PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 48, NO. 1–2, PP. 71–84 (2004)

MODELLING TCP TRAFFIC: A STATE-BASED APPROACH

Tuan Anh TRINH and 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: {tuan,molnar}@tmit.bme.hu

Received: December 15, 2003

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 propose the use of discrete-time batch Markov 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 *all* 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.

Keywords: TCP modelling, performance analysis.

1. Introduction

TCP modelling can be found at two main levels: packet level and fluid level. One of the motivations for the packet level approach is the possibility of applying existing discrete-time models [4], [9], [10], [12]. The motivation for fluid level model is the possibility of applying existing continuous-time models, [5], [6], [8], [11], [15], to name a few. In both approaches, essential points have been addressed and important, subtle results have been achieved. In [8], T. OTT et al. used stochastic differential equations to model TCP behaviour and first suggested the well-known *square-root* formula. J. PADHYE et al. in [9] extends 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 [15]. However, as TCP modelling is application-sensitive, a general-purpose 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 types 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, it is in the SPEAK state, otherwise it is in the SILENCE state. A natural question arises then: How about TCP? Our state-based model of TCP follows the white-box modelling approach. 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 in each state. Then we consider the jumping probability from one state to another state. From the statistics, we can build a model to estimate TCP throughput. We have developed our technique in a tool called TCP-ASD (TCP Automatic State Detection) that automates state analysis of TCP connections. This paper also reports some results achieved from the tool and simulations of TCP.

The remainder of the paper is organized as follows. In Section 2 we present a state-based model for TCP. A tool for validation is described in Section 3. Section 4 provides validation results of the proposed model. Finally, Section 5 concludes the paper.

2. A D-BMAP Model for TCP Stationary Throughput

2.1. The General Case

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)

First, let's consider a general model of discrete MAP [1]:

- 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:

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

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 $\mathbf{B}'(1) = d\mathbf{B}(z)/dz|_{z=1}$.

Now let us turn our attention to the computational (numerical) aspect of the above formula. First, let's determine Π. Since Π is defined as the stationary distribution of the modulating Markov chain ($\mathbf{P}$), it must satisfy the following equation: $\Pi = \Pi\mathbf{P}$. Consequently, to compute Π, we have to solve the linear system of equations (with constraint that the sum of elements of Π is 1). This is a well-known problem and we can use any of the methods available in the literature. A little more subtle question is how to determine $\mathbf{B}'(1)$. Suppose that we already know the elements of $\mathbf{P}$, so what we still have to compute is $\mathrm{d}\mathbf{B}_l(z)/\mathrm{d}z|_{z=1}$, for $l = \overline{0, N}$. These values are actually the expectations of the distributions of the number of packets sent in a slot, state by state. We estimate these values by the average of the samples from simulations. Before turning to the next part, we note that even though we can estimate the *distributions* of the number of packets sent in one slot (state by state), what we really need is only the *expectations* of the distributions.

We can discuss the D-BMAP model of TCP in a slightly different manner as follows. Consider a discrete-time Markov chain with transition matrix **D**. Suppose that at time k this chain is in state i, $0 \leq i \leq N$. At the next time instant $k+1$, a transition to another or, possibly , the same state takes place and a batch arrival may or may not occur. With probability $(d_0)_{i,j}$, $0 \leq i \leq N$, there is a transition to state j without an arrival, and with probability $(d_n)_{i,j}$, $n \geq 1$, there is a transition to state j with a batch arrival of size n. We have that

$$\sum_{n=0}^{\infty}\sum_{j=0}^{N}(d_n)_{i,j} = 1 \tag{1}$$

Clearly, the matrix $\mathbf{D}_0$ with elements $(d_0)_{i,j}$ governs transitions that correspond to no arrivals, while the matrices $\mathbf{D}_n$ with elements $(d_n)_{i,j}$, $n \geq 1$ govern transitions that correspond to arrivals of batches of size n.

