next* state, if a state change is detected.

B. Problems with state detection

We faced some difficulties when implementing the state detection mechanism of TCP. In this Section, we first state the problems, then we discuss the solution for them.

1) Problem 1: The first difficulty is the detection of the end of Time Out when TCP *backs-off* more than one time in Time Out. As long as TCP is in Time Out, the congestion window is constantly 1 and each time it backs-off, the slow start threshold is halved. In this case, to check how many times TCP backs-off we need to introduce a new variable, namely *ssthreshControl* to follow the halving of the slow start threshold.

2) Problem 2: Another difficulty is the version of TCP that we deal with. The state detection of Reno TCP, NewReno TCP, SACK TCP are more or less the same. The situation is different with Tahoe TCP. In TCP Tahoe, there is *no* Fast Recovery. It *slow starts* after resending the lost packet(s), of any kind. In TCP Tahoe, we have two kinds of Slow Start: Slow Start after Time Out and Slow Start after triple ACKs. So in this case, we need to use the trace file that contains the information about the duplicate acknowledgements.

3) Problem 3: We observe from our simulations that when the congestion window is small (less than 4), the event that the congestion window is decreased to 1 does not necessarily means Time Out. Again, we need the trace file that contains the information about the duplicate acknowledgements to deal with this ambiguity.

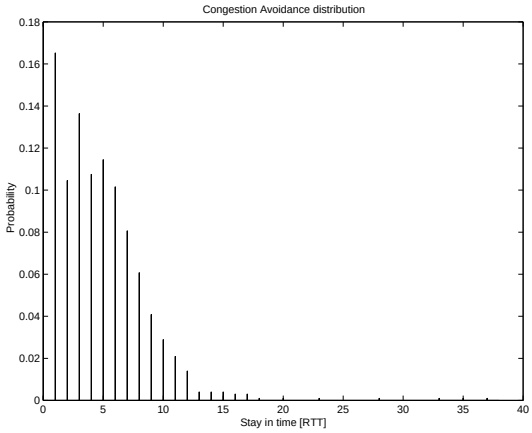
We believe these are the major problems that we had to deal with. There are still many problems relating to the state detection mechanism that we have fixed but, for the sake of simplicity, are not listed here.

IV. RESULTS AND VALIDATION

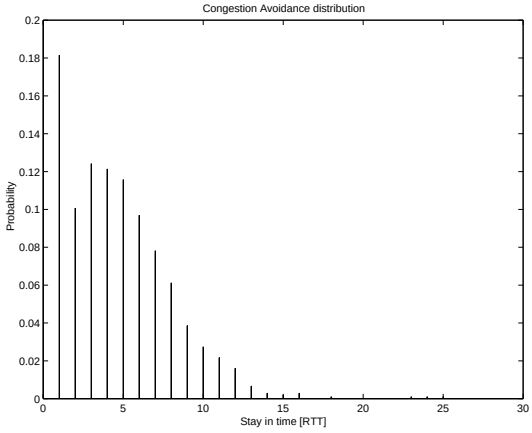
We need to validate two things. Firstly, we consider the validity of the Markov property in our model by examining the sojourn time distribution at different states of TCP. Secondly, we use our model to estimate the steady-state throughput of a TCP connection (with different versions) by validating it against different packet loss probabilities. In order to do so, we first need to carry out experiments to collect the statistics mentioned in the model construction part. The experiments were conducted using ns-2 simulator [7]. The topology that we used was a simple half-dumbbell topology. We added a loss module at the output port of the bottleneck link's router that can deliberately drop packets so that we can control the drop probabilities. In this way, instead of adding more connections to the background traffic we examine a *single* TCP connection. We believe that by deliberately tuning the loss probability, we are able to *emulate* different scenarios of background traffic because the effect of background traffic on a certain TCP connection ultimately results in the packet loss probability of that connection. For the details about our simulations, the packet size was 1000 bytes, the access link was 8 Mb/s with a delay of 0.1 ms, the bottleneck link was 800 Kb/s with a delay of 100 ms and the buffer size was 20 packets. Regarding the queue management schemes at the router, we consider both Drop-Tail and RED mechanisms. With RED-style queue managements, the *gentle* RED was used and the *adaptive* parameter was tuned in.

A. On the sojourn time distribution at the states

1) Congestion Avoidance: First, we examine the sojourn time distributions at Congestion Avoidance state. Since all versions of TCP perform identically in this state, we concentrate on Reno version. However, we examine Reno both with Drop Tail and RED router. As we can see in Figure 1, the sojourn time distribution at Congestion Avoidance is geometrically shaped both in Drop Tail and RED case. It supports the Markovian assumption of our model for this state.



