Fluid-flow Analysis of TCP Westwood with RED

Jiwei Chen, Fernando Paganini, Ren Wang, M.Y. Sanadidi, Mario Gerla
University of California, Los Angeles, CA 90095
Email: {cjw,paganini}@ee.ucla.edu, {renwang,medy,gerla}@cs.ucla.edu

Abstract—This paper concerns TCP Westwood, a recently developed modification of TCP, in combination with RED queue management. We develop a fluid-flow model of the protocol, and use it to study both equilibrium and dynamic features. On the equilibrium side, we identify the scaling of window with loss-probability, and compare it to TCP NewReno. We also use the model to find the boundary of stability, beyond which we see large oscillations; we find that the stable region of TCP Westwood is enhanced with respect to TCP NewReno. Furthermore we show preliminary evidence that oscillations, when they occur, have a limited impact on network throughput.

I. INTRODUCTION

TCP Westwood (TCPW) has been recently proposed [?], [?], [?] as a source-side modification of TCP, with the aim of achieving more efficient link utilization of the network bandwidth, particularly in environments where losses are not only due to congestion (e.g., wireless links). The main idea is to estimate through the ACK stream the bandwidth that is being successfully obtained by the source. Upon a loss, rather than the usual multiplicative decrease, the congestion window is set to the estimated bandwidth times the estimated propagation delay. In this way the window will not decrease if there is no queuing delay, and does so in a less conservative way than in current protocols. Experimental results on this protocol have been very encouraging [?], [?], [?].

This paper is aimed at gaining an analytical insight into this protocol, similar to what has recently been done for TCP-Reno [?], [?], [?] based on fluid-flow models. These models describe very coarsely the packet level effects, but nevertheless have substantial predictive power, both in regard to equilibrium properties (utilization, queues, fairness) and to dynamics (stability, oscillations). For instance, they capture the well-known square root formula relating equilibrium window to loss probability. On the dynamic side, they can match observations of the onset of oscillations in TCP-Reno with RED [?] with an unstable bifurcation in the differential equation model [?], [?], [?]. Here, we will employ a fluid model of TCPW to find out how this new protocol measures up to a similar analysis.

In Section II, we provide an overview description of TCPW; we then proceed in Section III to give a fluid-flow model, and validate it by comparison between flow-simulations in Matlab, and Ns-2[?] simulations of the protocol. In Section IV we use the model to extract some conclusions on equilibrium behavior of TCPW. In particular we find an expression for the equilibrium window in TCPW; while more complex than that of Reno, it shows an attractive feature in terms of the scaling of TCPW in the high window (low-drop probability) regime, an issue that is attracting more attention as one moves to high speed networks [?]. In Section V we look at the dynamics of TCPW in combination with RED queue management, particularly its stability. Again, we are able to match the stability boundary in the model with good approximation to the oscillations observed in the protocol; we find, however, that the stability region is larger than that of TCP-Reno. We also consider the question as to whether these oscillations are benign or whether they are significant enough to have impact on throughput. The aforementioned studies of TCP-Reno had indeed shown cases where queue oscillations are significant enough so that they would temporarily empty the queue. This aspect is difficult to predict analytically, but from simulation studies it appears that oscillations in TCPW, when they occur, do not cause the queue to stay at zero and hence do not degrade throughput. Conclusions are given in Section VI.

II. TCPW AND RATE ESTIMATION OVERVIEW

TCPW is a sender-side only modification of TCP NewReno. In TCPW, the sender continuously monitors ACKs from the receiver and computes the current eligible rate estimate for this connection[?], [?], [?]. In this paper, we use a Rate Estimation (RE) method suitable for wired network and build the analytical model upon. To obtain a connection’s RE sample, we count all the data acknowledged during the last RTT, and compute the sample as the amount of date successfully delivered during the last RTT divided by RTT. Then the RE samples are fed into a low-pass filter to get a smoothed estimate. In essence, the $RE$ is the rate that the TCP connection is currently achieving. Thus, it must be a “feasible” rate by definition. Upon a packet loss (3 DUPACKs or a timeout) the sender sets cwnd and ssthresh based on the $RE$. More precisely, in case of loss indication of 3 DUPACKs, TCPW sets cwnd and ssthresh to $RE \times RTT_{min}$; in case of a timeout expiration, cwnd and ssthresh are set to 1 and $RE \times RTT_{min}$ respectively.

