TCP Westwood: Analytic Model and Performance Evaluation

Andrea Zanella, Gregorio Procissi, Mario Gerla, M. Y. "Medy" Sanadidi

Computer Science Department, University of California, Los Angeles (UCLA)

Abstract-**We present a performance model of TCP Westwood (TCPW), a new TCP protocol with a sender-side modification of the window congestion control scheme. TCP Westwood controls the window using end-to-end connection bandwidth share estimation. The key innovative idea is to continuously estimate, at the TCP sender, the packet rate of the connection by monitoring the ACK reception rate. The estimated connection rate is then used to compute congestion window and slow start threshold settings after a congestion episode. Resetting the window to match available bandwidth makes TCPW more robust to sporadic losses due to wireless channel problems. An analytic model using Markov Chain techniques is developed in this paper, and then used to assess the performance improvements obtained using TCPW. The model takes into account the estimation and filtering method used in TCPW, as well as the following system parameters, bottleneck link bandwidth, buffer space at the bottleneck router, end-to-end propagation time, and error rate. The model reveals substantial TCPW gains over Reno whenever losses due to link or other errors are taken into consideration. The analytic model accuracy is confirmed by comparing to simulation results. The results match very well for the range of parameter values of interest.**

I. INTRODUCTION

TCP provides end-to-end, reliable, congestion controlled connections over the Internet [1]. The congestion control method includes two phases: slow-start and congestion avoidance [2]. Nowadays, TCP is called upon to provide reliable and efficient data transfer over communication paths with ever increasing bandwidth-delay product, and over a variety of link technologies including wired (i.e., cable and fiber optic), ground radio, and satellite links. More losses due to link failures, independent as well as correlated, are expected over these wireless links. The new high speed wired/wireless environment is expanding the domain, for which TCP was initially developed, tested and tuned. As a consequence, active research is in progress to extend the domain of effective TCP operability [3][4][5][6][7].

TCP Westwood (or TCPW for short) [8] design adheres to the end-to-end transparency guidelines set forth in [9]. Namely, a simple modification of the TCP source protocol stack allows the source to estimate the available bandwidth and to use the bandwidth estimation to recover faster, thus achieving higher throughput. TCP Westwood exploits two basic concepts: the end-to-end estimation of the available bandwidth, and the use of such estimate to set the slow start threshold and the congestion window. It is worth underscoring that TCPW does not require any intervention from network layer or proxy agents.

A TCPW source continuously estimates the packet rate of the connection by properly processing returning ACKs. The estimate is used to set the congestion window and slow start threshold to be used after a congestion episode. That is, after three duplicate acknowledgments or after a timeout. The rationale of this strategy is simple: in contrast with TCP Reno, which simply halves the congestion window after three duplicate ACKs, TCP Westwood (TCPW) attempts to make a more "informed" decision. It selects a slow start threshold and a congestion window that are consistent with the effective connection rate at the time congestion is experienced.

The use of bottleneck bandwidth and connection rate estimation has been proposed before in the TCP literature. The best known examples are Packet Pair [10] and TCP Vegas [4]. Both of these schemes use the bandwidth computation to estimate the bottleneck backlog. The larger the backlog, the larger the congestion. The backlog feedback is used in both cases to control the send window. The Packet Pair scheme achieves perfect bottleneck bandwidth sharing. Unfortunately, it requires per flow queuing and Round Robin scheduling – a feature not available in commercial routers.

The "key innovation" of TCPW is to use the bandwidth estimate "directly" to drive the window, instead of using it to compute the backlog. The rationale is that if a connection is currently achieving a given rate, then it can safely use the window corresponding to that rate without causing congestion in the network.

In this paper we focus on the behavior of TCPW in random packet loss, caused by link error or wireless interference. Like Reno, TCPW cannot distinguish between buffer overflow loss and random loss. However, in presence of random loss, TCP Reno overreacts and reduces the window by half. TCPW, on the other hand, after packet loss and retransmission timeout, resumes with the previous window as long as the bottleneck is not yet saturated (i.e., no buffer overflow).

In order to study the behavior of TCPW in presence of random errors, an analytic model was developed using Markov chain techniques. This model is an important contribution in that it provides further insight in TCPW operation and allows cross validation against simulation and measurements results.

The paper is organized as follows. In Section 2, we provide an overview of the TCPW algorithm. In Section 3, we develop the analytic model of TCPW. Performance behavior results are in Section 4, and Section 5 concludes the paper.