Hence, the process can be defined as a two-dimensional discrete-time Markov process $\{(N(k), J(k)), k \geq 0\}$ on the state space $\{(n, j), n \geq 0, 0 \leq j \leq N\}$ with the transition matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{D}_0 & \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 & \dots \\ \mathbf{0} & \mathbf{D}_0 & \mathbf{D}_1 & \mathbf{D}_2 & \dots \\ 0 & 0 & \mathbf{D}_0 & \mathbf{D}_1 & \dots \\ \vdots & \vdots & \ddots & \ddots & \dots \\ \vdots & \vdots & \ddots & \ddots & \dots \end{pmatrix}$$

The variable $\{N(k), k \geq 0\}$ represents the counting variable and $\{J(k), k \geq 0\}$ the phase variable. With this notation the transition from state (l, i) to state $(l + n, j)$ corresponds to an arrival of size n and a phase change of i to j. The matrix $\mathbf{D} = \sum_{n=0}^{\infty} \mathbf{D}_n$ is the transition matrix of the modulating Markov chain. With this D-BMAP interpretation of TCP we can estimate the long-term average throughput $(\overline{BW})$ of a TCP connection as follows:

$$\overline{BW} = \Pi \sum_{i=0}^{\infty} i\mathbf{D}_i \overline{e}[MSS/\overline{RTT}]$$

Furthermore, the stationary *distribution* can also be computed as follows:

$$Pr[BW = i] = \Pi \mathbf{D}_i \overline{e}[MSS/\overline{RTT}]$$

2.2. *On the Number of States of the TCP Model*

This subsection discusses the problems related to the number of states the model should have. Especially, when we model Time Out (Exponential Back-off), the main question is whether one state is sufficient or not. This question yields to the validity of the Markovian assumption. For example, modelling Time Out with Exponential Back-off by only one state of the Markov chain is usually not sufficient because it is well-known that the sojourn time distribution in this case is not exponential (it is in fact heavy-tailed). But increasing the number of states raises the complexity of the model. So we have to find a compromise here. As long as there is only one RTO of Time Out, we model it by one state. Otherwise, Time Out is modelled by a Markov chain itself (with the number of states being the number of ‘Back-offs’).

2.3. *Numerical Results*

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.

2.3.1. The TCP Reno Case

1. Determination of the characterization matrix

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 if 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 $p_{\text{loss}} = p_{TD} + p_{TO}$. In this way, the probability that TCP jumps from Loss Recovery to Congestion Avoidance (p_{13}) is $\frac{p_{TD}}{p_{\text{loss}}}$ and the probability that TCP jumps from loss Recovery to Time Out (p_{12}) is $\frac{p_{TD}}{p_{\text{loss}}}$. Now let's determine p_{01} and p_{31}. The event that TCP jumps from Slow Start to Loss Recovery implies that a loss has occurred and TCP was in Slow Start before the loss has been detected. Consequently, 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 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 $p_{01} + p_{31} = p_{\text{loss}}$ we have $p_{31} \approx p_{\text{loss}}$ and, consequently, $p_{33} \approx (1 - p_{\text{loss}})$. Now denote p_{threhold}=P[cwnd_=ssthresh_], then we have $p_{03} = p_{\text{threshold}}$ and consequently $p_{00} = 1 - p_{\text{threshold}}$.

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

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

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

2. Analytical analysis

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 holds

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

with the constraint $\pi_0 + \pi_1 + \pi_2 + \pi_3 = 1$.

Algebraic computation yields:

$$\pi_0 = \frac{p_{TO}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

$$\pi_1 = \frac{p_{\text{loss}} p_{\text{threshold}}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

$$\pi_2 = \frac{p_{TO} p_{\text{threshold}}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

$$\pi_3 = \frac{p_{\text{threshold}}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

Finally, we deal with the case when the slot times at the states are different. Let T_i be the slot time in 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 $k = \frac{\overline{RTO}}{\overline{RTT}}$. Note that in practice k is approximately equal to 4 ($k \approx 4$). Let $\rho = \frac{1}{1+(k-1)\pi_2}$ be the multiplicative correction term. The *corrected* stationary distribution of the modulating Markov chain can be expressed in *closed form* as follows:

$$\pi_0 = \frac{\rho p_{TO}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

$$\pi_1 = \frac{\rho p_{\text{loss}} p_{\text{threshold}}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

$$\pi_2 = \frac{k\rho p_{TO}\, p_{\text{threshold}}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

$$\pi_3 = \frac{\rho p_{\text{threshold}}}{(1 + p_{TD} + 2p_{TO})p_{\text{threshold}} + p_{TO}}$$

Note that for $k > 1$, $\rho < 1$ 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 *overestimate* 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 states are estimated by simulations.

