TCPW-F: A Congestion Control Algorithm Based on Noise-aware in Wireless Networks

Fei Zhao1, Yufei Jiang2, Yaming Wang2 and Junyong Tang2,*

1CCCI XI’AN ROAD CONSTRUCTION MACHINERY CO., LTD. Xi’an, China
2School of Computer Science & Engineering, Xi’an Technological University, Xi’an, China

*Corresponding author email: JYTang@xatu.edu.cn

Abstract. Due to the wireless link with more noise packet loss and higher latency as compared with the wired link, conventional TCP Westwood (TCPW) variants over wireless channels perform unsatisfactorily when competing for the shared capacity of the bottleneck link with other variants. The shortcoming is these variants mistakenly interpret noise packet loss as a congestion signal. This leads to frequent delay jitter and long latency due to the inability to distinguish the type of packet loss, which reduces the transmission efficiency of the network. Therefore, we propose a novel TCP variant (TCPW-F) that distinguishes noise loss and congestion loss. By constructing the congestion factor F, the real-time available bandwidth can be calculated. When receiving congestion signals, it ignores the false of them induced by noise packet loss upon the wireless link but responds to true congestion according to backlog-constrained-based judgment. Therefore, TCPW-F is widely used in wireless networks with large noise interference. Numerical analysis and simulation results show that TCPW-F resists noise interference, and smooths the delay jitter per unit time. And the throughput of TCPW-F is significantly higher than other TCPW variants. TCPW-F improves the transmission performance in wireless networks.

Keywords: Backlog-Constrained; Noise packet loss; NS3; Wireless network.

1. Introduction

The increasing interest in wireless networks has heightened the need for maintaining the high throughput and bandwidth of communication when wireless channels with the high bandwidth-delay product (BDP) are more and more prevalent. Recently, the focus of the subject has been to design the appropriate Transmission Congestion Protocol (TCP) variants to increase throughput and performance [1] because the wireless networks are susceptible to noise interference which is mainly derived from electronic waves and full of communication circumstances. The conventional TCP are mainly classified into four categories, i.e. bandwidth-estimation-based, queuing-delay-based, packet-loss-based, and learning-probe-based congestion algorithms. Corresponding to the four categories, the typical variants are TCP Westwood [2], TCP Vegas [3], TCP New Reno [4], and TCP Verus [5], respectively. Although these variants apply different mechanism to control and avoid network congestion, they commonly treat the packet loss as a congestion signal, i.e., the triple duplicative acknowledgements (Acks) to a TCP sender. Compared with wired networks, noise interference can cause frequent packet loss in wireless networks. Because the conventional TCP variants do not distinguish whether the packet loss is derived from noise interference or true congestion, they generally decrease the performance of wireless networks. As a result, wireless links with the BDP tend to lower bandwidth utilization.
Currently, TCP Westwood (TCPW) [6] is the most popular TCP variant for the BDP link. It is widely applied to wireless communication, especially in satellite networks where the transmission is unreliable and the delay time is long. TCPW estimates the effective bandwidth by continuously detecting the arrival rate of the packet Ack. As the congestion signal, i.e., triple duplicative Ack, arrives in a sender, TCPW infers a congestion occurrence in the same way as other TCP variants do. Its mechanism of congestion control, however, adjusts the congestion window (cwnd) value and slow start threshold (ssthresh) according to the bandwidth estimation, rather than directly halving cwnd like loss-based variants doing. So it avoids the excessive penalty of congestion response to improve the transmission rate effectively. For the above reasons, TCPW outperforms other conventional congestion control algorithms in wireless networks.

In recently, some studies combine with other congestion factors to improve the performance of TCPW in wireless networks. RbTCPW [7] adopts pre-estimation of the available buffer between the sender and the receiver. The solution mainly considers that the unacknowledged packets in the buffer of receivers impact the sender’s cwnd. Although it can prolong the congestion avoidance phase for achieving higher throughput, the method is only at the expense of round trip time (rtt) to significantly acerbate the performance of high-speed networks. The literature [8] proposed a nonlinear cwnd growth by a convex curve instead of the additive increasing schedule, the growth that transmits as possible as more packets in the congestion avoiding phase. Similar to TCPW, it does not consider the difference of packet loss, although it uses the ideal growth mode of cwnd. Furthermore, it doesn’t dynamically adjust the sending rate. In [9], the derivation of network congestion is modelled into a fuzzy problem. This schedule uses conditional probability to construct the membership function under different packet-loss modes, and infers the reason for delay according to the maximum match principle. However, when the capacity utilization increases, both the noise and the congestion packet loss increase the transmission delay. Consequently, the packet loss derived from noise affects the accuracy of fuzzy inference due to the packet loss confusion.