Proceedings of IEEE Globecom 2001, S. Antonio, Texas, Dec. 2001

II. TCP WESTWOOD PROTOCOL

In TCP Westwood the sender continuously computes the connection BandWidth Estimate (BWE) which is defined as the share of bottleneck bandwidth used by the connection. Thus, BWE is equal to the rate at which data is delivered to the TCP receiver. The estimate is based on the rate at which ACKs are received and on their payload. After a packet loss indication, (i.e., reception of 3 duplicate ACKs, or timeout expiration), the sender resets the congestion window and the slow start threshold based on BWE. More precisely, cwin=BWE x RTT.

Another important element of this procedure is the RTT estimation. RTT is required to compute the window that supports the estimated rate BWE. Ideally, the RTT should be measured when the bottleneck is empty. In practice, it is set equal to the overall minimum round trip delay (RTTmin) measured so far on that connection (based on continuous monitoring of ACK RTTs).

Setting cwin and ssthresh in TCPW

Let us first assume that a sender has determined the connection bandwidth estimate (BWE), and let us describe in this section how BWE is used to properly set cwin and ssthresh after a packet loss indication.

First, we note that in TCPW, congestion window increments during slow start and congestion avoidance remain the same as in Reno, that is they are exponential and linear, respectively. A packet loss is indicated by (a) the reception of 3 DUPACKs, or (b) a coarse timeout expiration. In case the loss indication is 3 DUPACKs, TCPW sets cwin and ssthresh as follows:

```
if (3 DUPACKs are received)
   ssthresh =  (BWE * RTTmin) / seg_size;
   if (cwin > ssthresh) /* congestion avoid. */
      cwin = ssthresh;
   endif
endif
```

In case a packet loss is indicated by a timeout expiration, cwin and ssthresh are set as follows:

```
if (coarse timeout expires)
   cwin = 1;
   ssthresh = (BWE * RTTmin) / seg_size;
   if (ssthresh < 2)
      ssthresh = 2;
   endif;
endif
```

The rationale of the algorithm above is that after a timeout, *cwin* and the *ssthresh* are set equal to 1 and BWE, respectively. Thus, the basic Reno behavior is still captured, while a speedy recovery is ensured by setting *ssthresh* to the value of BWE.

A. Bandwidth Estimation

The TCPW sender uses ACKs to estimate BWE. More precisely, the sender uses the following information: (1) the ACK arrival times and, (2) the increment of data delivered to the destination. Let assume that an ACK is received at the source at time tk, notifying that dk bytes have been received at the TCP receiver. We can measure the sample bandwidth used by that connection as $b_k=d_k/(t_k-t_{k-1})$, where t_k-1 is the time the previous ACK was received. Letting $\Delta t_k=t_k-t_{k-1}$, then $b_k=d_k/\Delta t_k$.

Since congestion occurs whenever the low-frequency input traffic rate exceeds the link capacity [11], we employ a low-pass filter to average sampled measurements and to obtain the low-frequency components of the available bandwidth.

III. ANALYTIC MODEL

In this section we develop an analytic model of TCPW congestion control mechanism. The objective is to study the throughput taking into account the impact of the filter operation, in addition to the following system parameters: the bottleneck link transmission speed, the round trip time, and the link error rate.

A. System Description

Following [12], [13], and [14] we consider a saturated TCP source that always has fixed–size packets to send, of length L. The sender releases packets into a limited FIFO buffer that can hold up to B packets (packet arriving to a full buffer are discarded). The packets are then sent over a single bottleneck link with a speed of μ packets per second and a two way propagation delay of d, where d is assumed deterministic and consists of propagation time and any other processing delays excluding the transmission time and the bottleneck link queuing delay. Let T denote the time between the beginning of a packet transmission and the reception of an ACK for the same packet, excluding the queuing delay in the buffer. Then $T=d+1/\mu$. Since a packet loss would occur once the buffer is full, the maximum window size a connection can achieve is $W_{max}=B+\mu T=B+C$, where $C = \mu T$ is the *pipe capacity*.

The throughput realized by the system can be derived from the time evolution of the congestion window, *W(t)*. The evolution of this parameter has a cyclic structure, in which *W(t)* grows until a packet loss is detected. At this point, the missed packet is retransmitted and the transmission stops until the cumulative ACK for the last window of packets is received. The cycle, then, starts again with the congestion window and the slow start threshold set according to the bandwidth estimation at the instant in which the missed packet was detected.