III. FLUID-FLOW MODEL AND VALIDATION

Our model originates in the work [?], [?] for TCP-Reno. As in those papers, we focus on the congestion avoidance phase, in which we are able to construct an analytic model which matches quite well with packet level simulation results in Ns-2.

A. Network Model

The network system is modelled as a set of $L$ links with capacities $c_l, l \in L$. There are $N$ sources indexed by $i$. Each source $i$ uses a set of links $L_i \subseteq L$. Define a $L \times N$ routing matrix

$$R_{li} = \begin{cases} 1 & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases}$$
We begin by modelling round trip time. Let $\tau_i$ be the round trip time of source $i$. $\tau_i$ consists of round trip propagation delay and queuing delay on the links in the set $L_i$:

$$\tau_i(t) = d_i + \sum_l R_{li} b_l(t)/c_l$$  \hspace{1cm} (1)

where $d_i$ is the round trip propagation delay and $b_l(t)$ is the instantaneous queue length at link $l$.

Associated with each link $l$ is a marking probability $p_l(t)$, e.g., drop probability in RED. $p_l(t)$ will be feedbacked to the sender as an indication of network congestion condition. Correspondingly, associated with each source $i$ is a congestion measure $q_i(t)$ (e.g., loss rate detected by the source), which is the aggregate delayed marking probability of the links in $L_i$:

$$q_i(t) = \sum_l R_{li} p_l(t - \tau^b_l(t))$$  \hspace{1cm} (2)

where $\tau^b_l(t)$ is the backward delay from link $l$ to source $i$.

As usual, the sending rate of $i$-th source is related to the window by the following equation:

$$w_i(t) = \frac{c_l}{\tau_i(t)} x_i$$  \hspace{1cm} (3)

At link $l$, the aggregate arriving rate of all connections $y_l(t)$ depends on the previous source rates:

$$y_l(t) = \sum_i R_{li} \frac{w_i(t - \tau^f_l(t))}{\tau_i(t - \tau^f_l(t))}$$  \hspace{1cm} (4)

where $w_i(\cdot)$ is the window size and $\tau^f_i(\cdot)$ is the forward delay from source $i$ to link $l$.

For all links $l \in L_i$, given the aggregate arriving rate and link capacity, we can calculate instantaneous queue length $b_l(t)(b_l(t) \geq 0)$ from:

$$b_l(t) = y_l(t) - q_i(t) - c_l = \sum_i R_{li} \frac{w_i(t - \tau^f_l(t))}{\tau_i(t - \tau^f_l(t))} - c_l$$

$$= \sum_i R_{li} \frac{w_i(t - \tau^f_l)}{d_i + \sum_l R_{lk} b_k(t - \tau^f_l)/c_k} - c_l$$  \hspace{1cm} (5)

This equation shows that a queue will build up when aggregate arriving rate exceeds the link capacity.

B. Active Queue Management

We involve AQM in our model, and choose the RED queue management [?] which is well captured in fluid flow models. In RED, drop probability $p_l(t)$ is given by the nonlinear function:

$$p_l(t) = \begin{cases} 0 & 0 \leq r_l(t) < b_{min} \\ \rho_l(r_l(t) - b_{min}) & b_{min} \leq r_l(t) \leq b_{max} \\ 1 & r_l(t) > b_{max} \end{cases}$$

where $b_{min}$, $b_{max}$ and $p_{max}$ are RED parameters as illustrated in Fig.1. $p_l = p_{max}/(b_{max} - b_{min})$ and $r_l(t)$ is the average queue length, which is an exponentially moving average of instantaneous queue length $b_l$. At packet level in Ns-2, the average queue length $r_l$ is computed when every packet arrives at the link $l$ using a queue averaging weight $\beta_l : 0 < \beta_l < 1$:

$$r_l(k + 1) = (1 - \beta_l) r_l(k) + \beta_l b_l(k)$$  \hspace{1cm} (6)

It is natural to convert this difference equation to a differential equation:

$$\dot{r}_l(t) = \alpha_l (r_l(t) - b_l(t))$$  \hspace{1cm} (7)

where $\alpha_l = c_l \log(1 - \beta_l)$ (for details, refer to [?]).