It is concluded that popular TCPW variants focus on how to increase cwnd in wireless networks to determine and respond to congestion quickly. However, they all excessively respond to the signal of congestion, i.e. triple duplicate ACKs, no distinguishing whether the packet loss is induced by congestion events or noise interference. According to the literature [10], the noise error rate on the wired links is relatively low, i.e., only 0.012%~1.2%, in contrast to 12% in wireless links. Thus, noise interference mainly causes wireless packet loss. Because current TCPW and its variants don’t identify the spurious congestion events caused by noise, they use the mechanisms of fast retransmission and fast recovery to the packet loss. The avoidance activity triggers the cwnd and its threshold frequently reduction that results in the network throughput under-utilized and the delay jitter increased.

As the above analysis, we found that noise interference causes unnecessary congestion avoidance in wireless networks. Therefore, our research aims to judge the derivation of packet loss by reasonable congestion factor instead of the triple duplicative Ack. Consequently, we propose a congestion control algorithm with a backlog-constrained factor based on TCPW, namely TCPW-F. The algorithm reduces or even stops the cut in cwnd once the triple duplicate Ack are identified as a spurious congestion signal by calculating the factor F in a sender. Therefore, infrequent congestion avoidance improves real-time throughput and cuts back delay jitter.

2. Mechanism of TCPW

Similarly to standard TCP NewReno[11], TCPW algorithm also includes two phases of cwnd increase (slow start phase and congestion avoidance phase) and two congestion control activities (fast retransmission and fast recovery). In the increase phase, TCPW keeps the same response as TCP New Reno. However, it avoids congestion differently when packet loss occurs. In general, the packet loss is considered as a congestion event, which includes the sender receiving triple duplicate Ack and the retransmission timer timeout (RTO). Furthermore, TCPW estimates the real-time bandwidth, \( B_t \), to optimize the ssthresh. The parameters and their setting in TCPW are shown in table 1.
Table 1. TCPW parameter for the cwnd and slow start threshold

| Source            | State         | cwnd   | ssthresh          |
|-------------------|---------------|--------|-------------------|
| Consecutive Dupacks| ssthresh      | (B_k × RTT_{min})/MSS |
| RTO Retransmission| 1            | (B_k × RTT_{min})/MSS |

B_k is estimated using (1):

\[ B_k = \frac{d_k}{(t_k - t_{k-1})} \]  

Where \( t_k \) is the \( k \)-th acquisition time in the ACK arriving in the sender in contrast to the previous acquisition \( t_{k-1} \), and \( d_k \) is the bytes confirmed during the \( k \)-th sampling period. As Table 1 shown, TCPW dynamically adjusts cwnd according to the minimum RTT in a consecutive sampling period. Since noise results in random packet loss and the burst of delay, the RTT_{min} is seldom updated in TCPW. TCPW, therefore, is insensitive to noise interference to a certain extent. Nonetheless, the insensitivity also affects the efficiency of the response to true congestion.

3. Backlog-constrained Congestion Control Algorithm

In this section, we propose a revised TCPW, namely TCPW-F, it overcomes the shortcoming of the existing algorithm. Since the bandwidth utilization is positively correlated with the degree of congestion, we also consider that the packet backlog represents the congestion in a sender, similar to VCP [12]. By constructing backlog factor F, TCPW-F distinguishes whether packet loss is noise loss or not. The backlog factor F is:

\[ F = (RTT_{\text{act}} - RTT_{\text{opt}}) \times C_{\text{act}} \]  

Where \( C_{\text{optimal}} \) is defined as the optimal transmission rate for (3)

\[ C_{\text{optimal}} = \left(\frac{cwnd_i \times MSS}{RTT_{\text{min}}} \right) \]  

Where cwnd_i is the current value of cwnd in the \( i \)-th sampling period, MSS is the maximum message segment. In the case of a sender receiving each of ACKs, it records the minimum RTT for the optimal transmission. So \( C_{\text{optimal}} \) is the highest rate of transmission in the current sampling time. In order to get accurate rtt, the smooth delay (rtt_smt) is defined as in (4)

\[ RTT_{\text{smt}} = \frac{RTT_{\text{all}}}{n} \]  

Where RTT_{all} is the sum of all RTT values in the sampling period. By the mean filtering, RTT is smoothed to alleviate the affection of burst delay from noise. The smooth delay derived from (4), the actual rate of transmission Caactual is calculated in (5).

