Fluid-based Modeling of TCP Veno

Ke Zhang, Cheng Peng Fu, Chuan Heng Foh, Maode Ma, Jian Ling Zhang
School of Computer Engineering
Nanyang Technological University, Singapore
{Y030069, ascpfu, aschfoh, emdma, zhan0229}@ntu.edu.sg

Abstract—This paper makes use of the fluid-based approach to model the throughput of TCP Veno flow over wired/wireless networks. A generalized formula is derived between Veno’s throughput and its window evolution parameters, packet loss rate and round-trip time. Simulation experiments and real network measurements are conducted to validate the accuracy of this model.

I. INTRODUCTION

TCP is a connection-oriented, reliable and in-order transport protocol. It is known that the current legacy TCP, namely TCP Reno[1], suffers from performance degradation in the wireless networks due to the lack of differentiation between the random and the congestion losses. TCP Veno [2], a sender-side TCP enhancement, was proposed to solve this problem and adopted by Linux Kernel since version 2.6.18 [3]. Veno has the ability to identify network states and adjust the additive increase multiplicative decrease (AIMD) strategy to tackle random losses. Specifically, Veno estimates the number of packet loss occurring in this state is considered as a random loss. Otherwise, the network is in a non-congestive state, and packet loss occurring in this state is considered as a congestion loss. In the current Veno implementation, the parameter \( \beta \) is set to 3.

Veno adjusts its AIMD algorithm based on the network states. It refines the multiplicative decrease algorithm as follows,

\[
N = (\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT}) \times BaseRTT
\]

where \( BaseRTT \) is the minimum of measured round-trip time and reset when packet loss is detected. \( RTT \) is the actual round-trip time of a tagged packet. Veno compares \( N \) with a threshold parameter, \( \beta \) to identify network states. If \( N \geq \beta \), the network is said to have evolved into a congestive state, and packet loss occurring in this state is considered as a congestion loss. Otherwise, the network is in a non-congestive state, and packet loss occurring in this state is considered as a random loss. In the current Veno implementation, the parameter \( \beta \) is not considered in that model [9], e.g., the model did not take into account of the delay increase factor; for computation, the formula also requires an input value of \( W_{max} \) (the maximum congestion window during the TCP Veno evolution), which must be obtained by empirical measurement in advance. In this paper, we eliminate these restrictions in [9] and make use of a fluid-based approach to derive a much more generalized throughput formula, and carry both simulation experiments and real network measurements to prove this theoretical formula. Furthermore, this generalized formula can predict how TCP performance is affected by its different AIMD parameters.

The remainder of this paper is organized as follows. In Section II, we develop the fluid-flow model and then derive the steady-state throughput formula for TCP Veno. Section III presents simulation results to validate our throughput formula. The real network measurement results are shown in Section IV. Section V concludes this paper and discusses future work.

II. TCP VENO THROUGHPUT MODEL

In this section, we will review the fluid-flow model for TCP Reno [4]-[5] firstly, and then make use of this fluid-flow model to derive the throughput formula for TCP Veno.

A. TCP Reno model

Consider that the network comprises of \( L \) links with capacity \( c_i, i \in L \). There are \( I \) sources indexed by \( i \). Each source \( i \) uses a set of links \( L_i \subseteq L \). So we have a \( L \times I \) routing matrix \( \{R_{li}\}, \) where \( R_{li} = 1 \) if \( l \in L_i \), or 0 otherwise. The round-trip时间 of source \( i \) at time \( t \), \( \tau_i(t) \), is

\[
\tau_i(t) = d_i + \sum_l R_{li} \frac{q_i(t)}{c_i} 
\]

where \( d_i \) is the round-trip propagation delay of source \( i \) and \( q_i(t) \) is the instantaneous queue in link \( l \) at time \( t \). The loss probability of link \( l \) at time \( t \) is \( u_l(t) \) for the drop-tail queue management scheme. \( u_l(t) = 1 \) if the queue is full, or 0 otherwise. Let \( \gamma_l(t) \) be the random loss rate in link \( l \) (if \( \gamma_l(t) = 0 \) for all \( l \in L \), our model depicts the wired networks). Consider that the random loss rate is small, then the end-to-end loss probability detected by source \( i \) is approximated as

\[
p_i(t) = min(1, \sum_l R_{li} u_l(t - \tau^b_{li}(t)) + \sum_l R_{li} \gamma_l(t - \tau^b_{li}(t)))
\]
where \( \tau_{li}^f(t) \) is the backward delay from link \( l \) to source \( i \). Denote the congestion window size of source \( i \) at time \( t \) by \( w_i(t) \), then sending rate, \( x_i(t) = \frac{w_i(t)}{\tau_i(t)} \). Further denote the aggregate arriving rate of link \( l \) at time \( t \) by \( y_l(t) \). For all links where \( l \in L \), given the aggregate delivery rate and link capacity, we can calculate the derivative of the queue length \( q_i(t) \) where \( q_i(t) > 0 \) by

\[
\dot{q}_i(t) = y_i(t) - c_i - \sum_l R_{li} x_i(t - \tau_{li}^f(t)) - c_l
\]