We assume that the system operates in the region $b_{min} \leq r_l(t) \leq b_{max}$ at steady state. Then the drop probability is:

$$p_l(t) = \rho_l(r_l(t) - b_{min})$$  \hspace{1cm} (8)

By differentiating equation (8) and from (7), we get:

$$\dot{p}_l(t) = -\alpha_l p_l(t) + \alpha_l p_l(b_l(t) - b_{min})$$  \hspace{1cm} (9)

C. TCP Westwood Model

In this subsection, we model the specific behavior of TCPW in congestion avoidance. Since the additive increase portion is unchanged from TCP-Reno, the increase term in the window dynamics in equation (10) below is left unchanged from [?]. Namely, if $w_i(t)$ is the congestion window of the $i$-th source, we include an increase of $1/w_i$ for every positive ACK received, and multiply this by the rate of such ACKs, $x_i(t - \tau_i) (1 - q_i(t))$, where $x_i$ is the sending rate.

TCPW differs from NewReno in the decrease portion: for each loss, instead of halving the window, we set the window of $i$-th source to $RE_i * d_i$; equivalently, we decrease the window by $w_i(t) - RE_i(t) d_i$ per loss, and multiply this by the rate of losses, $x_i(t - \tau_i) q_i(t)$.

Overall, the TCPW window behavior is described by the following nonlinear differential equation:

$$\dot{w}_i(t) = x_i(t - \tau_i) \frac{1 - q_i(t)}{w_i(t)} \left[ x_i(t - \tau_i) q_i(t) w_i(t) - RE_i(t) d_i \right]$$  \hspace{1cm} (10)

Next we proceed to model the Rate Estimation(RE) mechanism. As explained in the previous section, the rate estimate is calculated based on the rate of positive ACKs received, smoothed by a lowpass filter. This can be expressed as:

$$T \times RE_i(t) + RE_i(t) = x_i(t - \tau_i) (1 - q_i(t))$$  \hspace{1cm} (11)

where $T$ is the filter time constant or $1/T$ is the cutoff frequency.

D. Validation Experiments

The differential equations in the previous sections constitute a closed-loop system. We now proceed to validate the analytic model by comparing flow simulations in Matlab with Ns-2 simulations of TCPW/RED. In Matlab, we approximate the times $\tau_i(t), \tau^b_i(t), \tau^f_i(t)$ by their equilibrium values when they appear in the argument.

We designed several scenarios to validate our analytic model. The results indicate that our analytic model is capable of accurately capturing TCPW behavior.
Experiment 1: Single-link Case.

The network topology features a single bottleneck link shared by 20 identical long lived connections. The link capacity is $c=9000$ pkts/sec, packet size is 500 bytes, and RED is used at the bottleneck link as active queue management. RED parameters are $p_{\text{max}}=0.1$, $b_{\text{min}}=150$ pkts, $b_{\text{max}}=600$ pkts, and queue averaging weight in Ns-2 is 0.0001. Since we only consider the congestion avoidance phase, there is no need to compare the slow start phase. So we feed data collected from Ns-2 simulation into Matlab equations during the first 7 seconds and this would not affect the steady state behavior of the analytic model. The comparisons are shown in Fig.2, Fig.3 and Fig.4 with round trip propagation delay 40ms, 90ms, and 160ms respectively.

We compare the instantaneous queue length ($\text{instq}$), congestion window ($\text{cwnd}$) and rate estimation ($\text{RE}$). The $\text{cwnd}$ and $\text{RE}$ in Ns-2 are averaged congestion window and averaged rate estimation over 20 sources, respectively.