In order to make the mathematical analysis feasible, a simplified model of the system is considered with the following assumptions:

1. d and μ constants;
2. Independent packet errors with probability ν;
3. Single packet loss in each cycle;
4. System always in the congestion avoidance phase;
5. Error-free feedback path;
6. Exponential timer backoff, fast recovery ignored.

By assumption 3, retransmitted packets are never dropped and the buffer overflow at the bottleneck link produces a single packet loss. Furthermore, by assumption 4 we assume that the congestion window at the end of a cycle is always greater than or equal to the slow start threshold of the following cycle. With very high error probability, these assumptions may not be satisfied. However, the close match between theoretical and simulation results that will be shown in the following prove that these approximations are not crucial and do not affect the accuracy of the model.

In the next subsection, we describe the cycle evolution under the hypotheses considered. Then, in subsections C and D, we describe the evolution of the congestion window at the beginning of successive cycles through a Markovian model and we derive an expression for the average system throughput.

B. Cycle evolution

Let the initial window size of the m^{th} cycle be denoted by $W_0^{(m)}$; also let $N_{drop}^{(m)}$ be the index of the packet lost at the end of the cycle. The evolution of the cycle is then determined by these parameters. For the sake of clarity, the reference to the cycle number m will be omitted in the following, whenever unnecessary. During a cycle, acknowledgments arrive in a series of consecutive *bursts*. ACKs in the same burst arrive $1/\mu$ seconds apart while consecutive bursts arrive T second apart, until the pipe capacity, C, is reached. After this point, ACKs arrive continuously every $1/\mu$ seconds. The congestion window is increased by one at the end of each burst.

If W_0 is the initial congestion window, the first burst contains exactly W_0 packets, the second W_0+1 and so on. Let B_k identify the k-th burst of a cycle. We denote the total number of ACKs in the burst B_k, or the *burst length*, by l_k. The burst length is related to W_0 and k as follows:

$$l_k = W_0 + k - 1. \tag{1}$$

Let B_{k^*} be the burst at which the pipe capacity is reached. k^* can be determined from $k^* = C - W_0 + 1$. After we reached the pipe capacity, packets are buffered and ACKs arrive continuously, $1/\mu$ second apart. At the end of the burst number $k^{**} = k^* + B$, the congestion window is set to $W_{max}+1$, and the last packet sent is discarded due to buffer overflow. Skipping the details of the fast retransmission and recovery mechanisms, we assume that the cycle ends $(C+B)/\mu$ seconds later, when the ACK for the dropped packet is missed.

We number the packets sent in a cycle taking into account the bursts B_k. Let s_k be the first packet of B_k. s_k can be determined as follows:

$$s_k = 1 + \sum_{j=1}^{k-1} l_k = 1 + (W_0 - 1)(k-1) + \frac{k(k-1)}{2} \tag{2}$$

Let n_{of} be the packet dropped due to buffer overflow. It is related to W_0 through the following formula:

$$n_{of}(W_0) = s_{k^*} + 2(C + B) \tag{3}$$

Denoting with k_n the number of the burst that contains the packet number n, and with r_n the offset of the packet in the burst B_{kn}, we can express each n, $1 \le n \le n_{of}$, as $n = s_{k_n} + r_{k_n}$, with

$$k_n = \left\lfloor -W_0 + \frac{3}{2} + \sqrt{W_0^{\,2} - W_0 - \frac{7}{4} + 2n} \right\rfloor; \qquad r_{k_n} = n - s_{k_n}. \tag{4}$$

For simplicity, we assume that the congestion window is set according to the bandwidth estimation at that instant and that a new window worth of packets is immediately sent. The instant in which the n-th ACK is received is, then, given by

$$t(W_0, n) = \begin{cases} Tk_n + \dfrac{r_n}{\mu} & n \le s_{k^*} \\ Tk^* + \dfrac{n - s_{k^*}}{\mu} & n > s_{k^*} \end{cases} \tag{5}$$

