

A Flow Control Scheme based on Buffer State for Wireless TCP

Jin-Hee Choi, See-Hwan Yoo and Chuck Yoo

Abstract— Reliable transport protocols such as TCP (Transmission Control Protocol) are tuned to perform well in traditional networks where packet losses occur mostly because of congestion. However, on the wireless links where packet losses frequently happen, TCP shows a serious drop in performance. In this paper, we propose a scheme, called BSF (Buffer State based Flow control), which can solve this problem. BSF is implemented at the base station of a wireless network. So, it does not need any modification of TCP stacks in the end system. BSF uses the buffer state of base station as the criterion to detect wireless link error and reduces unnecessary contraction of congestion window in TCP. As a result, the buffer of base station maintains non-empty state to maximize the link utilization. In our simulation study, this approach achieved up to 250 % improvement in performance.

Keywords— TCP, heterogeneous network, flow control, performance evaluation, RED, wireless network

I. INTRODUCTION

Recently, the wireless network comes to be popular. So, it is expected to play an important role in the future internetworking. There are so many existing applications using TCP/IP, their use over wireless networks is a certainty. Thus, the performance of the TCP/IP over wireless networks is an important issue. TCP was designed for being used on stable link, where link errors do not occur frequently. TCP assume congestion in the network to be primary cause for packet losses and unusual delays. TCP shows a good performance in the wired environment by adapting end-to-end delays and packet losses caused by congestion. However, in the wireless environment, burst link error tends to occur more frequently than that of wired link. TCP sender misunderstands packet loss from this unstable link as packet loss from congestion in the network. This congestion control that results from link error reduces congestion window unnecessarily[1]. This results in a serious drop in TCP's performance.

If the reason of the packet loss is known, we can prepare for proper actions. However, it is so hard to know the reason of the packet loss in the end system. In addition, any modification of the TCP stack makes practical deployment difficult. We believe that more effective approach is inferring the reason of packet loss without any modification of TCP stack at the base station. We propose a new mechanism to improve performance in TCP, which performs a flow control based on buffer state at the base station.

Authors are with the Department of Computer Science and Engineering, Korea University, Korea (Phone: +82-2-3290-3639, fax: +82-2-922-6341, e-mail: {jhchoi, shyoo, hxy}@os.korea.ac.kr).

The rest of this paper is organized as follows. Section 2 describes the existing works to improve performance in wireless TCP. Section 3 gives a detailed explanation of Buffer State based Flow control scheme. Section 4 provides the results of simulation and its analysis. Finally, in section 5, we conclude the paper.

II. RELATED WORKS

A. Split connection scheme

The split connection[2], [3], [4] is an early form to improve TCP's performance in wireless networks. The aim of these protocols is separating packet losses resulting from the error of wireless link from the network congestion in wired network. So, these protocols split the connection of TCP at base station: from TCP sender to the base station and from the base station to TCP receiver. I-TCP is a representative protocol of split connection.

I-TCP has been proposed by Bakre et al.[2], [3]. I-TCP attempt to prevent TCP sender from misbehave isolating wired and wireless related problem in heterogeneous network. However, in this scheme, since the TCP stack of the base station directly send ACKs to TCP sender, the TCP sender does not know whether real TCP receiver gets the packet or not. In this case, ACK only confirms that the TCP stack at the base station receives the packet. In other words, the end-to-end semantics of TCP is destroyed. Also, when a packet arrives at the base station, it is processed twice. Such processing overhead in the base station would lead to reduce TCP's performance.

B. End-to-end scheme

There are many schemes in the end-to-end scheme, which are snoop protocol[5], Freeze TCP[6], TCP Westwood[7] etc.

Balakrishnan et al.[5] proposed a scheme called Snoop. In this scheme, there exists a snoop agent at the base station. The snoop agent monitors every packet that passes on the base station and caches TCP segments. When a packet loss occurs, the cached packets in the base station are retransmitted. Also, the snoop protocol suppresses duplicated ACKs, and this leads to the reduction of the number of duplicated ACKs at the TCP sender. As a result, such local retransmission and the suppression of duplicated ACKs let TCP achieve good improvement in performance in wireless network. However, the snoop protocol has a weakness, which does not cope effectively with network congestion. So, Hu et al.[8], [9] proposed a scheme called

FDA, in order to solve this problem.

