PAPER Special Section on Internet Technology VI

TCP BaLDE for Improving TCP Performance over Heterogeneous Networks*

Tuan-Anh LE[†a)], *Nonmember and* **Choong Seon HONG**[†b)], *Member*

SUMMARY Network congestion and random errors of wireless link are two well-known noteworthy parameters which degrade the TCP performance over heterogeneous networks. We put forward a novel end-to-end TCP congestion control mechanism, namely TCP BaLDE (Bandwidth and Loss Differentiation Estimate), in which the TCP congestion control categorizes the reason of the packet loss by estimating loss differentiation in order to control the packet transmission rate appropriately. While controlling transmission rate depends on the available bandwidth estimation which is apprehended by the bandwidth estimation algorithm when the sender receives a new ACK with incipient congestion signal, duplicates ACKs or is triggered by retransmission timeout event. Especially, this helps the sender to avoid router queue overflow by opportunely entering the congestion avoidance phase. In simulation, we experimented under numerous different network conditions. The results show that TCP BaLDE can achieve robustness in aspect of stability, accuracy and rapidity of the estimate in comparison with TCP Westwood, and tolerate ACK compression. It can achieve better performance than TCP Reno and TCP Westwood. Moreover, it is fair on bottleneck sharing to multiple TCP flows of the same TCP version, and friendly to existing TCP version.

key words: *bandwidth estimate, loss differentiation, congestion control, transport control protocol, wireless network*

1. Introduction

Wireless and mixed wired-wireless environments are gaining more popularity in recent years. As Transmission Control Protocol (TCP) constitutes almost all of local and wide-area network traffics, it is necessary to optimize the original TCP in order to bridge Internet services to the wireless world. TCP was originally designed for congestion control over wired environments which is characterized by very low error rates. One of the dramatic differences between wireless and wired part of heterogeneous network is in wireless network packet can be lost by random errors, signal fading or mobile handoff on wireless links. While original TCP assumes that every packet loss is an indication of network congestion. Therefore, in mixed wired-wireless environments, poor performance of TCP is erroneous in behaviors of the congestion avoidance if the packet loss does not concern the network congestion.

At the TCP sender side, the congestion control probes the available bandwidth of the bottleneck link by continuously increasing the congestion window size (*cwnd*) until reaching the network capacity. When the network congestion is detected by indicating Duplicate ACKs arriving at the sender, the congestion control decreases abundantly to one half of the current *cwnd* and sets to the slow start threshold (*ssthresh*). *cwnd* reset to one when retransmission timer is expired. If packet losses occur by random errors of wireless links before *ssthresh* reaches the actual network capacity, *ssthresh* can be given a smaller value. Therefore the sending rate is reduced blindly. Which indicates TCP performance is degraded unreasonably.

In this paper, we are interested in the end-to-end mechanism, in which the TCP congestion control categorizes the reason of the packet loss by estimating loss differentiation in order to appropriately control the packet transmission rate according to the available bandwidth estimated by Stable Accurate Rapid Bandwidth Estimate algorithm (SARBE).

The rest of this paper is organized as follows: Sect. 2 summarizes the related works. Section 3 articulates details of TCP BaLDE mechanism including an available bandwidth estimate algorithm, and a novel congestion control mechanism. Simulation results are presented in Sect. 4. Finally we conclude in Sect. 5.

2. Related Works

To improve TCP performance over wireless networks, several approaches have been proposed and classified into three classes [2]: the link layer approach, split connection approach, and end-to-end approach.

In the link-layer approach, techniques involve Forward Error Correction (FEC) [15], retransmission of lost packets in response to Automatic Repeat reQuest (ARQ) messages, or a hybrid of the two in AIRMAIL [6], [14]. In Snoop [10] scheme, at base station, a Snoop agent caches the transmitted segments that have not been acknowledged by the mobile host. When link local timeout or Duplicate ACKs indicates the packet loss over the wireless link, the snoop agent re-sends the packet from its local cache without TCP support. However, they require support of network routers at wireless link hops.