C. Slow start threshold evolution

At the beginning of a generic cycle m, TCPW sets the slow start threshold (and the congestion window) according to the value of the estimated bandwidth at the end of the previous cycle:

$$W_0^{(m)} \overset{def}{=} f_W\left(W_0^{(m-1)}, N_{drop}^{(m-1)}\right) \cong \max\left\{2, \left\lfloor \frac{BWE\left(W_0^{(m-1)}, N_{drop}^{(m-1)}\right)T}{L} \right\rfloor\right\} \tag{6}$$

where $N_{drop}^{(m-1)}$ is the ACK missed at the end of cycle $(m-1)$. The TCPW bandwidth can be modeled by:

$$BWE(n+1) \cong \alpha BWE(n) + (1-\alpha)\frac{b_{n+1} + b_n}{2}, \tag{7}$$

where α is related to the cut-off frequency of the low pass filter, while $BWE(0)$ is set to the bandwidth estimated at the

end of the previous cycle. In (7), b_n is given by $b_n = d_n / \Delta t_n$, where d_n is the amount of data acknowledged by the ACK for the *n-th* packet sent in that cycle, while Δt_n is the time elapsed since the reception of the previous ACK. On the basis of the previous observations about the ACK inter-arrival times, the bandwidth information carried by the *n*-th ACK is the given by:

$$b_n = \begin{cases} \dfrac{\mu}{C - l_{k_n - 1} + 1} & n = s_{k_n},\ k_n < k_C \\ \mu & \text{otherwise} \end{cases} \quad (8)$$

Substituting (8) in (7) and developing the recursion we obtain the bandwidth estimated after each ACK arrival.

D. Throughput estimation

On the basis of the formulas introduced in the previous subsections, it is easy to realize that the evolution of the initial congestion window size, in each cycle, can be modeled as a Markov process. Indeed, $W_0^{(m)}$ depends only on $W_0^{(m-1)}$ and $N_{drop}^{(m-1)}$, which is once again a function of the packet error probability and of $W_0^{(m-1)}$. Thus, the steady state probability vector for the initial window size can be derived from the state transition probability matrix of the process.

Let $P_{i,j}$ be the transition probability from $W_0^{(m-1)}=i$ to $W_0^{(m)}=j$:

$$P_{i,j} = \Pr\left\{W_0^{(m)} = j \middle| W_0^{(m-1)} = i\right\},\ \ i, j \in \{2,3,\ldots,C\} \quad (9)$$

The explicit computation of (9) would require the inversion of equation (7), in order to determine the value of $N_{drop}^{(m-1)}$ for which the bandwidth estimation at the end of the cycle corresponds to a window size of $W_0=j$, starting with a window $W_0=i$. Unfortunately, the equation cannot be inverted. Hence, the transition matrix is computed row by row, in the following way:

$$P_{i,j} = \sum_{n \in I_j(i)} p_{i,n}\ ; \quad i=2,...,C \quad (10)$$

where $p_{i,n}$ is the probability that the packet n is dropped, given that the initial window size is i, while $I_j(i)$ is the set of values of n for which the estimated bandwidth at the step n corresponds to a window size of j. Explicitly, we have

$$p_{i,n} = \Pr\left[N_{drop} = n \middle| W_0 = i\right] = \begin{cases} \nu(1-\nu)^{n-1},\ n < n_{of}(i) \\ (1-\nu)^{n-1},\ n = n_{of}(i) \end{cases} \quad (11)$$

and

$$I_j(W_0) = \left\{n : 1 \le n \le n_{of}(W_0),\ f_W(n, W_0) = j\right\}. \quad (12)$$

Finally, denoting with π_W the asymptotic probability of the initial window size W, the average throughput realized by the system is given by

$$\bar{\eta} = \sum_{W_0=2}^{C} \pi_{W_0} \sum_{n=1}^{n_{of}(W_0)} \frac{n-1}{t(W_0, n)} p_{W_0,n}\ . \quad (13)$$

IV. PERFORMANCE EVALUATION

In this section, we use the model described above to study the performance of TCPW and compare it with that of TCP Reno. The analysis is carried out considering the general case of a lossy channel as well as the ideal case of an error-free link.

A concise summary of the system parameters selected in our analysis is provided in Table 1.

Table 1. TCPW performance analysis parameters

Parameters	Lossy Link (2Mbps)
Packet- length	400
Link capacity μ (pkts/sec)	625
Round Trip Delay (T)	70 msec
Buffer size (B)	44
α	19/21

Link speed, packet length and round trip delay are typical, while the buffer size is set to the pipe capacity $C=\mu T$. The choice of the parameter α requires particular care. Basically, α represents the pole of the low-pass filter of equation (7), and it has to be large enough to properly smooth the bandwidth estimations. On the other hand, the response time of the filter must be short enough for the filter to quickly reach steady state bandwidth estimation. If the initial bandwidth estimation is zero and the input bandwidth samples carry a value of μ, the output of the filter is approximately:

$$BWE(n) = \mu\left(1 - \alpha^n\right)$$

We can determine an upper bound on α that will ensure that the relative error in the bandwidth estimation after the reception of n samples is less than k. This condition leads to $\alpha^C<k$ which, for our choice of k=0.1, $n=C$ and $\alpha=19/21$, is satisfied.

