Analysis of TCP Transient Behavior and Its Effect on File Transfer Latency

Navid Ehsan, Mingyan Liu
Electrical Engineering and Computer Science Department,
University of Michigan, Ann Arbor, MI 48109-2122
email: {nehsan, mingyan}@eecs.umich.edu

***Abstract*— In this paper we present a Markov Chain model for TCP congestion avoidance phase. With this model we are able to analyze congestion window behavior as a discrete-time stochastic process and distinguish between window transient period and steady state. Using this result we are able to obtain more accurate estimate of TCP latency over lossy links compared to existing models. We then simplify the proposed model and show that the transient period evolves with an exponential rate. Our results are validated using NS2 simulation and show significant improvement in latency estimate for a wide range of file sizes.**

I. Introduction

Transmission Control Protocol (TCP), as the most widely used reliable transport layer protocol, constitutes the majority of the current Internet traffic. Its behavior over lossy links has been the subject of extensive study in recent years. It has been shown by multiple independent studies and via different approaches that the steady-state throughput of TCP (or throughput of an infinite source) in an environment with random losses is inversely proportional to both the round-trip time (RTT) of the connection and the square-root of the loss rate, see for example, [1], [2], [3], [4]. Furthermore, [2] studied asymmetric connections with finite buffers and determines throughput as a function of buffering, round trip times and normalized asymmetry. [3] modeled TCP's congestion window size as a stochastic process and presents analysis for derivation of its steady-state distribution. In [4] a more accurate model is developed to determine the steady-state throughput of Reno TCP by explicitly modeling timeouts.

All the above work is steady-state analysis and implies long-lasting connections. [5] and [6] introduced methods to estimate the latency for finite or small file transfers. Both methods are based on the assumption that the connection enters steady state immediately after the first loss it experiences. In particular, the latency analysis is divided into two parts. In the first part the congestion window grows exponentially during slow-start until the first expected loss occurs. The number of packets successfully sent and the amount of time spent during this stage can be calculated explicitly. In the second part, the connection is assumed to be in steady state and thus the throughput can be estimated using existing results, e.g., from [3], [4]. The time spent in this stage is then the amount of remaining packets in the connection divided by the steady-state throughput. The total latency of the connection is obtained by taking the sum of time spent in each of the two parts. The limitation of this approach is illustrated in Figure 1. We compared the actual average congestion window size (averaged over 100 independent simulations using different random seeds), over the duration of the connection, to that obtained via the above analysis assuming that the connection reaches steady state immediately following the first loss. The main observation is that the congestion window will go through a *transient* period (decay in this example) before reaching the steady-state throughput. If the file is not large enough then the congestion window may never reach steady state and therefore the slow-start and this transient period will dominate the window evolution. The above assumption can result in fairly accurate estimates if the file is very small (i.e., likely to finish transfer before the expected first loss occurs) or very large (i.e., the connection reaches steady-state and remains in steady-state for sufficiently long so that the effect of this transient period is diluted). However, as will be shown later, neglecting this transient part leads to large error in estimating the latency for a wide range of intermediate file sizes.

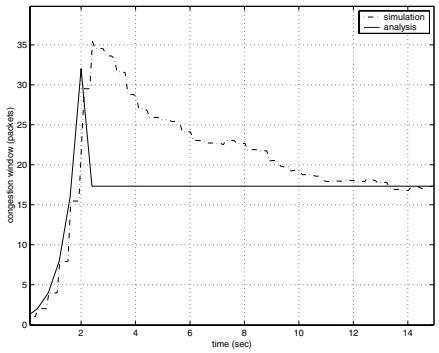

Fig. 1. Congestion window evolution. Slow-start threshold(W_{sst})=40,Bandwidth=5Mb/sec, random loss probability(q)=0.005, RTT=400msec

This has motivated us to find a more accurate latency estimate that accounts for this transient period. In this paper we present a Markov Chain model that gives accurate estimates on file transfer latency regardless of the file size. In addition, our model captures the complete characteristics of the window evolution as a function of time, specifically its probability mass

function. This allows for immediate extension to analysis that potentially involve probability distribution or higher moments of the window size.