Fig.2 shows a stable case (see the definition and analysis of stability in Section V), the instantaneous queue length and congestion window of the analytic model are reasonably well matched with Ns-2 results (see Fig.2 (a) and (b)), except for some random oscillations which we attribute to packet level noise. In Fig.2 (c), rate estimation don’t match perfectly. Rate estimation from Ns-2 simulation slightly overestimates the fair bandwidth, the reason being that TCP traffic is bursty and RTT computation in Ns-2 is not accurate, but the bias is small (less than 20 pkts/sec). Fig.3 is a barely unstable case and Fig.4 is a completely unstable case. The instantaneous queue length and congestion window are matched quite well and their oscillation frequencies are quite close. In fact, as the system grows more unstable, the oscillation is more deterministic and better approximated by the fluid flow model.

We also have validation experiments for multi-link scenarios. Due to space constraints, please refer to [?] for details.

IV. Analysis of the Equilibrium State

We have validated the fluid model as an approximation to the packet level behavior. In this section, as a first application of the model, we use it to analyze the equilibrium behavior of TCPW, in particular the relationship between window size and loss probability, and compare with those of TCP NewReno.

Setting to zero the differential terms in our differential equations in section III, we derive the following expression for the equilibrium state $(w^*, p^*)$ of TCPW:

$$w_{\text{tcpw}}^* = \sqrt{\frac{1 - p^*}{p^*[1 - (1 - p^*)d/\tau^*]}}$$

(12)

In comparison, the equilibrium for NewReno from [?] is

$$w_{\text{reno}}^* = \frac{2(1 - p^*)}{p^*}$$

(13)

In NewReno, $p^*$ is usually set small so that $w^*$ is square-root related with $p^*$. This property has significant implications when we try to operate with large congestion window, which is necessary to achieve high throughput. In that case, consider the product $w^*p^*$ which reflects how many losses on average one observes per RTT. These losses are precisely the congestion signals received by the sources. In NewReno, this product will be proportional to $\sqrt{p^*}$, and hence become vanishingly small as one moves to larger windows. In other words the congestion “epochs”, the intervals between consecutive drops will become longer and longer, which makes it very difficult to operate the network in a satisfactory way. For more discussion on this issue see [?].

We studied this issue for TCPW. The first remark is that the expression (12) contains $d/\tau^*$ and hence is not completely “intrinsic” to TCPW, but also depends on other factors (e.g. AQM) that influence this value. Nevertheless, our preliminary study using RED is very encouraging, as shown by the results in Fig.5 and Fig.6. The second remark is that for high speed networks, $b^*/c$ would be very small, so $d/\tau^* \simeq 1$. For same $p^*$, from equation (12) and (13),

$$\frac{w_{\text{tcpw}}^*}{w_{\text{reno}}^*} \simeq \frac{1}{\sqrt{2p^*}}$$

(14)
Fig. 5. Log Plot of $w^*$ and $p^*$, TCPW achieves larger cwnd than NewReno

Equation (14) shows that, especially for small loss rate, we will have much larger window in TCPW than that in NewReno. Another observation in this case, from equation (12) we will have $w_{TCPW}^* p^* \approx 1$. This indicates that at steady state, there is almost constantly one loss per RTT regardless of window size.

Fig.5 shows the log plot of $w^*$ and $p^*$ of analytic model and Fig.6 shows the relation between the losses $w^* p^*$ and window size $w^*$ using results in Fig.5. In Fig.5, log $w^*$ of TCPW and NewReno are both linearly related with their log $p^*$, but the slope for TCPW is steeper, which means for the same loss rate, TCPW can achieve larger window, thus higher throughput.

We use Ns-2 to verify this analytic result. Fig.7 shows the result of NewReno(analytic model from [?]), when link capacity goes up from 2Mb to 36Mb. In Ns-2, we sample Ns-2 data every 10ms and compute the average window over all sources and the average drop probability respectively from 10 seconds to 70 seconds. For all experiments in this section, we fix the round trip propagation delay at 90ms using 20 sources and same RED parameters as in previous section. In Fig.7, our analytic $w^*$ and $p^*$ are very close to those Ns-2 results, and further verify this advantage of TCPW.