A. Lossy Connection

Fig.1 shows the average throughput attained by TCP Westwood and Reno evaluated using simulations and through the application of the analytic model over a wireless 2Mbps bottleneck. The first point of interest is the close match between analytic and simulation results for both the TCP protocols. The second and perhaps more significant result is the higher throughput TCPW achieves throughout a wide range of PER values. The two protocol throughputs converge in the limiting cases when either the errors are very frequent (trivially, the throughput tends to zero) or very rare (this last case will be elaborated upon in subsection *B*).

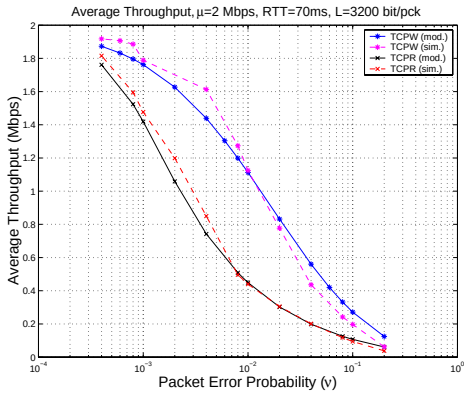

Fig.1 TCPW and Reno: analytic and simulation average throughput for a 2 Mbps lossy bottleneck

B. Ideal Error-free Connection

In the previous subsection, we analyzed TCPW performance in the general case of a lossy link bottleneck. In order to study the limiting behavior of the protocol, in addition to its sensitivity to the main system parameters, we also investigate the TCPW performance under the ideal condition of an error-free channel. The results presented in this subsection are obtained taking advantage only of the analytic model of TCPW developed in the previous section.

Fig. 2 shows the average throughput against the bottleneck buffer size B. We note that the throughput achieved by TCP Reno is always below that of TCPW, while the two curves are almost superimposed for values of B greater than the pipe size. The rationale of this result is that TCPW, due to its bandwidth estimation algorithm, starts each cycle filling the pipe capacity; while, TCP Reno halves its window size. Hence, for $B<C$, the new initial window is not sufficient to saturate the link. On the other hand, for $B>C$ the new window is greater than the pipe capacity and the performance is maximized. Finally, it is worth emphasizing that increasing B beyond the value of C does not result in a considerable gain in terms of throughput and this justifies our choice of $B=C$.

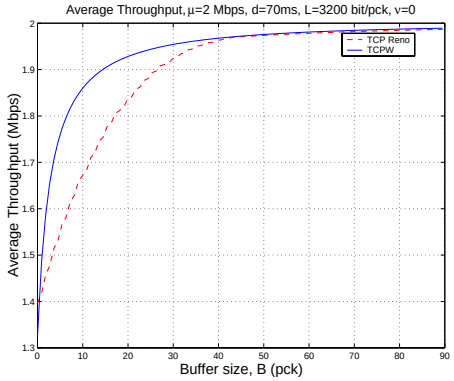

Fig. 2 TCPW and Reno: analytic average throughput vs. bottleneck buffer size in error-free channel

Fig. 3 shows the average throughput of TCPW and Reno with respect to variations of the bottleneck speed. In this case, the buffer size has been kept equal to the nominal pipe capacity (B=44 pcks). For link bandwidth less than the nominal value (μ=2Mbps), the two protocols exhibit the same performance. As the link bandwidth increases, the measurement-based nature of TCPW allows it to track the bandwidth variations of the bottleneck and to linearly build-up its performance; TCP Reno also improves its performance but in a less considerable way.