(1)

where \( \tau_{li}^f(t) \) is the forward delay from source \( i \) to link \( l \). Let \( b \) denote the number of packets that are acknowledged by a received ACK (e.g. the receiver sends a cumulative ACK for \( b \) packets). This quantity is important to TCP Veno as Veno uses \( N \) as a congestion control parameter.

**B. Modeling of TCP Veno**

The route backlog \( N \) can be expressed as

\[
N = w_i(t)(1 - d_i/\tau_i(t)).
\]

(3)

This quantity is important to TCP Veno as Veno uses \( N \) to estimate the network states and adjust its AIMD algorithm. If \( N < \beta \), each received ACK adds \( w_i(t) \) by \( \frac{1}{w_i(t)} \). However, each packet loss detection adjusts \( w_i(t) \) to \( \frac{1}{w_i} \) of its original value, where the packet loss rate is given by \( x_i(t - \tau_i(t))p_i(t) \). For the case of \( N \geq \beta \), each received ACK adds \( x_i(t - \tau_i(t))p_i(t)w_i(t) \) by \( \frac{1}{w_i(t)} \). Besides, each packet loss detection adjusts \( w_i(t) \) to \( \frac{1}{2} \) of its original value. Given the above, the window evolution rate can be determined by

\[
\dot{w}_i(t) = \begin{cases} 
  x_i(t - \tau_i(t))(1 - p_i(t))/b \cdot w_i(t) & \text{if } N < \beta \\
  x_i(t - \tau_i(t))(1 - p_i(t))/b \cdot mnw_i(t) & \text{if } N \geq \beta 
\end{cases}
\]

(4)

Similar to [4]-[5], we linearize (4) to study its performance around the equilibrium. Linearization on (4) and applying

\[
\lim_{T \to \infty} \frac{1}{T} \int_0^T \dot{w}_i(t)dt = 0
\]

leads to

\[
w_0 = \begin{cases} 
  \sqrt{\frac{n}{b \rho_0}}, & \text{if } N < \beta \\
  \frac{2}{m \rho_0}, & \text{if } N \geq \beta
\end{cases}
\]

(5)

where \( w_0 \), \( \rho_0 \), and \( \tau_0 \) are the quantities of \( w_i(t) \), \( p_i(t) \), and \( \tau_i(t) \), respectively. These values are observed at the equilibrium state for flow \( i \). In the above, we also apply \( \frac{1}{\rho_0} \approx 1 \) as \( \rho_0 \) is small implying \( \frac{1}{\rho_0} \gg 1 \).

Let \( P(N \geq \beta) \) be the probability that \( N \geq \beta \), (5) becomes

\[
w_0 = \sqrt{\frac{n}{b \rho_0}} - \left( \sqrt{\frac{n}{b \rho_0}} - \sqrt{\frac{2}{m \rho_0}} \right) P(N \geq \beta).
\]

(6)

We now focus on solving for \( P(N \geq \beta) \). Fig. 1 depicts the typical evolution of \( N \) in a period between two consecutive packet loss events. At the start of the period, a packet loss occurs causing Veno to reset its BaseRTT and reduce its \( w(t) \) resulting \( N = 0 \). As time progresses, \( w(t) \) increases, and backlog increases up on the links first, then at the router’s buffer. When \( N \geq \beta \), Veno enters the congestive state with the setting \( m = 2 \) (i.e. a slower increasing rate).

Let \( n_p \) and \( n_c \) be the duration measured in RTT between two consecutive packet loss events and the duration of the congestive state within, then

\[
P(N \geq \beta) = \frac{n_c}{n_p}.
\]

(7)

With a common assumption that each packet is equally likely to drop with probability \( p_0 \) [8], the number of packets transmitted within \( n_p \) is geometrically distributed with mean of \( \frac{1}{p_0} \). We also know that in the equilibrium state, a source transmits \( w_0n_p \) packets within \( n_p \). Hence we get

\[
n_p = \frac{1}{w_0p_0}.
\]

(8)

We now turn our attention to \( n_c \). The quantity of \( N \) in the equilibrium state, denoted \( N_0 \), can be computed by

\[
N_0 = \frac{S}{n_p}
\]