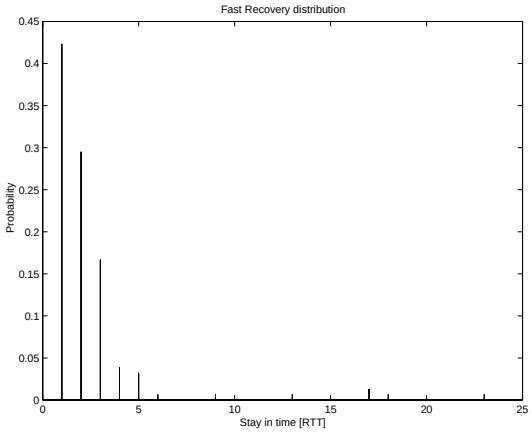
(a) Drop Tail case



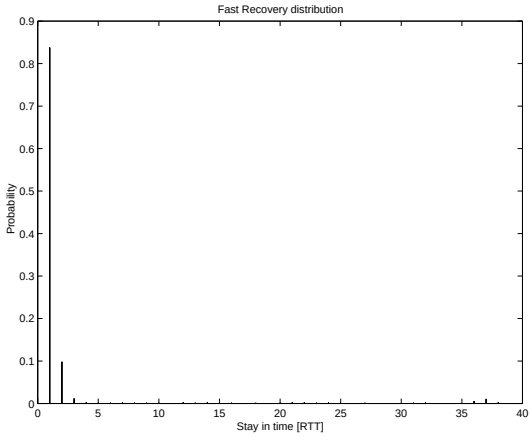
(b) RED case

Fig. 1. Sojourn time distribution: Congestion Avoidance

2) Loss Recovery: In Reno, according to specification, it takes approximately one RTT for TCP to get out of Fast Recovery and TCP enters Congestion Avoidance or Time Out triggers. The situation is different with NewReno and SACK. These versions of TCP is equipped with mechanisms to avoid Time Out in case of multiple losses in a window by *longer* Fast Recovery. With NewReno and SACK versions TCP can stay in Loss Recovery for several round-trip times, depending on the number of losses occurred in a window. So here, we can talk about the distribution of sojourn time. As we can see in Figure 2, the sojourn time distribution at Congestion Avoidance is geometrically shaped both in NewReno TCP and SACK TCP cases. This confirms the Markovian behavior of TCP in this state.



(a) NewReno TCP



(b) SACK TCP

Fig. 2. Sojourn time distribution: Loss Recovery

B. On the stationary performance of TCP

The stationary performance is one of the most important metrics of TCP. This section provides the validation of the stationary throughput of TCP. We compare our numerical results that we achieved from our analysis with the simulation results of ns2 under the same configuration. We go through versions of TCP, version by version.

1) The TCP Reno and Tahoe case: In TCP Reno and Tahoe case, all four states are possible. The probability that Time Out (Exponential Back-off) exists depend on the magnitude of the packet loss probability. If the loss probability is very small (less than 1 percent), then Time Out is rare with TCP Reno, at least with our configuration. If the loss probability is increased, then the probability of more packets dropped in a window of packets increases. Consequently, the probability of Time Out events also increases. We observe from our simulations that

if the packet loss probability gets to 10 percent or higher, Time Out is frequent with Reno. This has severe effect on the performance of TCP Reno. So we validate our model in different packet loss scenarios. We basically examine three types of losses: small (less than 1 percent), average (up to 5 percent), high (higher than 10 percent). Figure 3 shows