\[ C_{\text{act}} = \frac{cwnd}{RTT_{\text{smt}}} \]  

When \( C_{\text{act}} > C_{\text{optimal}} \), the F indicates that packets be sent are accumulated in the queue of the sender. It reflects the variation of congestion state as an agile signal of secondary congestion. Whenever \( C_{\text{act}} \) surpasses the benchmark, F is updated on the current value. In figure 1, we propose the congestion threshold value of f [13]. When a congestion signal arrives, TCPW-F contrasts the values between F and f. When F < f, the packet loss is categorized as noise loss. In this condition, TCPW-F unnecessarily re-estimates the bandwidth to set new ssthresh. When F > f, the current links are identified to congestion occurrence. In summary, TCPW-F improves the throughput of congestion avoidance. Its pseudo-code is as follows algorithm 1.

Algorithm 1. TCP-Westwood-Factor

/*When the sender receives an acknowledgement ACK from the receiver*/
TCP-Westwood-Factor (ssthresh, MSS, cwnd, B_k,RTT_{min})
RTT_{smt}=RTT_{all}/n;
/*calculate the current smoothing delay*/
\[
C_{\text{actual}} = \frac{\text{cwnd}}{\text{RTT}_{\text{smi}}}; \\
\text{/* calculate the actual sending rate */}
\]

\[
C_{\text{optimal}} = \frac{\text{cwnd}}{\text{RTT}_{\text{min}}}; \\
\text{/* calculate the ideal sending rate */}
\]

if \(C_{\text{actual}} > C_{\text{optimal}}\) then

/* update the \(\text{RTT}_{\text{min}}\) */

set the current RTT to \(\text{RTT}_{\text{min}}\)

/* construct congestion factor \(F\) and update \(\text{cwnd}\) and \(\text{ssthresh}\) */

else if set \(F = \frac{(\text{RTT}_{\text{smi}} - \text{RTT}_{\text{min}}) \times C_{\text{actual}}}{C_{\text{optimal}}}\) then

if \(F < f\) then

/* update the \(\text{ssthresh}\) and \(\text{cwnd}\) */

\(\text{ssthresh} = \text{cwnd} + F\)

\(\text{cwnd}_{i} = \text{cwnd}_{i-1}\)

else if \(\text{ssthresh} = \left(\frac{B_{i} \times \text{RTT}_{\text{min}}}{\text{MSS}}\right)\) then

/* update \(\text{cwnd}\) in two different situations */

if \(\text{Fast\_Retransmission} = \text{True}\) and \(\text{cwnd} > \text{ssthresh}\) then

\(\text{cwnd} = \text{ssthresh}\)

else \(\text{cwnd}_{i} = \text{cwnd}_{i-1}\)

else if \(\text{Fast\_Retransmission} = \text{True}\) then

set the \(\text{cwnd}\) to 1

end

end

In order to avoid advancing congestion, TCPW-F copes with noise loss and ensure the transmission quality, because the cache is still some packet backlog. Therefore, it adopts a conservative strategy of \(\text{ssthresh}\) reduction, as defined as (6)

\[
\text{ssthresh} = \left(\frac{B_{i} \times \text{RTT}_{\text{min}}}{\text{MSS}}\right)
\]

By comparing TCPW-F and TCPW upon the congestion response, the reduction of \(\text{cwnd}\) is shown in figure 1.

![Figure 1. Comparison of window growth stages](image)

4. Performance Evaluation

In this section, we conduct simulation experiments to evaluate the performance of TCPW-F by using Network Simulator [14-15] version 3.29. In the testbed, some senders and a receiver (n0) connect to each other, and the other nodes generate background flows that communicate with two intermediate routers, i.e., n1 and n2, as wireless bridges. Moreover, the bridge n1 connects to the fixed host n0 upon a point-to-point wired network, and the bridges are attached to all clients by utilizing the two-way wireless channel. The wired trunk is a high-speed link with 100Mbps, and the other wireless channels are 5ms delay and 10Mbps bandwidth. The testbed is a dumbbell topology, as illustrated in figure 2.
In figure 3, the $cwnd$ of TCPW-F is significantly higher than TCPW. The reasons are two-fold. First, the TCPW algorithm frequently reduces $cwnd$ when packet loss occurs. But the TCPW-F keeps the window value growing in the same case. Another reason is that the TCPW algorithm directly reduces the window to an approximated minimum for responding to a false congestion event, whereas TCPW-F decreases to a suitable value and then increases. Compared with the original algorithm, TCPW-F can distinguish the packet loss type according to the congestion factor F. When noise packet loss is detected, it can set the appropriate new window parameters and keep the window growing.