2.3.2. The TCP New Reno and TCP SACK Case

As we have mentioned in previous Section, the modelling of Loss Recovery phase of TCP New Reno and TCP SACK is not negligible since TCP, in order to avoid Time Out, stays relatively long in Loss Recovery phase to recover when multiple (consecutive or closely placed) packet losses within a window of data in transition. At this point, it seems that we might need a much more complicated fundamental matrix to model the performance of TCP New Reno and TCP SACK. Interestingly, by contrast, it is not the case. TCP New Reno and SACK, instead of going into Time Out, compensate packet losses in Loss Recovery. Consequently, there is *no* Time Out phase in TCP New Reno and SACK, in most cases. We state ‘in most cases’ because there is an exception when the congestion window is very small (to the extreme case when too many TCP flows competing (sharing) for a very slow bottleneck link). In this case, Time Out could happen even with TCP New Reno and TCP SACK because there is not enough acknowledgment for triple ACK case, and TCP is forced to Time Out. We shall deal with this problem in a separated part, but for the time being, we assume that there is enough acknowledgements flying back and so there is no Time Out with TCP New Reno and SACK. The exclusion of Time Out implies that, apart from the first Slow Start, there is *no* Slow Start phase because Slow Start could *only* happen after a Time Out period. Consequently, we have only two states in consideration, namely Congestion Avoidance and Loss Recovery. Note that our simulations and measurements as well as results from earlier publications (e.g. [9]) suggested that the number of packets lost in a window could be approximated by a geometrical distribution. This implies that the time TCP Reno and SACK stay in Loss Recovery is geometrically distributed (Markov property). Furthermore, the probability of Time Out is approximately zero ($p_{TO} \approx 0$). As a result the probability that TCP jumps from Congestion Avoidance phase to Loss Recovery phase is the probability of loss event (p_{loss}) and the probability that

TCP stays in Congestion Avoidance is $1 - p_{\text{loss}}$. Let p_{recovery} be the probability that TCP stays in Loss Recovery phase and, as a result, $(1 - p_{\text{recovery}})$ is the probability TCP jumps from Loss Recovery to Congestion Avoidance. So the characterization matrix can be filled as follows:

$$\mathbf{P} = \begin{pmatrix} 1 - p_{\text{loss}} & p_{\text{loss}} \\ p_{\text{recovery}} & 1 - p_{\text{recovery}} \end{pmatrix}$$