The rest of the paper is organized as follows. In the next section we present our model and underlying assumptions. In section 3 we briefly go over the slow-start analysis, since this part is widely studied in the literature. In section 4 we analyze congestion avoidance phase of TCP using a Markov Chain model and solve the chain according to its initial conditions. In section 5 the steady state of this Markov Chain is studied and the result is compared to the existing steady-state analysis. In section 6 we simplify our model and derive an exponential evolution for the window size in steady state. In section 7 we verify our model and compare it with existing model using NS2. Section 8 concludes the paper.

II. Network Model and Assumptions

In Section 4 we will model the dynamics of the congestion window (assuming TCP is in its congestion avoidance stage) using a finite state Markov Chain $\{cwnd_n\}$, where $cwnd_n$ denotes the congestion window size at step n, $cwnd_n \in \{1, 2, \cdots, W_{max}\}$, and n represents the n^{th} RTT. In doing so we make the following assumptions: (i) The variance in RTT is small compared to the observed period or the connection duration. Therefore RTT is approximated as being constant and each RTT is considered a "step" in this model; (ii) Losses in the network are random and each packet experience independent losses with a fixed loss probability q. This assumption is not unreasonable considering the wide deployment of Random Early Discard (RED) routers but it is different from the observations from tail-drop queues. (iii) Any change in the congestion window size can be made at the edge of each step. In reality when a packet loss is detected, action is taken immediately, but in our model TCP keeps its window size constant within each RTT. If a packet is lost in a window, the server will be notified before the next window starts, so each loss in the window will affect the next window size. (iv) The congestion window is halved at most once in each RTT. This assumption makes our model closest to NewReno [7] or TCP Sack [8] rather than TCP Reno [9]. Note that this assumption can be easily relaxed and the same modeling framework can be applied to TCP Reno. (v) Whenever the congestion window is halved, it is rounded to the closest integer below the halved value. This is obviously an approximation in exchange for a finite state space and less computation. (vi) The return path for ACK is lossless. Since acknowledgments are cumulative, this is not an unrealistic assumption. (vii) The send rate of the TCP source is constrained by the congestion window size rather than by the network link capacity, therefore can be approximated using $rate = \frac{cwnd}{round-trip-time}$. (viii) For simplicity reasons we will also ignore timeouts during congestion avoidance. It has been shown in [4] that not considering timeouts can lead to large error in TCP throughput estimation especially when loss probability is high. However, since our stochastic model is mainly for the transient stage, leaving out timeouts has limited effect. Existing model with timeout can be used for steady-state analysis.

In the next section we will deviate briefly to discuss the slow-start window evolution and will resume the discussion on congestion avoidance in Section 4.

III. Slow-Start Window Evolution

Denoting by W_{sst} the slow-start threshold and W_0 the initial window size, we first analyze what happens before the first loss occurs or W_{sst} is reached. TCP starts by setting initial window size to W_0 and increases it by one for every ACK from the receiver. If TCP enters the congestion avoidance phase at the n_0^{th} RTT, then as shown in [5] we have:

$$cwnd_i = W_0 \cdot r^i \qquad where\, i = 0, 1, 2, ..., n_0 - 1 \qquad (1)$$

where $r = 1 + \frac{1}{b}$ and b indicates the number of packets received before an ACK is sent ($b > 1$ means delayed ACK). $n_0 - 1$ is the time when either the first loss is detected or W_{sst} is reached (this is the last RTT in slow-start).

Claim 1: Suppose the number of packets sent before the first loss is detected or W_{sst} is reached is d and let the file size to be M packets. We have:

$$E[d] = \sum_{i=1}^{k} (i-1) \cdot q \cdot (1-q)^{i-1} + k \cdot (1-q)^k \qquad (2)$$

where $k = \min((W_{sst} - W_0)b, M)$.