Figure 4 and 5 show the delay and jitter of TCPW, TCPW-F and RbTCPW. Compared with the other two algorithms, TCPW-F has lower latency, less jitter and faster response time. As shown in figure 5, the jitter of TCPW-F does not severely oscillate, because TCPW-F effectively distinguishes the noise packet loss. It not only quickly increases the transmission rate, but also decreases the excessive requirement of packets queue in nodes. So the performance of TCPW-F is the best in delay-jitter sensitive networks.

Figure 6 shows the throughputs of TCPW-F compared to TCPW and RbTCPW in a scenario which datarate is 1Mbps. The throughput of the three algorithms is nearly close to 1.2 Mbps. However, the rate of packets (10Mbps) is equal to the channel bandwidth so that the congestion and noise loss occur simultaneously, thereby reducing throughput. As shown in figure 7, Whereas the real-time throughput can not reach the maximum value, the throughput growth of TCPW-F is significantly higher than the other algorithms. TCPW-F significantly enhances the competition with other TCP variants in wireless networks.
5. Conclusion
In this paper, we propose a novel TCPW variant, namely TCPW-F. It is an effective congestion control algorithm that distinguishes the type of packet loss in a wireless network. Through theoretical analysis and experimental verification, we find that the current TCPW algorithm based on the bandwidth estimation should be improved, because it incorrectly responds to noisy packet loss, which leads to excessive delay. Therefore, we design an identification mechanism (congestion factor F), which distinguishes between noise loss and congestion loss. When the sender receives a regular congestion signal (triple duplicative Acks), this algorithm resets the new cwnd. By simulating experiments in NS3, the results show that the TCPW-F algorithm improves performance in terms of delay jitter and throughput, and effectively competes with other congestion algorithms.

References
[1] Cheng Jiyu, Cross Layer Optimization of TCP Transmission in Wireless Networks[D]. Beijing University of Posts and Telecommunications,2018.
[2] Casetti C E. TCP Westwood: End-to-end Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks[J]. Acm Mobicom, 2001.
[3] Brakmo L S, Peterson L L. TCP Vegas: End to End Congestion Avoidance on a Global Internet[J]. IEEE Journal on Selected Areas in Communication, 1995, 13(8):P.1465-1480.
[4] Nishida Y. The NewReno modification to TCP’s fast recovery algorithm[J]. Standards Track, PP, 2012: 1-16.
[5] Zaki Y, Potsch T, Chen J, et al. Adaptive Congestion Control for Unpredictable Cellular Networks[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(5): 509-522.
[6] Purvang, Mohanchur S, Nikhil K, et al. Refining TCP's RTT dependent mechanism by utilizing link retransmission delay measurement in Wireless LAN[J]. International Journal of Communication Systems, 2017, 30(5):e3007.1-e3007.20.
[7] Hao J, Feng H, Wu G. Improving Data Transmission Reliability with Network Coding in Wireless Sensor Network[C]// Multimedia Information Networking and Security (MINES), 2010 International Conference on. IEEE Computer Society, 2010.
[8] Tropea M, Fazio P. Evaluation of TCP versions over GEO satellite links[C]// International Symposium on Performance Evaluation of Computer & Telecommunication Systems. IEEE, 2013.
[9] Wang R, Aryasomayajula N C, Ayyagari A, et al. An Experimental Performance Evaluation of SCPS-TP over Cislunar Communications Links[C]// Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE. IEEE, 2007.
[10] Mascolo S. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links[J]. Proc. ACM MOBICOM’2001, Jul, 2001.
[11] Nishida Y. The NewReno modification to TCP’s fast recovery algorithm[J]. Standards Track, PP, 2012: 1-16.
[12] Yun-Long Z, Yan-Long M A, Xiang L I. A link-independence multi-path routing strategy in wireless Mesh networks[J]. Journal of Harbin Engineering University, 2008.
[13] Casetti C, Gerla M, Mascolo S, et al. TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks[J]. Wireless Networks, 2002, 8(5):467-479.
[14] Casoni M, Grazia C A, Klapez M, et al. Implementation and validation of TCP options and congestion control algorithms for ns-3[C]//Proceedings of the 2015 Workshop on Ns-3. 2015: 112-119.
[15] Coudron M, Secci S. An implementation of multipath TCP in ns3[J]. Computer Networks, 2017, 116: 1~11