Stable Accurate Rapid Bandwidth Estimate for Improving TCP over Wireless Networks*

Le Tuan Anh and Choong Seon Hong

Computer Engineering Department, Kyung Hee Univerity,
1, Seocheon, Giheung, Yongin, Gyeonggi 449-701, Korea
letuanh@networking.khu.ac.kr
cshong@khu.ac.kr

Abstract. This paper presents a stable accurate rapid bandwidth estimate (SARBE) algorithm to improve TCP performance over wireless networks. The proposed algorithm estimates the bandwidth sample of the forward path of connection by monitoring the sending time intervals of ACKs. By using the stability-based filter, the current bandwidth samples are eliminated from transient changes, while keeping reasonably persistent changes in the absence of noise. TCP congestion control then uses the estimated bandwidth to properly set the slow start threshold (*ssthresh*) and congestion window size (*cwnd*) rather than halving the current value of the *cwnd* as in TCP after fast retransmit. In the wireless environment, SARBE achieves robustness in aspect of stability, accuracy and rapidity of the estimate, and better performance compared with TCP Reno, Westwood and DynaPara. Furthermore, SARBE is fair in bottleneck sharing and friendly to existing TCP versions, and also tolerates ACK compression in the backward path.

1 Introduction

TCP was originally developed for the wired networks, where bit-error rate is trivial and packet losses are caused by network congestion. However, cellular/wireless networks and wired-wireless networks issue performance degradation challenges to TCP because TCP congestion control cannot distinguish between the packet losses caused by the random error of the wireless link, signal fading and mobile handoff processing, and those caused by network congestion.

At the TCP sender, there are two state variables, the congestion window size and the slow start threshold are maintained by congestion control to the control the transmission rate. In the slow start phase, at beginning of connection, *cwnd* is increased by 1 for every arrived ACK at the sender. Until *cwnd* reaches *ssthresh*, the sender TCP congestion control goes to the congestion avoidance phase, where *cwnd* is increased by $1/cwnd$ for every arrived ACK at the sender.

* This work is supported by University ITRC of MIC. Dr. Hong is corresponding author.

P. Lorenz and P. Dini (Eds.): ICN 2005, LNCS 3421, pp. 141–148, 2005.

cwnd is additively increased to reach the network bandwidth and is abundantly decreased to one half of current value when the condition of network congestion occurs by indicating received Duplicate ACKs at the sender. TCP Reno then sets *ssthresh* to be the same as *cwnd*. If the packet losses occur by the random errors before *ssthresh* reaches the network capacity, *ssthresh* can be gotten a smaller value, thus blindly reducing the sending rate (i.e degrading TCP performance).

To improve TCP performance over wireless networks, several approaches have been proposed and [2] classified into three classes: the link-layer approach, which improves wireless link characteristics or hiding non-congestion caused packet losses from TCP; the split-connection approach, in which the sender have to be aware of the existence of wireless hop; the end-to-end approach, which retains TCP semantics, but requires to improve the protocol stack at either the sender side or the receiver side.

In this paper, we proposed SARBE algorithm for improving TCP over wireless networks. SARBE achieves robustness in aspect of stability, accuracy and rapidity of the estimate, and better performance compared with TCP Reno, Westwood and DynaPara. Furthermore, SARBE is fair in bottleneck sharing and friendly to existing TCP versions, and also tolerates ACK compression in the backward path.

The rest of this paper is organized as follows. Section 2 summarizes the previous works in improving the TCP performance for wired-wireless networks. Section 3 presents our proposed SARBE algorithm in detail. Various simulation results presented in section 4. Finally, section 5 concludes the paper.

2 Related Works

The end-to-end approach improves TCP performance at either the sender or receiver without violating end-to-end semantic. TCP SACK [9] (option in TCP header) improves retransmission of lost packets using the selective ACKs provided by the TCP receiver. In Freeze-TCP [5], before occurring handoff, the receiver sends Zero Window Advertisement (ZWA) to force the sender into the Zero Window Probe (ZWP) mode and prevent it from dropping its congestion window. Freeze-TCP can only improve TCP performance in handoff cases.

TCP Westwood scheme [3], [4], and TCP scheme with dynamic parameter adjustment (DynaPara) [7] are the end-to-end approach. In these schemes, the sender estimates available bandwidth dynamically by monitoring and averaging the rate of ACKs receiving. The sender then updates *cwnd* and *ssthresh* to the estimated bandwidth when fast retransmit or retransmission timeout occurs. DynaPara used a simple filter with dynamic adjusting of a filter parameter according to the sate of wireless link. Although, the filter of TCP Westwood is complex, it cannot reflect the rapid changes of the network condition, whereas in DynaPara algorithm, the estimated bandwidth fluctuates frequently. In addition, when the ACK packets encounter queuing with cross traffic along the backward path, their time spacing is no longer the transmission time upon leaving the queue. These time spacing may be shorter than original time spacing, called

ACK compression [10]. In this case, both schemes overestimate the available bandwidth.