Goff et al.[6] proposed a scheme called Freeze TCP. This scheme achieves improvement in TCP's performance, only with modification of TCP receiver's protocol stack. In this scheme, when TCP receiver discovers that the connection is unstable or seems to be broken, send out ZWA(Zero Window Advertisement) packets to TCP sender. On receiving ZWA packets, the TCP sender get into persist mode, and this makes TCP's all states suspended. The result is that Freeze TCP prevent TCP from misbehave with a little modification of the protocol.

Mascolo et al.[7] proposed a scheme called TCP Westwood. This scheme is a sender-side modification of the TCP congestion window algorithm that improves the performance of TCP in heterogeneous network. In this scheme, TCP sender continuously measures the rate of incoming ACKs, and infers the state of network based on this information. Being based on this estimation, TCP Westwood compute congestion window size and slow start threshold after a congestion episode, that is, after receiving three duplicated ACKs or after retransmission timer expired. This algorithm resulted in a faster recovery from congestion episode than TCP Reno, and achieved good improvement in performance.

III. BUFFER STATE BASED FLOW CONTROL

A. Variation of the buffer state

On a wireless link, bit errors tend to be bursty. Therefore, as the link error rate increases, packets in the same TCP window are contiguously lost, which leads TCP to shrink the window size. Then, the base station has no packets to queue, which results in a waste of the bandwidth. Figure 1 shows the buffer length in a base station when the error rate of wireless link is zero. Even though it shows the fluctuation of buffer length, empty buffer is not observed except for the beginning of packet transmission. When the packet transmission begins, there are not enough queued packets in the buffer of the base station. On the other hand, when the error rate is three percents, Figure 2 shows that the base station becomes an idle buffer state (i.e. when buffer length in figure is zero) frequently. Fig. 2 also shows that the average queue size is less than that in Fig. 1. As the less packets are queued in the base station, the link utilization gets lower. Therefore, it is clear that preventing the buffer in a base station from running underflow is also important issue as much as preventing overflow(overflow means congestion).

The above figures also indicate that the empty buffer state of base station has a close relationship with the error rate of wireless link although it does not always mean bit errors. A key idea in this paper is that we can use the buffer state of base station as an indicator to determine whether packet loss is due to wireless link error or network congestion.

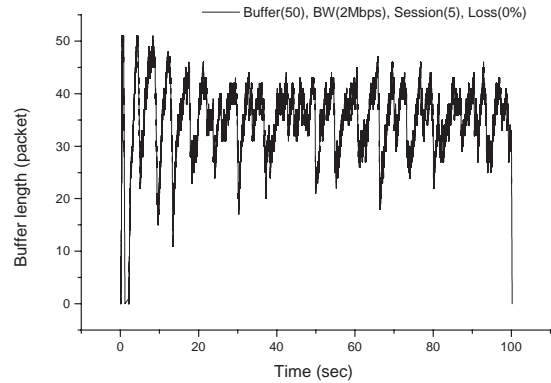


Fig. 1. Variation of buffer length based on wireless error rate(0%).

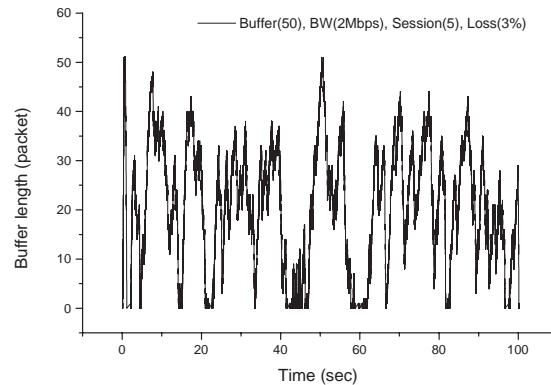


Fig. 2. Variation of buffer length based on wireless error rate(3%).

B. Overview of BSF

We propose a mechanism called Buffer State based Flow control (BSF), which prevents the buffer of base station from being empty by wireless link errors. BSF separates TCP and other protocols and put the packets into two different queues called TCP buffer and UDP buffer respectively as shown in 3. The whole idea of BSF is based on inferring the timing when burst packet losses occur. The pseudo code in Figure 4 summarizes the scheme.

BSF consists of two functions: monitoring to detect burst packet losses and freezing TCP's all states. BSF monitors the average buffer length of TCP buffer. The average buffer length(avg) is calculated as follows.

$$avg = (1 - w)avg + wq, \quad (1)$$

where w is the low pass filter's weight, and q is the instantaneous buffer length at the time of calculation.