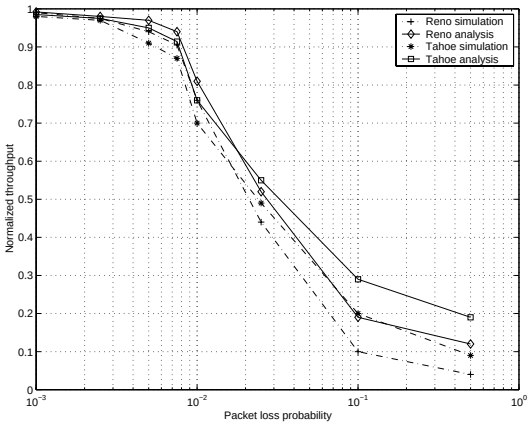


Fig. 3. TCP Reno and Tahoe throughput

the throughput of Reno TCP by simulation and analysis. It presents that our model is in accordance with the simulation results, although we experience some overestimation. However, the overestimation is small enough (less than 1 percent), especially when the packet loss probability is small. The overestimation is already discussed in previous Sections (assumption of exponentially distributed sojourn time as well as the presence of Exponential Back-off where we have given up some details for the sake of simplicity of our model). Regarding the relative performance of TCP Reno and Tahoe, we observe that when packet loss probabilities are small, TCP Reno performs slightly better than TCP Tahoe *both in simulation and analysis*. As the packet loss probability gets higher, the situation changes. TCP Tahoe seems to perform better than TCP Reno, at least in our experiments. This is the reason why in wireless environment, when packet loss probability is high and not necessarily because of congestion, TCP Tahoe performs somewhat better than TCP Reno, as widely suggested in the literature.

2) The TCP NewReno and SACK case: In TCP NewReno and SACK case, we basically have only two states, namely Congestion Avoidance and Loss Recovery. TCP NewReno and SACK perform more or less identically most of the time. The only difference is in the Loss Recovery phase where TCP SACK, by adapting to the *pipe*, is a little bit more aggressive than TCP NewReno. This results in the unfairness between TCP NewReno and TCP SACK when they are in presence.

Figure 4 shows the throughput of NewReno and SACK TCP by simulation and analysis. We observe that SACK performs slightly better than NewReno in most of the cases. This observation supports our view on the unfairness between TCP NewReno and TCP SACK. And surprisingly enough, the model error is *smaller* than in the Reno/Tahoe case. We believe that this is because there was *Time Out* in these cases. At this point we believe that Exponential Back-off is the major cause

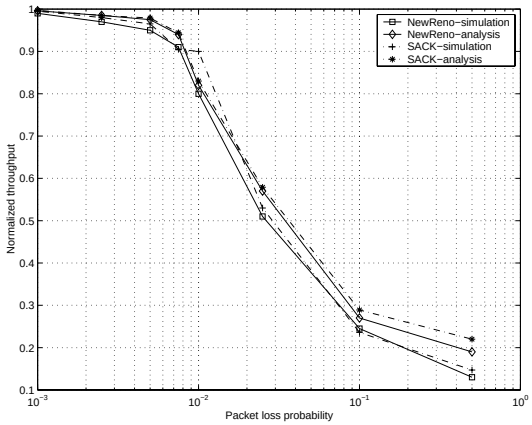


Fig. 4. NewReno and SACK throughput

of error, but more analysis is still needed.

V. Conclusion

We have presented a unified model for some well-known versions of TCP based on the states of TCP itself. We have introduced a new concept, namely the TCP characterization matrix and showed how to use this matrix to model the stationary performance of TCP. We have described a novel technique to automatically detect the states of TCP and developed it into a tool for our state-based analysis. We have applied this tool to collect useful statistics to validate our state-based model of TCP.

Topics of ongoing investigations include the study of the effect of Exponential Back-off on the performance of TCP. We are also working on applying the state-base approach to model and characterize the performance of newly proposed versions of TCP like FAST, Scalable TCP and HighSpeed TCP.

References

[1] C. Blondia, O. Casals, *Statistical multiplexing of VBR sources: A matrix-analytic approach* Performance Evaluation 16, pp. 5-20, 1992.

[2] H. Heffes, D. Lucantoni, *A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performace,* IEEE Journal on Selected Areas in Communications, Vol. 4, No. 6, September 1986, pp. 856-867.

[3] A. Kumar, *Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link,* IEEE/ACM Transactions on Networking, vol. 6, pp. 485–498, August 1998.

[4] Steven Low et al, *Dynamics of TCP/RED and a Scalable Control,* INFOCOMM 2002.

[5] V. Misra, W. Gong and D. Towsley, *A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED.* SIGCOMM 2000.

[6] Marcel Neuts, *Matrix-Geometric Solutions in Stochastic Models - An Algorithmic Approach,* The Johns Hopkins University Press, Baltimore, Maryland, 1981.

[7] Network Simulator (ns2), *www.aciri.org/ns2*

[8] T. Ott, J.H.B. Kemperman, M. Mathis, *The Stationary Behavior of Ideal TCP Congestion Avoidance.* Bell Lab Technical Report, 1996.

[9] J. Padhye et al, *Modeling TCP Reno Throughput: A Simple Model and Its Empirical Validation.* SIGCOMM'98, 1998.

[10] A. Veres, M. Boda, *The Chaotic Nature of TCP Congestion Control,* INFOCOM 2000, Tel Aviv, 2000.