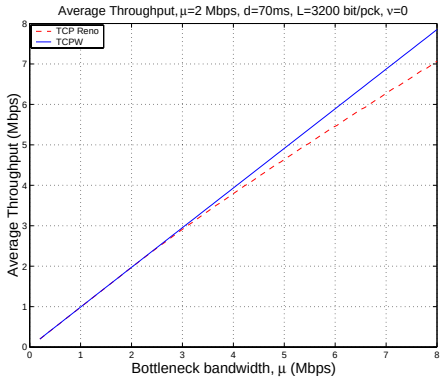

Fig. 3 TCPW and Reno: analytic average throughput vs. bottleneck bandwidth size in error-free channel

Finally, Fig. 4 shows the behavior of the average throughput achieved by TCPW and Reno against the round trip delay. From the figure it is clear that TCPW looks much more robust with respect to variations of RTT. In particular, it is worth pointing out that the Reno curve touches the TCPW curve for the nominal value of RTT=70 msec, for which the buffer size is equal to the pipe size.

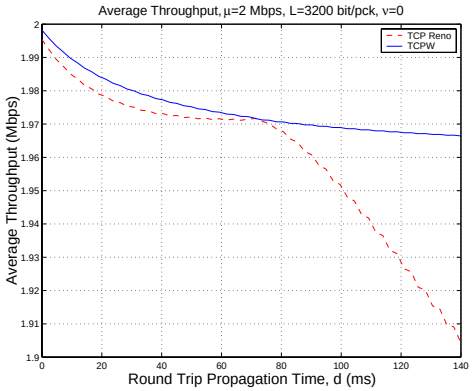

Fig. 4 TCPW and Reno: analytic average throughput vs. round trip delay in error-free channel

V. A SIMPLIFIED ANALYTIC MODEL

The analytic model we developed in Sec.III takes into account all the information the protocol needs in its dynamic

behavior. Unfortunately, such a highly detailed description makes the model somewhat cumbersome. On the other hand, for many purposes a simpler though slightly approximate model would be sufficient. In this section, on the basis of a reasonable approximation, we will derive a simpler model of TCPW behavior and show that the performance results we achieve match closely the ones obtained from the detailed model.

The major drawback of the detailed model is the lack of a closed form expression for the bandwidth estimated by the low pass filter. This is due to the irregular bandwidth samples the filter receives at the beginning of every burst. In the following we postulate that those samples do not significantly affect the estimated bandwidth. We assume that all samples are equal to μ except the first one in the cycle, which equals $1/RTT$. The input process of the filter can then be simplified as follows:

$$\tilde{b}_n = \begin{cases} 1/RTT & n = 0 \\ \mu & \text{elsewhere} \end{cases} \tag{14}$$

The solution of (7) is then straightforward for $n \geq 3$ and is given by:

$$BWE(n) = \mu - [\mu - BWE(2)]\alpha^{n-2}, \tag{15}$$

while $BWE(0)=W_0/RTT$ and the values of $BWE(1)$ and $BWE(2)$ can be easily derived through the recursion (7).

Formula (15) can be successfully applied to the evaluation of the transition probabilities $P_{i,j}$. The transition expressed as $\{W_0^{(m)}=j | W_0^{(m-1)}=i\}$ can be re-written in terms of the estimated bandwidth as:

$$\left\{ \left\lfloor RTT \cdot BWE(N_{drop}) \right\rfloor = j \,\middle|\, BWE(0) = \frac{i}{RTT} \right\} \tag{16}$$

and thus:

$$\left\{ \left\lfloor \frac{j}{RTT} \right\rfloor \leq BWE(N_{drop}) < \left\lfloor \frac{j}{RTT} \right\rfloor + 1 \,\middle|\, BWE(0) = \frac{i}{RTT} \right\} \tag{17}$$

Now, the relation in (17) can be solved exactly with respect to N_{drop}, obtaining as a result the set $[n_l, n_u)$. For the sake of conciseness we omit here the trivial but cumbersome explicit computation of n_l and n_u as they may assume different value when either or both are less than 3. The transition probabilities $P_{i,j}$ are then given by:

$$P_{i,j} = \Pr\left[W_0^{(m)} = j \,\middle|\, W_0^{(m-1)} = i \right] = \sum_{n=n_l}^{n_u - 1} p_{i,n} \tag{18}$$

where $p_{i,n}$ is given by (11).

Once the transition probability matrix has been obtained, the procedure to get the throughput follows the one presented in section III. Figure 5 shows the results obtained using the simplified models compared to the ones obtained from the complete model. From a quick visual inspection it is easy to notice that the two approaches lead to very close results, while the simplified technique is much faster in calculating the transition matrix probabilities.