When the average buffer length gets smaller than a threshold, BSF_{thresh} , BSF is initiated. BSF collects the following information from each incoming packet: flow id, the most recent TCP sequence number, IP addresses of

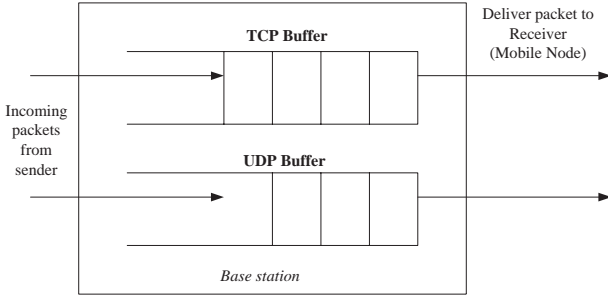


Fig. 3. Queue of BSF.

source and destination node, and ECN (Explicit Congestion Notification) bit history. Flow id is used to identify each session. TCP sequence number is for sending an ACK, which is used to freeze TCP's all state in the sender. IP address is to construct a packet header of ACK. ECN[13], [14] bit history is used to distinguish network congestion from the error of wireless link. Even if the average buffer length gets smaller than a BSF_{thresh} , a session with recent n ECN bit (we choose a heuristic value, 3) is excluded from the target of flow control.

When the average buffer length gets smaller than a threshold of $FLUSH_{thresh}$, BSF performs a flow control based on the collected information. Since the objective of flow control in this case is freezing TCP's all state, we choose to use the persist mode of TCP.

-
1. **FOR** (each packet arrival)
 2. $avg = (1 - w)avg + wq$
 3. **IF** ($avg < BSF_{thresh}$)
 4. **IF**($avg < FLUSH_{thresh}$) **THEN**
 5. flush flow information
 6. force to send ZWA packet
 7. **ELSE**
 8. collect flow information
 9. **END**
-

Fig. 4. BSF pseudo code

C. Policy on thresholds selection

In previous section, we introduce two thresholds: BSF_{thresh} and $FLUSH_{thresh}$. BSF_{thresh} determines the time of information collection:

$$TCP_{buff} = TOTAL_{buff} - UDP_{buff}, \quad (2)$$

$$BSF_{thresh} = N + \left\lceil \frac{TCP_{buff}}{N} \right\rceil, \quad (3)$$

where N is the number of TCP sessions ($N > 5$).

The information of sessions is collected between BSF_{thresh} and $FLUSH_{thresh}$. $FLUSH_{thresh}$ decides the

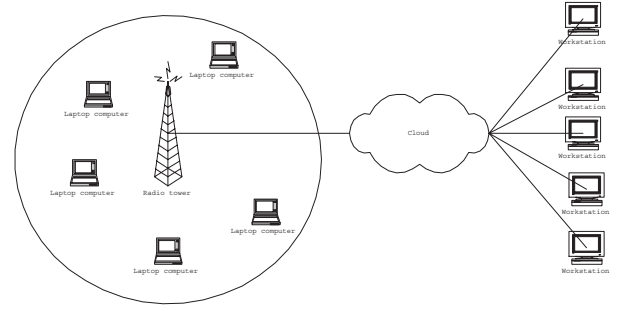


Fig. 5. Simulation topology.

time to flush the session information and the time to freeze TCP's all states:

$$FLUSH_{thresh} = \left\lfloor N - \left\lceil \frac{TCP_{buff}}{N} \right\rceil \right\rfloor, \quad (4)$$

All thresholds should be chosen to satisfy following relationship.

$$FLUSH_{thresh} < BSF_{thresh} < TCP_{buff}. \quad (5)$$

D. Flow control

If the average buffer length gets smaller than $FLUSH_{thresh}$, BSF tries to freeze the TCP window because the network is under-utilized. Without modification of TCP stack, the best way to freeze TCP's state is to use the TCP's persist mode. So BSF sends ZWA to the sender TCP. Sessions that recently received packets have the ECN bit in their headers are excluded from targets; this inference would be more accurate. Once the sender in persist mode, it regularly probes the receiver's buffer size. If the receiver's buffer size is zero, the TCP sender remains in the persist mode. In the persist mode, TCP sender does not shrink congestion window. As a result, TCP retains the window size as before the persist mode and BSF reduces the empty state of TCP buffer, and therefore the utilization of wireless link increases. When the receiver buffer size is non-zero, the TCP sender exits from the persist mode and resumes the transmission.