(9)

where \( S \) is the total number of backlogged packets that the router carries during the \( n_p \) period. According to Fig. 1,

\[
S = \frac{1}{2}n_c^2 + \frac{1}{2}(2n_c + \beta)\beta \approx \frac{1}{2}n_c^2.
\]

(10)

Practically, \( \beta \) is chosen to be small for effective operation. Hence, \( S \) is dominated by the first term in (9) which gives the described approximation. Combining (7)-(9) yields

\[
N_0 = \frac{S}{n_p} = \frac{1}{2mn_c^2w_0p_0}.
\]

(11)

Linearizing (3) in the equilibrium state gives

\[
N_0 = w_0(1 - \frac{\tau_{\text{min}}}{\tau_0})
\]

(11)
where $\tau_{\text{min}}$ is the smallest observed RTT (i.e. BaseRTT).

Solving (10) and (11) for $n_c$ gives

$$n_c = \sqrt{\frac{2m}{p_0} \left(1 - \frac{\tau_{\text{min}}}{\tau_0}\right)}.$$  \hfill (12)

With (7) and (12), $P(N \geq \beta)$ can be expressed as

$$P(N \geq \beta) = \frac{n_c}{n_p} = w_0 \sqrt{2mp_0 \left(1 - \frac{\tau_{\text{min}}}{\tau_0}\right)}.$$ \hfill (13)

The above formula finally yields

$$w_0 = \frac{1}{\sqrt{b + (\sqrt{2mn} - 2)\sqrt{1 - \tau_{\text{min}}/\tau_0}}} \cdot \sqrt{\frac{n}{p_0}}$$ \hfill (14)

and $B$, the Veno’s throughput, as

$$B = \frac{w_0}{\tau_0} = \frac{1}{\tau_0(\sqrt{b + (\sqrt{2mn} - 2)\sqrt{1 - \tau_{\text{min}}/\tau_0}})\sqrt{\frac{p_0}{n}}}.$$ \hfill (15)

It is easy to see that when applying Reno’s setting (i.e. $m = 1, n = 2$), with $b = 1$, (14) is reduced to $w_0 = \sqrt{2/p_0}$ which is identical to the earlier results developed for TCP Reno (i.e. (2) in [6] and (13) in [7]).

It is also interesting to evaluate the throughput when Veno detects a random loss and chooses not to decrease its congestion window. This is done by applying $n \to \infty$. With this application, Veno’s throughput becomes

$$B = \frac{1}{\tau_0\sqrt{2mp_0(1 - \tau_{\text{min}}/\tau_0)}}.$$ \hfill (16)

### C. Modeling of TCP Veno with timeouts

Now we extend our analysis to include the effect of Time-outs (TO). Following the approaches given in [5], [8], let $Q(w_0)$ be the probability that a loss is a TO loss given that there is a loss event, and $Q(w_0) \approx \min(1, 3/w_0)$. Hence, a TO event occurs with probability $p_0Q(w_0)$.

The sender transmitting a window of data now needs a period of $\tau_0$ plus a possible timeout period. Given by Padhye et al. [8], the timeout period for one TO loss in the system equilibrium state, denoted $T_P$ here, is expressed as

$$T_P = T_0 \frac{1 + p_0 + 2p_0^2 + 4p_0^3 + 6p_0^4 + 16p_0^5 + 32p_0^6}{1 - p_0}$$

where $T_0$ is the duration of the waiting period for a sender to retransmit an unacknowledged packet, and each following unsuccessful retransmission doubles the waiting period, until $64T_0$ is reached. Applying this extension, the throughput for Veno can be rewritten as

$$B = \frac{w_0}{\tau_0 + w_0p_0Q(w_0)T_P}.$$ \hfill (17)

### D. TCP Veno and TCP Reno

In (15), we give the throughput formula for TCP Veno, precisely, $B_{\text{Veno}} = B(2, 5)$, where

$$B(m, n) = \frac{1}{\tau_0(\sqrt{b + (\sqrt{2mn} - 2)\sqrt{1 - \tau_{\text{min}}/\tau_0}})\sqrt{\frac{p_0}{n}}}.$$ \hfill (18)

With the above, the throughput for TCP Reno can be obtained simply by $B_{\text{Reno}} = B(1, 2)$. When the two
flows operate together sharing a common router, to find appropriate settings for Veno that give similar bandwidth utilization between the two flows, we have the condition that $B(m, n) = B(1, 2)$ where $m$ and $n$ correspond to Veno’s settings. Assuming same packet loss rate between the two flows, the condition yields

$$m = \frac{\left(\frac{\sqrt{\frac{\pi}{2}} - 1}{\sqrt{1 - \frac{1}{mn}}} + 2\right)}{2n}. \quad (18)$$

The above result describes the relationship between $m$ and $n$ settings of Veno where Veno will utilize the same bandwidth as that of Reno when two flows operate together. In Fig. 2, we plot the above result. As can be seen, for a wide range of $n$, a practical $m$ is between 1 and 2. Veno chooses $m = 2$ as its setting. This allows Veno to share similar bandwidth with Reno when they operate together.