Proof: Suppose $i-1$ packets are sent successfully and the i^{th} packet is lost. The probability of this event is $q \cdot (1-q)^{i-1}$. If none of the packets are lost by the time $cwnd = W_{sst}$, then the congestion avoidance phase starts. Since TCP increases its window size for each receiving ACK, the maximum number of packets that could be transmitted before entering congestion avoidance is $(W_{sst} - W_0)b$ if there is enough packets to be sent, otherwise the maximum number of packets will be equal to M. Therefore the maximum number of packets that can be transmitted during slow-start is $k = \min((W_{sst} - W_0)b, M)$. If there is any loss before that, congestion avoidance starts right after the loss. If there is no loss by the k^{th} packet congestion avoidance phase starts. The probability of the latter is $(1-q)^k$. ∎

The following result is shown in [5]. For self-sufficiency we state it here with a very simple proof. Note that in our notation, the last step to transmit in slow-start is $n_0 - 1$.

Proposition 1: Using the above claim for deriving $E[d]$, we can calculate $E[n_0]$ as follows.

$$E[n_0] = 1 + \lfloor \log_r \frac{E[d]}{b \cdot W_0} + 1 \rfloor \qquad (3)$$

proof: The total number of packets transmitted up to step $(E[n_0] - 1)$ is:

$$E[d] = \sum_{n=0}^{E[n_0]-1} W_0 \cdot r^n = W_0 \cdot b \cdot (r^{E[n_0]-1} - 1)$$

Solving for $E[n_0]$, (3) is obtained. ∎

IV. Transient Behavior of the TCP Window During Congestion Avoidance

In this section we model the the evolution of congestion window as a finite state Markov Chain based on assumptions discussed in Section 2. Congestion avoidance phase has previously been modeled as a Markov chain (see for example [3]), to derive the stationary behavior of TCP. To the best of our knowledge, our study in this paper is the first on analyzing the transient phase.

Following our assumptions, if there is no error in the n^{th} window/RTT of size $cwnd_n = i$ and assuming no delayed ACKs are used, we have $cwnd_{n+1} = i+1$. Otherwise if there is one or more errors TCP will halve its window in the next RTT: $cwnd_{n+1} = \lfloor \frac{i}{2} \rfloor$. This results in a finite state space $S = \{1, 2, ..., W_{max}\}$ where W_{max} is the maximum window size. Figure 2 illustrates the transition between states as an example when the maximum window is 6. $P(i)$ denotes the probability of having error (one or more) in a window of size i, thus $P(i) = 1 - (1-q)^i$ where q is the per-packet loss probability. The following state-dependent Markov Chain can thus be constructed:

$$p_{ij} = \text{Prob}\{cwnd_{n+1} = j \mid cwnd_n = i\} = \begin{cases} 1 - P(i) & \text{if } j = i+1, 1 \le i < W_{max} \text{ or} \\ & \text{if } j = i = W_{max} \\ P(i) & \text{if } j = \lfloor \frac{i}{2} \rfloor, 1 < i \le W_{max} \text{ or} \\ & \text{if } j = i = 1 \\ 0 & \text{else} \end{cases}$$

This model can easily be extended to account for delayed acknowledgment. For example if one ACK is sent for every two TCP segments ($b = 2$), we have $cwnd_{n+1} = cwnd_n + \frac{1}{2}$ if there is no error in the n^{th} window and $cwnd_{n+1} = \frac{cwnd_n}{2}$ when there is error. In order to have finite number of states we approximate $cwnd_{n+1}$ using $\frac{X_n - \frac{1}{2}}{2}$ whenever X_n is not an integer and there is a loss. This results in a state space $S = \{1, 1.5, 2, 2.5, 3, ..., W_{max}\}$. Similar transition probabilities can then be derived. The rest of analysis presented here is for the case when delayed ACKs are not used. Extending them to the case where delayed ACKs are used is straightforward. We will point out major differences in the analysis where necessary.