IV. SIMULATION AND ANALYSIS

In this section, we evaluate the performance of TCP with BSF and without BSF by simulations. The simulation study is done with the NS2 network simulator[16]. NS-2 is a discrete event simulator that was developed as part of the VINT project at the Lawrence Berkeley National Laboratory. NS-2 was extended by the CMU Monarch project[17] to simulate mobile nodes connected by wireless network interfaces. We used CMU's NS-2 and implemented BSF in the NS-2 RED[18] protocol implementation. The TCP version is TCP Reno, and wireless link is modeled on two states Markov chain[19]. Figure 5 shows our simulation topology. In order to simulate the future wireless network, we set the bandwidth of wireless link to 2Mbps and wired link to 10Mbps. Also, for purpose of more accurate result,

we simulated all experiments for 200,000 seconds.

Our main performance metric is the total throughput of TCP sessions. Its definition is:

$$Throughput = \sum_{i=1}^n \frac{total_packet_i \times packet_size}{simulation_time} \quad (6)$$

where n is the number of session, i is session identifier. The throughput is measured by varying the error rate from 1% to 10%.

Figure 6 shows a comparison of TCP performance with BSF and without BSF. We can see that the performance drops dramatically as the error rate increases without BSF. The reason is, as the error rate increases, multiple packet loss also increases. Multiple packet losses causes TCP to wait for retransmission timer to be expired, and packets cannot be exchanged for a significant amount of time, which is RTO (Retransmission Time Out) value. After recovering from the timeout, TCP must do slow start. If all sessions perform slow start concurrently, recovery could be delayed. This situation would lead to buffer overflow. BSF prevents the buffer of base station from being empty, and so it reduces the probability by which multiple concurrent slow starts occur.

Figure 7 shows the impact of the number of sessions to BSF. As the number of sessions in a cell increases (5, 10, 15 sessions respectively), the number of packets queued in the buffer also increases. Therefore, it shows the highest throughput in simulation when 15 sessions are in a cell. However, the throughput per session and the impact of BSF is reduced. It is because the timing to detect $FLUSH_{thresh}$ is affected as the number of sessions increases. It is important issue to determine $FLUSH_{thresh}$. Too high $FLUSH_{thresh}$ could lead BSF to perform unnecessarily. The reverse, too low $FLUSH_{thresh}$ could lead BSF to perform too late. Although we propose a simple scheme in equation (4), it would not be best estimation in any case. However, In spite of such factors, BSF still shows an outstanding improvement in the total throughput, as error rate increases.

Figure 8 shows the ratio of RED only to BSF in performance, varying the number of TCP session in a cell. When the number of session is 5 and error rate is 8%, BSF achieves 250% improvement in performance.

V. CONCLUSION

This paper proposes a new flow control scheme named BSF to improve the TCP performance in wireless networks. The proposed scheme uses the buffer state of base station as criterion to detect wireless link error. It can be easily integrated with networks already established, because it does not require any modification of TCP stacks in end systems. In our simulation study, this approach achieves up to 250 % improvement in performance.

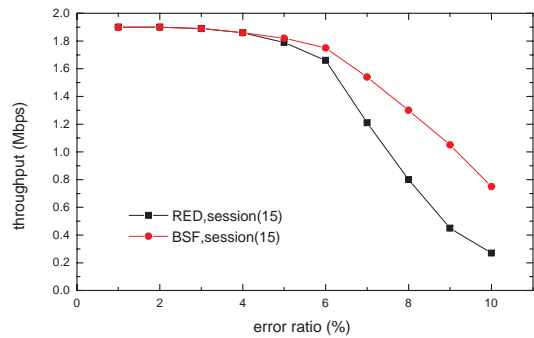


Fig. 6. Comparison of TCP performance based on flow control scheme (RED only and BSF).

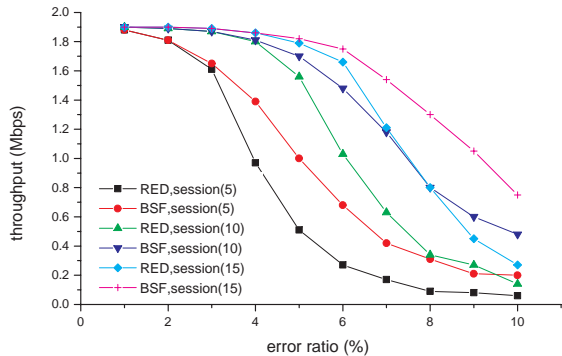


Fig. 7. TCP performance depending on the number of TCP session in a cell.