## III. Model validation

We use network simulation tool NS-2 to validate our developed throughput formula (17). Fig. 3 depicts the simulated network topology. Each source connects to a router using a 10 Mbps link with 0.1 ms propagation delay. The bottleneck link runs at 3 Mbps with 40 ms delay. Drop-Tail router with buffer 60 packets is used at the bottleneck link. Destination connections are wireless links with random loss rate varying from $10^{-5}$ to $10^{-1}$ following exponential distribution. Packet size is 1000 bytes and ACK is 40 bytes.

The first experiment considers a single flow with different values of delay. We vary the one-way delay of the flow from 40 ms to 320 ms. Fig. 4 shows that the model matches the experiment results quite well under different values of delay and random loss rate.

The second experiment considers a single flow with UDP background traffic. The UDP traffic follows Pareto distribution (shape parameter 1.7), with burst and idle periods of 50 ms, and rates varying from 0.5 Mbps to 3 Mbps. The one-way propagation delay of TCP Veno flow is fixed at 80 ms. As shown in Fig. 5, an excellent match between the analytical and simulation results is reported.

Next, we measure the performance of Veno under various settings of $m$ and $n$. Matching results between the analytical and simulation results presented in Fig. 6 validate the accuracy of our model. It is also recognized that Veno’s performance can be improved with the setting of $n$ to be a bigger value.

Finally, the performance evaluation for mixed multiple flows is carried. In details, we consider the same numbers of Veno and SACK TCP flows co-existing in the network. The bottleneck bandwidth is set to 10 Mbps, and the number of total connections varies from 4 to 32. The values of delay for Veno and SACK flows are both uniformly distributed between 40 ms and 160 ms. Fig. 7 gives the analytical and simulation results of the Veno flow of 160 ms delay, which also confirms the accuracy of our analytical model.

## IV. Real network experiments

The topology of the real network is shown in Fig. 8. The server is running Linux 2.6 where TCP Veno is implemented, and it is connected to the gateway via Ethernet. The gateway is FreeBSD 4.9 with Dummynet [10] enabled. The receiver side comprises of two clients. The first client is a FreeBSD 4.9 computer connected to the gateway via wired network. The second client is a Compaq laptop with W200D wireless card in a wireless LAN. The wireless Access Point is Cisco Linksys WRT54G. The bandwidth of the wireless network is 11 Mbps and the frequency is 2.4 GHz.

To validate the formula, we let the server continuously send data to the client. The packet size is 1460 byte. For every 4 Mbytes transmitted data, we calculate the average throughput and packet loss rate during this interval. The values of average round-trip time and minimum round-trip time are also traced.
and plugged into Veno’s throughput formula to calculate the theoretical results. For each experiment case, the theoretical results and 20 experiment results are reported.

1) Wired network. The first experiment we considered is TCP Veno transmission over the wired network. The server sends data to the client in the wired network. Dummynet is used to shape the bandwidth to 2 Mbps and the queue size to 10 packets. The traced values of average round-trip time and minimum round-trip time are 0.214 second and 0.165 second, respectively. We plot the theoretical curve and the experiment results in Fig. 9. As can be seen, the theoretical curve accurately predicts the experiment results.

2) Wireless network. Next, we consider TCP Veno transmission over the wireless network. Fig. 10 shows the case that Dummynet shapes the link with 0.01 random packet loss rate, and Fig. 11 shows the case that the client in the wireless LAN. The laptop client moves in the campus so that the random loss rate varies in a larger range. As shown in these figures, the theoretical results are quite accurate in both cases.

3) Co-existence between TCP Veno and TCP Reno flows. Finally, we consider the case of TCP Veno and TCP Reno co-existing in the network. The server opens a TCP Reno connection to send data to the client FreeBSD 4.9. Meanwhile, the server opens a TCP Veno connection to send data to the laptop client in the wireless LAN. As shown in Fig. 12, the matching results also proves the accuracy of TCP Veno throughput formula.

V. CONCLUSIONS

In this paper, we employed fluid-flow model and derived the steady-state throughput formula for TCP Veno. The formula forms the relationship between Veno’s throughput and its given AIMD parameters, the packet loss rate and the network RTT. Simulation results and real network experiments validate the accuracy of our model. In the future, we aim to use TCP Veno throughput formula to improve the equation-based TCP friendly congestion control mechanisms such as TFRC and TFMCC.

REFERENCES