3 SARBE Algorithm

3.1 Available Bandwidth Estimate

In comparison with TCP Westwood, we take the advantage of using ACKs sending time interval to achieve a more accurate available bandwidth estimate. SARBE employs the ACKs sending time intervals to compute the available bandwidth of the forward path via the timestamp in ACK. In SARBE approach, the estimate of the forward path is not be affected by ACK compression that results in overestimate.

When the kth ACK arrives, the sender simply uses information of the kth ACK to compute an available bandwidth sample, which can be written as

$$Bw_k = \frac{L_k}{ts_k - ts_{k-1}} \tag{1}$$

where L_k is the amount of data acknowledged by the kth ACK, ts_k is timestamp of the kth ACK; ts_{k-1} is the timestamp of the previous ACK arrived at the sender. It can be seen obviously, sample Bw_k represents the current network condition, which faces noises. So the bandwidth estimator has to eliminate transient noise but responds rapidly to persistent changes.

We used the stability-based filter [6] which is similar to the EWMA filter, except using a measure function of the samples' large variance to dynamically change the gain in the EWMA filter. After computing the bandwidth sample Bw_k from (1), the stability-based filter can be expressed in the recursive form

$$U_k = \beta U_{k-1} + (1-\beta) \mid Bw_k - Bw_{k-1} \mid \tag{2}$$

$$U_{max} = max(U_{k-N}, ..., U_{k-1}, U_k) \tag{3}$$

$$\alpha = \frac{U_k}{U_{max}} \tag{4}$$

$$eBw_k = \alpha \cdot eBw_{k-1} + (1-\alpha)Bw_k \tag{5}$$

where U_k is the network instability computed in (2) by EWMA filter with gain β, β was found to be 0.8 in our simulations; $Umax$ is the largest network instability observed among the last N instabilities ($N = 8$ in our simulations); and eBw_k is the estimated smoothed bandwidth, eBw_{k-1} is the previous estimate and the gain α is computed as (4) when the bandwidth samples vary largely.

3.2 TCP Congestion Control Modification Algorithm

As mentioned in Section 1, *ssthresh* represents the probed network bandwidth; while the above estimated bandwidth value also represents the current available bandwidth of download link. Consequently, we have to transform the estimated

value into equivalent size of the congestion window for updating *ssthresh*. [4] proposed the interrelation of estimated bandwidth with the optimal congestion window size (*oCwnd*) as

$$oCwnd = \frac{eBw \cdot RTT_{min}}{Seg\_size} \tag{6}$$

where RTT_{min} is the lowest Round Trip Time, *Seg_size* is the length of the TCP segment.

The pseudo code of the the congestion control algorithm for updating *ssthresh* and *cwnd* is the following:

```
if (Duplicate ACKs are detected)
    ssthresh = oCwnd;
    if (cwnd >= ssthresh)
        /* in the congestion avoidance phase*/
        cwnd = ssthresh;
    else /* in the slow start phase */
        cwnd = cwnd; /*keeping cwnd*/
end if

if ( retransmission timeout expires)
    ssthresh = oCwnd;
    cwnd = 1;
/* restarting from the slow start phase */
end if
```

Whenever the sender detects Duplicate ACKs representing that the sent packets were lost due to the light network congestion or the random error of the wireless link, the congestion control updates *ssthresh* to the optimal congestion window (*oCwnd*); and sets *cwnd* to *ssthresh* during the congestion avoidance phase, or keeps the current *cwnd* value during the slow start phase.

If the sender is triggered by the retransmission timeout event due to the heavy network congestion or the very high bit-error rate of wireless link, its congestion control sets *ssthresh* to the optimal congestion window, and sets *cwnd* to one for restarting from the slow start phase.

4 Simulation Results

Our simulations were run by the NS-2 simulation network tool. We used the recent Westwood module NS-2 [8] for comparison.

4.1 Effectiveness

We first evaluate the stability, accurateness and rapidity of SARBE. The simulation network scenario is depicted in Fig. 1(a). We used an FTP over TCP and an UDP-based CBR background load with the same packet size of 1000 bytes. The CBR rate varies according to time as the dotted line in Fig 1(b).