We can now define a $W_{max} \times W_{max}$ matrix P, the transition probability matrix, whose elements p_{ij} are defined above. Denoting the probability that $cwnd_n = i$ as $w_n(i)$ for all i, and $\mathbf{w}_n$ to be the column vector of all such probabilities at time n, we have:

$$\mathbf{w}_{n+1}{}^T = \mathbf{w}_n{}^T \cdot P \tag{4}$$

This gives us complete description of the congestion window dynamics during the congestion avoidance stage. In order to calculate window distribution we need the initial probability distribution $\mathbf{w}_{n_0}$, where n_0 is the time when congestion avoidance phase starts.

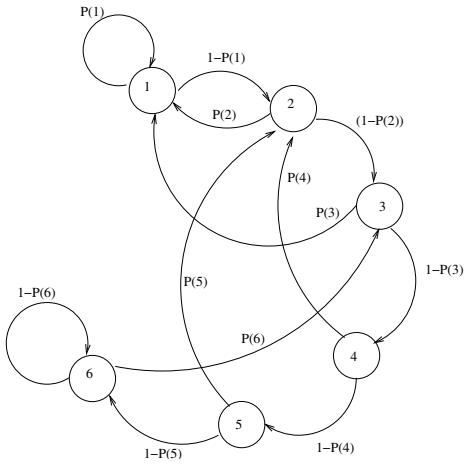

Fig. 2. Markov Chain of window evolution, no delayed ACK, $W_{max} = 6$.

Claim 2: If there is enough packets in the file such that slow start is completed by reaching W_{sst}, then

$$w_{n_0}(i) = \begin{cases} q(1-q)^{b\cdot(2i-W_0)} \cdot (1 + (1-q)^b) & \text{if } W_0 \le 2i < W_{sst} \\ (1-q)^{b\cdot(i-W_0)} & \text{if } i = W_{sst} \\ 0 & \text{else} \end{cases}$$

where $i \in \{1, 2, \cdots, W_{max}\}$ when $b = 1$ and when $b = 2$, $i \in \{1, 1.5, \cdots, W_{max}\}$.

Proof: Suppose congestion avoidance phase starts from $cwnd_{n_0} = i$ by detecting a loss in slow-start. It means that at the time when loss was detected the congestion window size was $2i$ or $2i+1$. On the other hand, in slow-start window is increased by one upon receiving each ACK, i.e. successful transmission of a packet. Considering that the initial window is equal to W_0 there must have been either $(2i - W_0)$ or $(2i - W_0 + 1)$ acknowledgements received. If the sender receives one ACK for each b successful transmissions, then there must have been $b.(2i - W_0)$ or $b.(2i - W_0 + 1)$ successful transmissions prior to the detection of the first loss and by using the independence of losses we can easily verify that the probability of this event is $q.(1-q)^{b.(2i-W_0)} + q.(1-q)^{b.(2i-W_0+1)}$. If there is no loss in the first $W_{sst} - W_0$ packets then slow-start ends as the threshold is reached and the probability of this event is $(1-q)^{b.(i-W_0)}$. Thus verifying the claim. ∎

Claim 3: If there is not enough packets in the file to reach W_{sst} (i.e., $W_{sst} > W_0 + \frac{M}{b}$), then

$$w_{n_0}(i) = \begin{cases} q(1-q)^{b\cdot(2i-W_o)} \cdot (1 + (1-q)^b) & \text{if } W_0 \le 2i < W_0 + \frac{M}{b} \\ (1-q)^M & \text{if } i = W_0 + \frac{M}{b} \\ 0 & \text{else} \end{cases}$$

where $i \in \{1, 2, \cdots, W_{max}\}$ when $b = 1$ and when $b = 2$, $i \in \{1, 1.5, \cdots, W_{max}\}$.

Proof of this claim is the same as the the previous one and is thus not repeated.

Using the above initial conditions, (4) can be solved and the average window size in the n^{th} RTT is

$$w_{ave}(n) = \sum_{i=1}^{W_{max}} i \cdot w_n(i), \quad n > n_0 \tag{5}$$

We now have a complete characterization of TCP window size evolution. To summarize, we have the exponential growth for the congestion window from the beginning until n_0^{th} window when congestion avoidance starts. The initial condition for this instant is given by claim 1 and 2. From this point on, the state of the system is given by (4). To calculate the average window size in each RTT, (5) is used. In order to translate from step n to real time axis, we will need to multiply n by RTT.