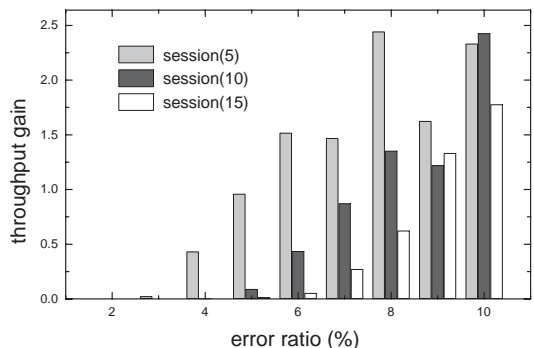


Fig. 8. The ratio of RED only to BSF in performance.

In the future wireless network, it is expected that both the cell size and the number of sessions in a cell are reduced to provide services requiring high bandwidth. When there are less sessions in a cell, the buffer state is more affected by the error of wireless link. So, we believe that BSF would become more reasonable solution in the future.

REFERENCES

- [1] G. Xylomenos, G.C. Polyzos and M. Saaranen , "TCP Performance Issues over Wireless Links," *IEEE Communications Magazine*, April 2001.
- [2] A.V. Bakre and B.R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proc. IEEE ICDCS'95*, 136-143, 1995.
- [3] A.V. Bakre and B.R. Badrinath, "Implementaion and Performance Evaluation of Indirect TCP," *IEEE Transactions on Computer*, VOL.46, NO.3, March 1997.
- [4] Z.J. Haas, "Mobile-TCP: an asymmetric transport protocol design for mobile systems," in *Proc. IEEE ICC'97*, 1054-1058, 1997.
- [5] H. Balakrishnan, V.N. Padmanabhan, S. Seshan and R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," in *Proc. ACM SIGCOMM'96*, August 1996.
- [6] T. Goff, J. Moronski, D.S. Phatak, V. Gupta, "Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments," in *Proc. IEEE 19th InfoCom'2000*, 1537-1545, 2000.
- [7] 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*, July, 2001.
- [8] J.H. Hu and K.L. Yeung, "FDA: A Novel Base Station Flow Control Scheme for TCP over Heterogeneous Networks," in *Proc. IEEE 20th InfoCom'2001*, April 2001.
- [9] J.H. Hu and K.L. Yeung, "Hierarchical Cache Design for Enhancing TCP over Heterogeneous Networks with Wired and Wireless Links," in *Proc. IEEE GLOBECOM'2000*, November 2000.
- [10] J. Borras and Roy D. Yates, "Infostation Overlays in Cellular Systems," in *Proc. IEEE WCNC'1999*, September 1999.
- [11] David J. Goodman, "The Wireless Internet: Promises and Challenges," *IEEE Computer*, July 2000.
- [12] Frenkiel, Richard, B.R. Badrinath, Joan Borras, and Roy D. Yates, "The Infostations Challenge: Balancing Cost and Ubiquity in Delivering Wireless Data," *IEEE Personal Communications* 2000, 66-71.
- [13] P. Bagal, S. Kalyanaraman and B. Packer, "Comparative study of RED, ECN and TCP Rate Control," *Technical Report*, March 1999.
- [14] H. Balakrishnan, V. Padmanabhan and R.H. Katz, "The Effects of Asymmetry on TCP Performance," in *Proc. ACM MOBICOM'1997*, September 1997.
- [15] R. Stevens, "TCP/IP Illustrated, Volume I," *Addison-wesley*, 304-306, 1994.
- [16] "UCB/LBNL/VINT Network Simulator," <http://www-mash.CS.Berkeley.EDU/ns>.
- [17] "NS-2 with Wireless and Mobility Extensions," <http://www.monarch.cs.cmu.edu>.
- [18] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, V.1 N.4, August 1993, p. 397-413.
- [19] G. Bolch, S. Greiner, K. S. Trivedi, H. Meer, "Queueing Networks and Markov Chains : Modeling and Performance Evaluation With Computer Science Applications," *Wiley-Interscience*, 1998.