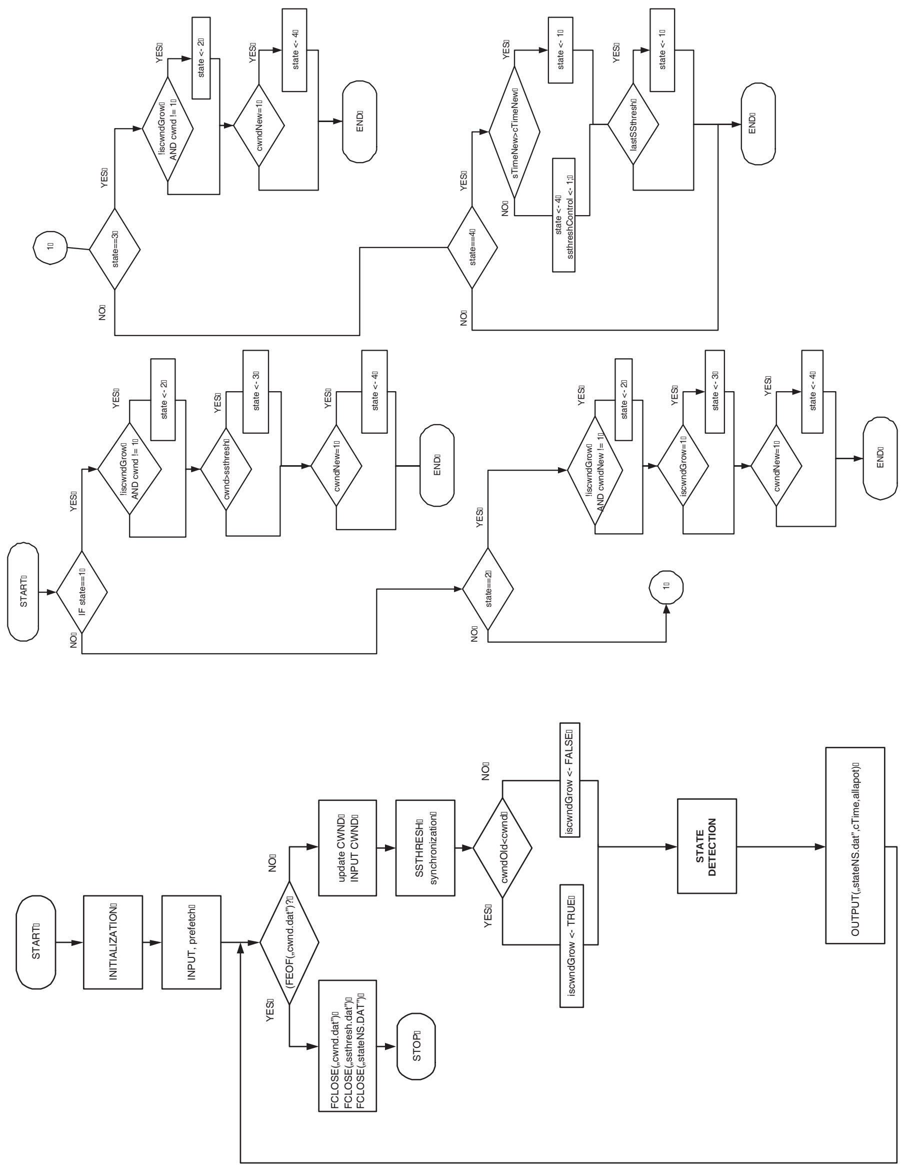
Thus the stationary distribution (π_0, π_1) of this two-state Markov chain is $(\frac{p_{\text{recovery}}}{p_{\text{loss}}+p_{\text{recovery}}}, \frac{p_{\text{loss}}}{p_{\text{loss}}+p_{\text{recovery}}})$, respectively. At this point, it seems that the model of TCP is as simple as the well-known ON/OFF model for voice traffic. In a sense, it's true. However, it should be noted that our model is different from the traditional ON/OFF model in a number of ways. Firstly, in our case, TCP does not have OFF period. Data is sent actively in both states (Congestion Avoidance and Loss Recovery). Secondly, we assume *general* distributions for packets sent at each time slot in contrast to the very limited assumption of Poisson process as in the voice model. The determination of the distributions of the number of packets sent in each time slot (i.e. each round-trip time) in Congestion Avoidance and Loss Recovery is identical to the TCP Reno case.

3. 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.

3.1. The Mechanism

The idea of our 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 a particular 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. *Fig. 1* shows the flow diagrams of the basic steps of our algorithm. In the diagrams, Slow Start is marked with 1 and 2 stands for Lost Recovery, 3 for Congestion Avoidance, 4 for Time Out (Exponential Back-off), respectively.



(b) State detection (in more detail)

(a) Flow diagram: basic steps

Fig. 1. Flow diagrams of state detection

3.2. The Difficulties

We faced some difficulties when implementing the state detection mechanism of TCP. The first difficulty is the detection of the end of Time Out when TCP *backs-off* more than once 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. 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. 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, they are not listed here.