V. Steady-State analysis

In this section we study the convergence of the Markov Chain presented in the previous section.

Due to space limit, complete derivation of steady-state probabilities is omitted, and can be found in [10]. Here we use the results to calculate the average window size in steady sate. Figure 3 compares the steady state obtained using the above procedure with the steady-state analysis derived using the square-root formula without considering timeouts. It is shown [3], [4] that TCP throughput is proportional to $\frac{1}{RTT}\frac{a}{\sqrt{q}}$ in packets per unit time. a equals $\sqrt{3/2}$ in the special case of TCP Reno [4].

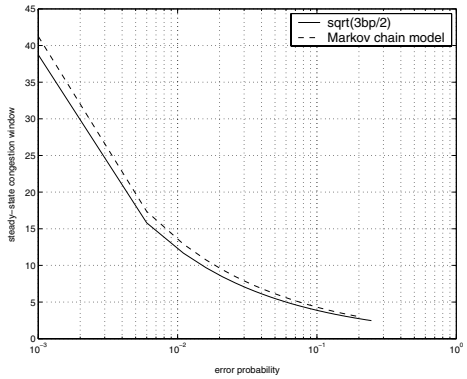

Fig. 3. Steady-state congestion window size vs. error probability, $W_{sst} = 40, W_{max} = 80$

VI. Simplifying the Proposed Model

Following analysis in the previous sections, the TCP window evolution can be divided into three distinct regions: (1) Slow-start where the window grows exponentially; (2) Congestion avoidance transient phase; and (3) Steady state. These three regions are illustrated in Figure 4.

Analysis of the first and third regions results in relatively simple closed form solutions, but for analyzing the transient part, we have to use the Markov Chain model which involves a large amount of calculation and does not provide a lot of intuition. In this section we simplify our transient analysis calculation and present it in a more illustrative way.

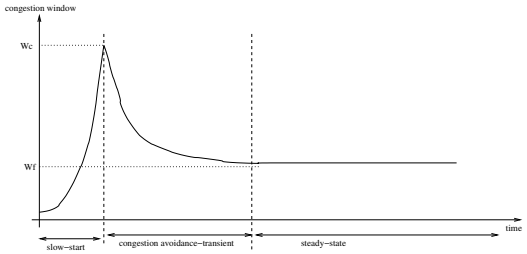

Fig. 4. Summary of the congestion window evolution of TCP

We use the *Perron-Frobenius Theorem* from [11] to simplify our model. Based on this theorem, the system described by the state transition probability matrix P converges to steady state with rate $|\lambda_2^n|$, where λ_2 is the second largest eigen value (in absolute value) (SLEV) of P. This implies that the TCP congestion window size converges to steady state with rate $|\lambda_2^n|$. Since the average window size is averaging over all possible window size values, this average also converges to its steady state with the rate $|\lambda_2^n|$. $|\lambda_2|$ is calculated from P. Compared to executing (4) multiple times this results in significant savings in computation. Moreover, the simplified model is mathematically more tractable.

To summarize, we have the following model for TCP window size evolution.

$$w_{avg}(t) = \begin{cases} r\frac{t}{RTT} & \text{if } t \le t_0 \\ W_f + (W_c - W_f)|\lambda_2|^{\frac{t}{RTT}} & \text{if } t > t_0 \end{cases} \tag{6}$$

where W_c is the expected window size when congestion avoidance starts, and is obtained using $W_c = \sum_{i=1}^{W_{max}} i \cdot w_{n_0}(i)$. t_0 is the time when $w_{avg}(t) = W_c$ (when congestion avoidance starts), i.e. $t_0 = RTT \log_r W_c . w_{n_0}$ is given by either claim1 or claim 2. W_f is the steady-state window size calculated either from the state transition probability matrix P (see [10]) or by using the formula derived in [4]. Here timeouts can also be considered. Note that results in [4] are based on a different loss assumption where following the first loss in a window, all remaining packets in the same window are also assumed to be lost. This is somewhat different from the independent loss probability we have assumed in this paper. However when loss rate is low the assumptions become close.