V. STABILITY OF TCPW AND LINK UTILIZATION

A. Stability of TCPW

In this section we study the stability of analytic model and focus on the single link case. There are $N$ identical sources using the bottleneck link with RED. Linearizing around equilibrium points $(w^*, \tau^*, b^*, p^*, RE^*)$ and dropping all subscripts of variables in section III), we have the closed loop system transfer function:

$$L(s) = -\frac{1}{s} \left[ \frac{X w^* p^* (1 - p^*)}{M(s)} e^{-\tau^* s} \right. + \left. \frac{Y (1 - p^*) (s + 1) M(s)}{(Ts + 1)} e^{-\tau^* s} + Z \right]$$

(15)

where $M(s) = \tau^* \alpha s + \tau^* s^2 + e^{-\tau^* s} (\alpha + s) + N w^* \alpha p^*$, $X = x^* [1/w - (w^* - RE^* + d)]$, $Y = x^* p^* d$, $Z = x^* [(p - 1)/(w^*)^2 - p^*]$

Using the Nyquist stability criterion, the system is stable if and only if the loop function, evaluated at $s = j\omega$, doesn’t encircle the critical point(-1,0).

We repeat the experiments to find when the system is stable or barely unstable based on its linear model, and produce its stability region. We use the single link with capacity $c = 8000, 9000, ..., 15000$ pks/sec, $N = 20, 30, ... 60$, and increase the round trip propagation delay every 5ms (the resolution for delay is 5ms), to identify the stability region. For all experiments, we fix the same RED parameters as in Experiment 1. For every pair $(c, N)$, we compute the critical delay which makes the system barely unstable.

Fig.8 shows their stability region (The stable region is below respective curve and unstable region above). TCPW/RED is always more stable than NewReno/RED for each $(c, N)$ pair, which means TCPW can achieve better network performance, such as throughput and link utilization, which is very desirable in current networks. Also the above simulation shows that TCPW/RED is similar with NewReno/RED, i.e., more unstable when delay or capacity increases, but more stable when $N$ increases.

B. Impact of Stability on Link Utilization

Whether stability of a protocol is a crucial property is a subject of considerable debate. In particular, while oscillations of the queue can give undesirable effects such as delay jitter, it is
the oscillation amplitude is so significant that the queue spends not worry about oscillations. Utilization could be affected if the oscillation amplitude is so significant that the queue spends non-trivial amounts of time in the empty state.

In order to find when the link utilization will drop, repeating Experiment 1, Fig.10 shows the link utilization of TCPW and NewReno when round trip propagation delay increases from 50ms to 500ms. The link utilization for TCPW is high, and keeps high link utilization after the first drop in spite of longer delay, while the utilization in NewReno decreases with increased delay. In the simulation the link utilization is computed from 150 to 250 seconds for the purpose of excluding synchronization effect of slow start. For every pair \((c, N)\), we give the first utilization drop curve of TCPW and NewReno, which points out at what round trip propagation delay (critical delay) that the utilization drops below 1 for the first time. The critical delay is found by increasing delay every 10ms to find the first utilization drop. Fig.11 shows the first utilization drop curves for different \((c, N)\). In the simulation, TCPW always achieves better link utilization than NewReno.

VI. CONCLUSION

TCPW is a recently proposed congestion control scheme which can improve TCP performance under various network conditions, and maintain reasonable friendliness toward current TCP variant, namely, Reno/NewReno[7].

We have developed an analytic model of TCP Westwood combined with RED queue management, that approximates the large-scale features observed in Ns-2 simulations. In particular, it is able to accurately predict equilibrium windows and queues, and to a reasonable degree the point of transition to instability, where one begins to see wide oscillations.

The analytical model has helped us show that TCPW has some interesting properties as compared with the existing protocols. In particular, in regard to the scaling of equilibrium windows with loss-probability, TCPW has an improved behavior allowing in principle for accurate control at high equilibrium windows. Also we find that the stability region of the protocol is significantly enhanced for all system parameters: number of sources, RTT and link capacity. Finally, we explored via Ns-2 simulations the impact of oscillations on performance, defined by link utilization. We find that, in contrast with NewReno/RED that exhibits a significant drop in utilization as the protocol becomes more unstable, TCPW/RED maintains a high utilization.

This analytic model gives us insight into TCPW behavior under error free condition. An investigation is still needed in regard to its behavior in the presence of random losses as for example in a wireless environment. This we will pursue in our future research.

REFERENCES