4. Validation

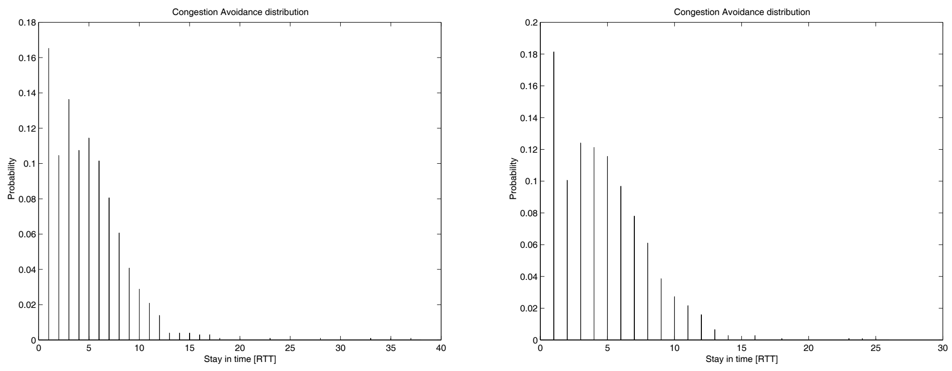
4.1. Simulation Setup

We use simulation (ns2) to validate our model. We added a loss module to ns2 that can deliberately drop packets so that we can control the drop probabilities. So instead of adding more connections to the background traffic, we examine a *single* TCP Reno 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. The parameters of the simulations used to validate are as follows. The topology was a simple half-dumbell topology. 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. We investigate the performance of TCP both with Drop-Tail and RED router mechanisms.

4.2. On the Sojourn Time Distribution at States

4.2.1. 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



(a) Drop Tail case (b) RED case

Fig. 2. Sojourn time distribution: Congestion Avoidance

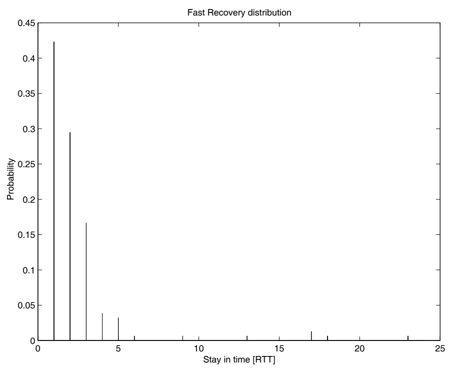
we can see in *Fig. 2*, the sojourn time distribution at Congestion Avoidance is geometrically shaped both in Drop Tail and RED case. It supports the Markovian assumption of our model for this state.

4.2.2. Loss Recovery

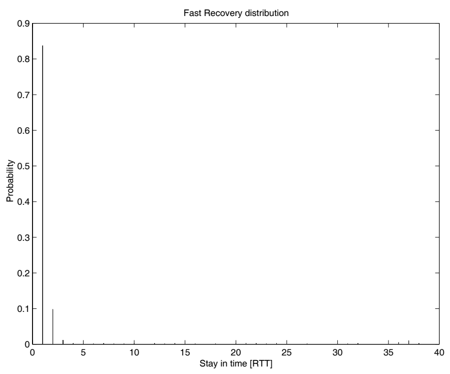
In Reno, according to specification, it takes approximately one RTT for TCP to get out of Fast Recovery and enter Congestion Avoidance or Time Out triggers. The situation is different with NewReno and SACK. These versions of TCP are equipped with mechanisms to avoid Time Out in case of multiple losses 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 *Fig. 3*, the sojourn time distribution at Congestion Avoidance is geometrically shaped both in NewReno TCP and SACK TCP cases. This confirms the Markovian behaviour of TCP in this state.

4.3. On the Stationary Throughput

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 achieved from our analysis with the simulation results of ns2 under the same configuration. We go through versions of TCP, version by version.



(a) NewReno TCP



(b) SACK TCP

Fig. 3. Sojourn time distribution: Loss Recovery

First we validate our model for TCP Reno case, then we validate our model for TCP NewReno and SACK case.

4.3.1. The TCP Reno Case

In TCP Reno case, all four states are possible. The probability that Time Out (Exponential Back-off) exists depends on the magnitude of the packet loss probability. If the loss probability is very small (less than 1 percent), then Time Out is rare even 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 rises. Consequently, the probability of Time Out event also grows. We observe from our simulations that if the packet loss probability gets 10 percent or higher, Time Out is frequent. This has severe effects on the performance of TCP. 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). *Fig. 4* shows 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).

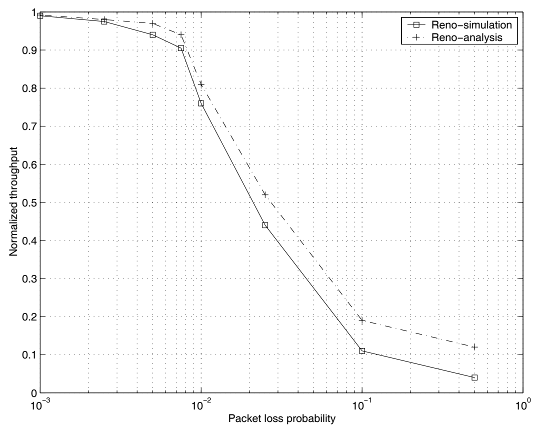


Fig. 4. Reno throughput

4.3.2. *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.