VII. Validating the results

In this section we use the network simulator NS2 to validate our results. All simulation results are the average over 100 independent simulation runs using different random seeds. We used TCP SACK for our simulation. The size of each TCP packet is set at 1Kbytes and $W_{max} = 2W_{sst}$.

Figure 5(a) compares the result from our analysis to the result obtained from simulation. We assume that the sender always has packets to send and we compare the average number of packets sent at each time with our analysis. It is compared to the results obtained without considering the transient part (this is essentially the same model obtained in [5]). This figure shows that both results (considering transient part or not) are "diverging" from the result from simulation.

This is due to the fact that when comparing the number of transmitted bytes in time, errors are cumulative. However if we compare the resulting throughput (Figure 6), the error stays within a small range.

Figure 5(b) compares congestion window size evolution for the same scenario (compared to Figure 1). It can be seen that although our window size analysis perfectly match that from simulation, file transfer latency is underestimated (still much more accurate than not considering the transient part). This is mainly because timeouts are not considered in the transient part of our model.

From the example shown in Figure 5(a) (where W_{sst} is set to 40), we can see that for files smaller than 103 Kbytes or bigger than 800 Kbytes considering the transient phase does not provide special benefit. This is because in the former the transfer is most likely to complete within the slow-start period so the transient part has very small effect on average, whereas in the latter the effect of the transient part is diluted by the extended steady state phase. Figure 5(b) shows that the amount of time that the window size is in transient phase is roughly 10 seconds. Note that these quantities are case specific. For example, if we increase W_{sst}, the slow-start phase will be longer, so the transient part will only be effective starting from some file-size greater than the 103 Kbytes in this example. If the error probability is increased, the slope of the transient part will increase and the system will go to steady state faster.

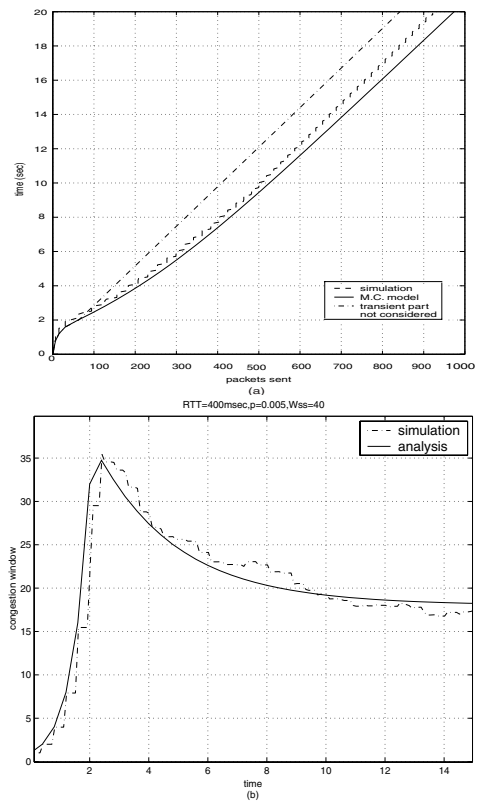

Fig. 5. One way delay = 200ms, $q = 0.005$, $W_{sst} = 40, C = 5Mb/sec$ (a)Delay (b)congestion window size.

Figure 6 considers the same scenario and compares the throughput obtained from analysis to the simulation results. Throughput is simply defined as file size divided by its latency.