In split-connection approach [5], [7], a base station separates the wireless connection from the wired connection, and is responsible for retransmission of lost packets on wireless link. The base station requires resources as large buffers and processing to maintain two TCP connections. Further-

Manuscript received July 1, 2005.
Manuscript revised October 3, 2005.

[†]The authors are with the Department of Computer Engineering, Kyung Hee University, 1 Seocheon, Giheung, Yongin, Gyeonggi 449-701, Korea.

*This work was supported by MIC and ITRC Project. Dr. CS Hong is corresponding author.

a) E-mail: letuanh@ptithcm.edu.vn
b) E-mail: cshong@khu.ac.kr

DOI: 10.1093/ietcom/e89–b.4.1127

more, this can not preserve the end-to-end semantic of TCP connection.

The end-to-end approach improves the TCP performance either at the sender or the receiver without violating the 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 [4], before occurring handoff, the receiver sends Zero Window Advertisement (ZWA) to force the sender into the Zero Window Probe (ZWP) mode and prevents it from dropping its congestion window. The sender then freezes all retransmission timeout timers, *cwnd* and *ssthresh*, and enter a persist mode. The TCP sender sends ZWPs until the receivers window opens up. Freeze-TCP can only improve the TCP performance in handoff cases.

TCP Westwood scheme [16] falls into the end-to-end approach. In this scheme, the sender estimates available bandwidth dynamically by monitoring the rate of ACKs receiving as

$$b_k = \frac{d_k}{t_k - t_{k-1}}$$

where d_k is the amount of data acknowledged by the k ACK, t_k is the time of the kth ACK and t_{k-1} is the arrival time of previous ACK. To smooth the bandwidth sample, a filter was designed on gradually decreasing the bandwidth estimate as time elapses with absence of ACKs. The sequence of bandwidth estimates are expressed as

$$\begin{aligned}
\hat{b}_k &= \frac{2m-1}{2m+1}\hat{b}_{k-1} + \frac{0+b_{k-1}}{2m+1}, \\
\hat{b}_{k+1} &= \frac{2m-1}{2m+1}\hat{b}_k, \\
&\vdots \\
\hat{b}_{k+h} &= \left(\frac{2m-1}{2m+1}\right)^h \hat{b}_k.
\end{aligned}$$

where $\hat{b}_k$ is the filtered bandwidth at time t_k, $1/\tau$ is cutoff frequency of the filter. If time τ/m ($m \geq 2$) of timer has gone without receiving any new ACKs then the filter assumes the reception of a virtual sample $b_k = 0$.

The sender then updates *cwnd* and *ssthresh* to the estimated bandwidth in terms of window size as the following equation when fast retransmit or retransmission timeout event occurs.

$$ssthresh = \frac{BW}{Seg\_size} RTT_{\min}$$

where BW is the estimated bandwidth, RTT_{min} is the minimum of round trip time, and Seg_size is the segment size of TCP.

When ACK packet in Fig. 1 encounters queuing with cross traffic along the backward path of TCP connection, its time spacing is no longer the transmission time upon leaving router queue. This time spacing may be shorter than original time spacing ($\triangle t' < \triangle t$, in Fig. 1), called ACK compression

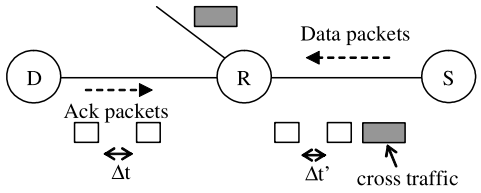

Fig. 1 ACK compression phenomenon.

[18]. In this case, TCP Westwood shows the estimated available bandwidth.

The end-to-end Loss Differentiation Estimate Algorithms (LDEA) categorize the packet losses explicitly through different estimation without any support from the intermediate routers, such as Flip Flop [13], Vegas [8], and Non Congestion Packet Loss Detection (NCPLD) [8]. They are based on the TCP state variables and information of ACKs to identify the reason of packet loss. NCPLD categorizes the nature of the error by detecting the knee point of the throughput-load bend.