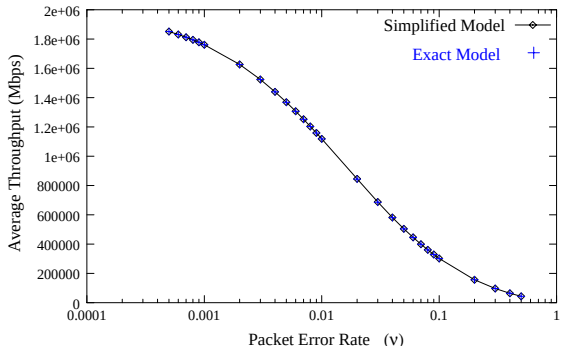

Fig. 5 Exact vs. simplified Model for a 2 Mbps lossy bottleneck

VI. Conclusion

In this paper we developed an analytic model for the TCP Westwood (TCPW) protocol. TCPW is a new TCP scheme, which requires modifications only in the TCP source stack and is thus compatible with TCP Reno and Tahoe destinations. Basically it differs from Reno in that it adjusts the *cwin* (congestion window) after a loss detection by setting it to the *measured rate* currently experienced by the connection, rather than using the conventional multiplicative decrease scheme.

The analytic model was developed in order to study the behavior of TCPW in presence of random errors and under different scenarios. It also represents a useful contribution in that it provides further insight in TCPW operation and allows cross validation against simulation and measurements results.

Following previous works on TCP Reno, we described the evolution of the congestion window by means of a Markov chain. However, unlike previous analytic models, the TCPW model must address the bandwidth estimation mechanism and its impact on system behavior.

The model was validated by means of simulation in the general case of a lossy link. Then we used it to derive TCPW performance in the ideal error-free channel.

The results show that, for a single connection case, TCPW protocol performs better than or, at least, as well as TCP Reno in terms of average throughput. The results also show that TCPW is more robust under varying buffer size, round trip delays and bottleneck bandwidth. The multiple connections case is under investigation and will be reported on in the near future.

REFERENCES

[1] V. C. Cerf and R. E. Kahn, “A Protocol for packet Network Interconnections,” IEEE Transactions on Communications, vol. COM-22, no. 5, pp. 637-648, May 1974.

[2] V. Jacobson, “Congestion Avoidance and Control,” ACM Computer Communications Review, 18(4) : 314 - 329, August 1988.

[3] T. Bonald, “Comparison of TCP Reno and TCP Vegas: Efficiency and Fairness,” In Proceedings of PERFORMANCE'99, Istanbul, Turkey, October 1999.

[4] U. Hengartner, J. Bolliger, and T. Gross, “TCP Vegas Revisited,” In Proceedings of IEEE INFOCOM'2000, Tel Aviv, Israel, March 2000.

[5] M. Gerla, R. Lo Cigno, S. Mascolo, and W. Weng, “Generalized Window Advertising for TCP Congestion Control,” CSD-TR 990012, UCLA, CA, USA, February 1999.

[6] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, “Explicit Window Adaptation: A Method to Enhance TCP Performance,” In Proceedings of IEEE INFOCOM'98, San Francisco, Ca, USA, March/April 1998.

[7] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, “Freeze-TCP: a True End-to-end TCP Enhancement Mechanism for Mobile Environments,” In Proceedings of IEEE INFOCOM'2000, Tel Aviv, Israel, March 2000.

[8] C. Casetti, M.Gerla, S. Lee, S. Mascolo and M.Y. Sanadidi. “TCP with faster recovery,” MILCOM 2000, Los Angeles, CA, October 2000.

[9] D. Clark, “The design philosophy of the DARPA Internet protocols,” In Proceedings of Sigcomm’88 in ACM Computer Communication Review, vol. 18, no. 4, pp. 106 - 114, 1988.

[10] J. C. Hoe, “ Improving the Start-up of A Congestion Control Scheme for TCP”, Proc. ACM SIGCOMM ’96, pp. 270-280.

[11] S. Q. Li and C. Hwang, “Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks,” IEEE/ACM Transactions on Networking, vol. 3, no. 1, pp. 10 - 25, Feb. 1995.

[12] L. Zhang, S. Shenker, and D.D. Clark. ``Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic". Proc. of ACM SIGCOMM '91, pages 133-147, Sep. 1991.

[13] T. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," IEEE/ACM Transactions on Networking, vol. 5, no. 3, 1997.

[14] Alhussein Abouzeid, Sumit Roy, Murat Azizoglu , Stochastic Modeling of TCP over Lossy Links, INFOCOM2000