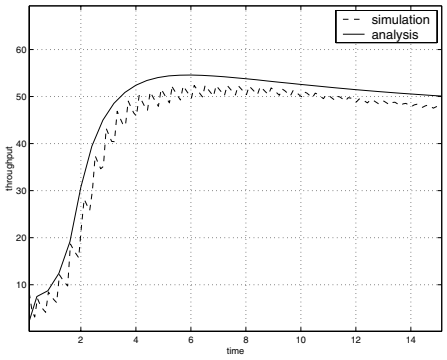

Fig. 6. Throughput comparison - One way delay = 200ms, $q = 0.005$, $W_{sst} = 40, C = 5Mb/sec$.

It is worth pointing out that the window size does not necessarily need to be decreasing in order for our analysis to hold. In fact Figure 4 can be modified for the case where $W_f > W_c$. Figure 7 illustrates one such case.

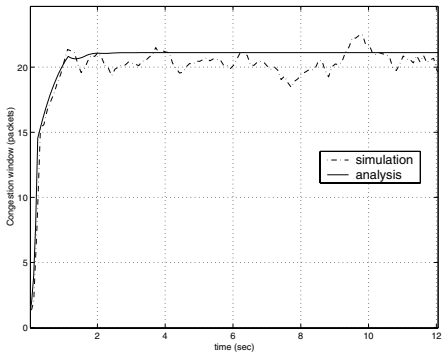

Fig. 7. Congestion window size - One way delay = 30ms, $q = 0.003$, $W_{sst} = 15, C = 5Mb/sec$.

Results above were derived using the exact Markov Chain analysis. Now we show the accuracy of our approximation. W_f is calculated using the square-root approximation given earlier. Figure 8 illustrates the accuracy of this approximation.

VIII. Conclusion

In this paper we studied TCP's congestion window evolution over time. We presented a Markov Chain for analyzing the congestion window behavior following the first loss and before entering steady state. We showed that window size evolves with an exponential rate based on the window probability distribution. This transient part affects the estimate of file transfer latency for a wide range of file sizes. The effect of this transient part becomes more significant as the SLEV

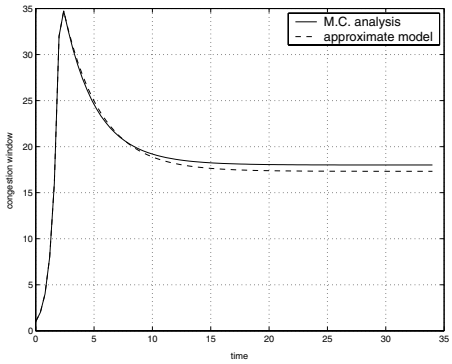

Fig. 8. Comparison between the approximation on the congestion window size and the exact analysis from the Markov Chain. One way delay = 200ms, $q = 0.005$, $W_{sst} = 40, C = 5Mb/sec$.

increases. Using our model we showed very accurate estimate in congestion window size and latency.

REFERENCES

[1] T.V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE Trans. Networking, vol.5, no.3, pp. 336-350*, 1997.

[2] T. V. Lakshman, U. Madhow, and B. Suter, "TCP Performance with Random Loss and Bidirectional Congestion," *IEEE Trans. Networking*, vol. 8, no. 5, pp. 541–555, 2000.

[3] Teunis J. Ott, J.H.B. Kemperman, and Matt Mathis, "The Stationary Behavior of Ideal TCP Congestion Avoidance," *ftp://ftp.bellcore.com/pub/tjo/TCPWindow.ps*, 1996.

[4] J.Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Transactions on Networking, Vol.8, No.2*, 2000.

[5] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," *IEEE INFOCOM*, 2000.

[6] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK," *To appear in the Proceedings of IEEE GLOBECOM, San Antonio, TX*, 2001.

[7] S. Floyd and T. Henderson, "The newreno modification to tcp's fast recovery algorithm," *IETF RFC 2582*, 1999.

[8] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," 1996.

[9] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *IETF RFC 2001*, 1997.

[10] N. Ehsan and M. Liu, "Analysis of TCP Transient Behavior and Its Effect on File Transfer Latency," Tech. Rep., Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, 2001.

[11] P. Bremaud, *Markov Chains, Gibbes fields, Monte Carlo simulations and Queues*, Springer, 1991.