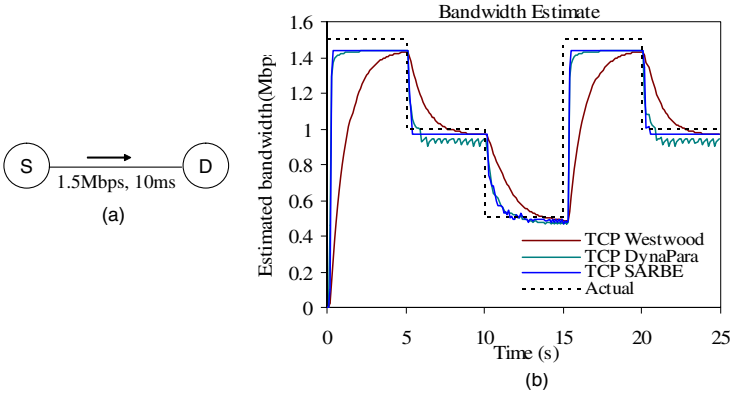

Fig. 1. (a) Single bottleneck link; (b) Comparison of Bandwidth estimate algorithms

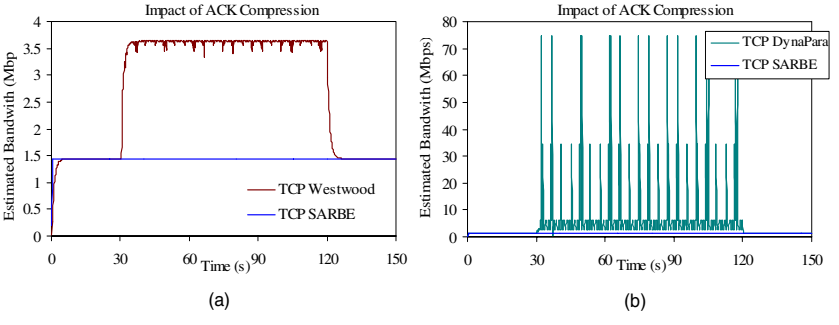

Fig. 2. The overestimated bandwidth of TCP Westwood (a) and TCP DynaPara (b) in the presence of the ACK compression

The result is shown in Fig. 1(b); TCP Westwood is very slow to obtain the available bandwidth changes, while DynaPara estimates fluctuantly and inaccurately when the bottleneck bandwidth is occupied by CBR. By contrast, SARBE reaches the persistent bandwidth changes rapidly, which closely follow the available bandwidth changes. This is due to adaptability of dynamic changes of gain α when the bandwidth samples vary largely.

To investigate the impact of ACK compression on estimate, we used the network scenario as Fig. 1(a) and supplemented a traffic load FTP in the reverse direction. The traffic load FTP was started at time 30 s and ended at 120 s for 150 s simulation time. In this interval, Westwood estimates over 2 Mbps more than SARBE, which is quite near the actual available bandwidth, as in Fig 2(a). In DynaPara, the estimated values fluctuate and are many times more than SARBE, shown in Fig 2(b).

Next, we evaluate TCP performance in the scenario as given in Fig. 3. The simulation was performed on one FTP in 100 s with the packet size of 1000 bytes, the wireless link random errors ranging from 0.001% to 10% packet loss. In Fig

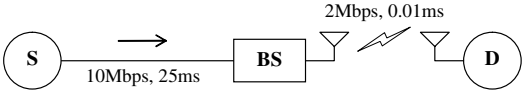

Fig. 3. The wired-wireless network scenario

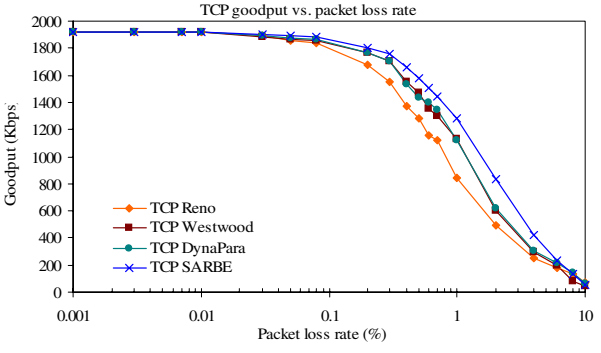

Fig. 4. TCP goodput vs. packet loss rate

4, for the random error rate lower than 0.01%, the goodput of all TCP versions is same. Over that error rate, SARBE' goodput is better than other versions. For example, at 1% packet loss rate, SARBE achieves better performance than TCP Reno, Westwood and DynaPara by 34.6%, 14% and 13%, respectively.