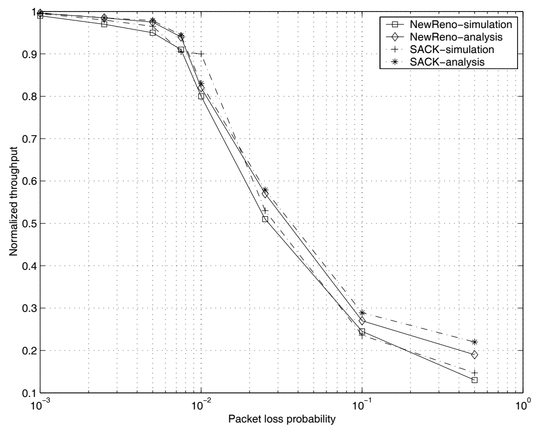


Fig. 5. NewReno and SACK throughput

Fig. 5 shows the throughput of NewReno and SACK TCP by simulation and analysis. We observe that SACK performs slightly better than NewReno in most cases. This observation supports our view on the unfairness between TCP NewReno and TCP SACK. Surprisingly enough, the model error is *smaller* than in the Reno 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 of error, but more analysis is still needed.

5. Conclusion

We have presented a unified model for all 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 for our state-based model of TCP.

Topics of ongoing investigations include the study of the effect of Exponential Back-off on 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] BLONDIA, C. – CASALS, O., Statistical Multiplexing of VBR Sources: A Matrix-Analytic Approach, *Performance Evaluation*, **16** (1992), pp. 5–20.
[2] CARDWELL, N. – SAVAGE, S. – ANDERSON, T., *Modeling TCP Latency*, INFOCOMM 2000, Tel Aviv, 2000.
[3] CASETTI, C. – MEO, M., A New Approach to Model the Stationary Behaviour of TCP Connections, In. *Proc. of IEEE INFOCOM*, March 2000, pp. 367–375.
[4] KUMAR, A., Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link, *IEEE/ACM Transactions on Networking*, 1998.
[5] LOW, S. et al., *Dynamics of TCP/RED and a Scalable Control*, INFOCOM 2002.
[6] MISRA, V. – GONG, W. – TOWSLEY, D., *A Fluid-Based Analysis of a Network of AQM Routers Supporting TCPFlows with an Application to RED*. SIGCOMM 2000.
[7] Network Simulator (ns2), `www.aciri.org/ns2`
[8] OTT, T. – KEMPERMAN, J. H. B. – MATHIS, M., *The Stationary Behavior of Ideal TCP Congestion Avoidance*, Bell Lab Technical Report, 1996.
[9] PADHYE, J. et al., *Modeling TCP Reno Throughput: A Simple Model and Its Empirical Validation*. SIGCOMM'98, 1998.
[10] TINNAKORNSRISUPHAP, P. – MAKOWSKI, A., *Limit Behavior of ECN/RED Gateways Under a Large Number of TCP Flows*, INFOCOMM 2003.
[11] SCHWEFEL, *Behavior of TCP-Like Elastic Traffic at a Buffered Bottleneck Router*, INFOCOMM 2001.
[12] TRINH, T. A. – MOLNÁR, S., A State-Based Model of TCP, In: *Proceedings of the Neumann János Conference*, October 2, Budapest, Hungary.
[13] TRINH, T. A. – MIHÁLY, K. – MOLNÁR, S., A State-Based Analysis of TCP, *IEEE/IFIP Workshop on the Next Generation Networks*, 8–10 September 2003, Balatonfüred, Hungary.
[14] TRINH, T. A. et al., On Some Metrics of TCP, *The 25th International Conference on Local Area Networks* (*LCN'00*), November 4–7, 2000, Tampa, Florida, USA.
[15] VERES, A. – BODA, M., *The Chaotic Nature of TCP Congestion Control*, INFOCOM 2000, Tel Aviv, 2000.