4.2 Fairness

Another evaluation for TCP is fairness that a set of connections of the same TCP, which can share fairly the bottleneck bandwidth. The index of fairness was defined in [1] as

$$fi = \frac{(\sum_{i=1}^{n} x_i)^2}{n(\sum_{i=1}^{n} x_i^2)}, \quad 1/n \leq fi \leq n \tag{7}$$

where x_i is the throughput of the ith TCP connection, n is the number TCP connections considered in simulation. The fairness index has a range from $1/n$ to 1.0, with 1.0 indicating fair bandwidth allocation. Using the same scenario as Fig. 3 with ten same TCP connections, we simulated the different TCP versions individually. The buffer capacity of bottleneck link is equal to the pipe size. The comparison result is shown in Fig. 5. TCP Reno is always manifested the best fair sharing, the second is SARBE, which achieves quite high fairness index.

4.3 Friendliness

The friendliness of TCP implies fair bandwidth sharing with the existing TCP versions. Our experiments were run on the scenario of Fig. 3. We considered a total of ten connections mixing SARBE with TCP Reno, Westwood and DynaPara

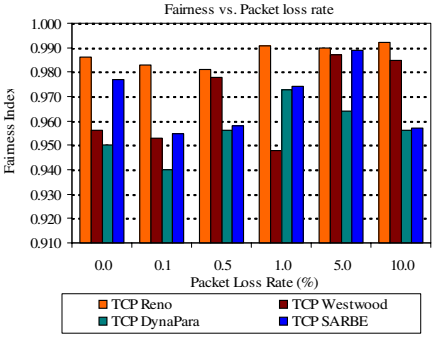

Fig. 5. Fairness vs. packet loss rate

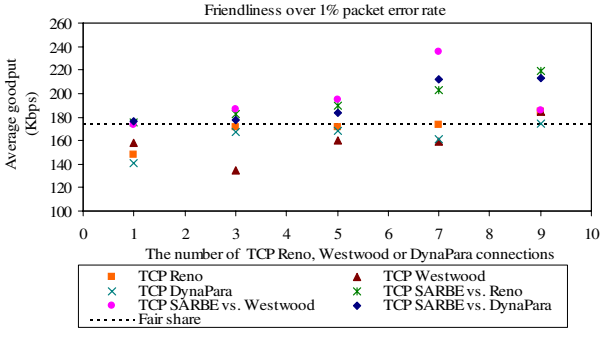

Fig. 6. Friendliness of TCP Reno, Westwood and DynaPara compared with SARBE, respectively, over 1% packet loss link

at 1% packet loss rate. The x-axis of Fig. 6 represents the number of TCP Reno, Westwood or DynaPara connections, the remaining connections use SARBE. The dotted line is the fair share. In Fig. 6, SARBE still preserves friendliness with the existing TCP versions, but outdoes in goodput. This result accords with the above performance evaluation result with the presence of 1% packet loss rate.

5 Conclusions

This paper has presented a novel approach in the available bandwidth estimate. Firstly, estimating the forward path of connection provides the true available bandwidth of download link and tolerates ACK compression. Secondly, applying the stability-based filter can resist transient changes, while keeping reasonably persistent changes of bandwidth samples. Thirdly, the modified congestion control algorithm sets *ssthresh* to the optimal congestion window size upon fast retransmit or retransmission timeout events.

Simulation results have shown that, our algorithm achieves robustness in aspect of stability, accuracy and rapidity of the estimate, and better performance in comparison with TCP Reno, Westwood and DynaPara. As a final point, SARBE is fair in bottleneck sharing and friendly to the existing TCP versions.

References

1. R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC, Rep.TR-301, 1984.
2. H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," IEEE/ACM Trans. Networking, vol. 5, no. 6, pp. 756769, 1997.
3. S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi,and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in Proc. ACM MobiCom 2001, Roma, Italy, pp. 287-297, July 2001.
4. S. Mascolo, C. Casetti, M. Gerla, and S.S. Lee, M. Sanadidi, "TCP Westwood: Congestion Control with Faster Recovery," UCLA CS Technical Report, 2000.
5. T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in Proc. IEEE INFOCOM 2000, Tel-Aviv, Israel, pp. 1537-1545, Mar. 2000.
6. M. Kim and B. D. Noble, "SANE: stable agile network estimation," Technical Report CSE-TR-432-00, University of Michigan, Department of Electrical Engineering and Computer Science, Ann Arbor, MI, August 2000.
7. N. Itaya, S. Kasahara, "Dynamic parameter adjustment for available-bandwidth estimation of TCP in wired-wireless netwoks," Elsevier Com. Comm.27 976-988, 2004.
8. TCP Westwood - Modules for NS-2 [Online]. Available: http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.html, 2004.
9. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," RFC 2018, Oct. 1996.
10. L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic," Proc. SIGCOMM Symp. Comm. Architectures and Protocols, pp. 133-147, Sept. 1991.