The Vegas predictor measures the lowest Round Trip Time (RTT_{min}) during the TCP connection and computes the expected throughput ($cwnd/RTT_{min}$). When the sender receives a new ACK, it computes the actual throughput ($cwnd/RTT$). The literature [8] defined extra packets between two thresholds α and β in the network as following equation.

$$D_{Vegas} = RTT_{\min} \times \left(\frac{cwnd}{RTT_{\min}} - \frac{cwnd}{RTT}\right) \quad (1)$$

If $D_{Vegas} \geq \beta$, the Vegas predictor detects the network becoming incipient congestion. Otherwise, if $D_{Vegas} \leq \alpha$, there are more available bandwidth for connection. In the other hand, the network state is kept the same as in the last estimate when $\alpha < D_{Vegas} < \beta$.

The literature [19] showed that the predictor achieves the highest accuracy if $\alpha = 1$ and $\beta = 3$.

3. TCP BaLDE: Bandwidth and Loss Differentiation Estimate

3.1 SARBE Algorithm

In comparison with TCP Westwood, we take advantage of using ACK sending time interval to achieve more accurate available bandwidth estimate. SARBE [11], [20] is based on ACK inter-sending time interval to compute the available bandwidth of the forward path via the timestamp of ACK. In SARBE scheme, the estimate of the forward path is not affected by ACK compression that results in overestimate.

When the kth ACK in Fig. 2 arrives, the sender simply uses information of the kth ACK to compute an available bandwidth sample (Bw), which can be written as

$$Bw_k = \frac{L_k}{ts_k - ts_{k-1}} \quad (2)$$

where L_k is the amount of data acknowledged by the kth

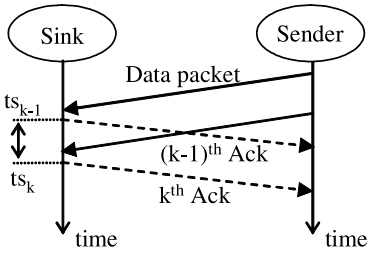

Fig. 2 Bandwidth estimate is based on ACK inter-sending time.

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. Consequently that the bandwidth estimator has to eliminate transient noise but responds rapidly to persistent changes.

We use the stability-based filter [17] similar to the Exponentially Weighted Moving Average (EWMA) filter, except using a measure function of the sampleslarge variance to dynamically change gain in the EWMA filter. After computing the bandwidth sample Bw_k from Eq. (2), the stability-based filter can be expressed in the recursive form as

$$U_k = \beta U_{k-1} + (1-\beta)|Bw_k - Bw_{k-1}| \quad (3)$$

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

$$\alpha = \frac{U_k}{U_{\max}} \quad (4)$$

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

where U_k is the network instability computed in Eq. (3) by EWMA filter with gain β, β was found to be 0.95 in our simulations; U_{max} 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 Eq. (4) when the bandwidth samples vary largely.

3.2 TCP BaLDE Mechanism

The congestion control maintains the packet transmission rate via variables *cwnd* and *ssthresh*. Therefore, the success of an algorithm mainly depends on updating these variables.

In our design, we propose a new scheme by incorporating SARBE and LDEA, namely, TCP BaLDE. In LDEA, we apply Eq. (1) to detect the network becoming incipient congestion and to distinguish the packet losses caused due to congestion from those caused due to random errors of wireless links. And then, relying on distinguishing the causes of losses, our scheme adjusts the packet transmission rate precisely according to the estimated bandwidth after receiving a new ACK, fast retransmit or retransmission timeout event occurs.

S sthresh 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 in terms of the window size for updating *ssthresh*. The literature [16] proposed the interrelation of estimated bandwidth with the optimal congestion window (*oCwnd*) size as

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

where RTT_{min} is the lowest round trip time, Seg_size is the length of the TCP segment.

The pseudo code of our algorithm is presented following.

3.2.1 Algorithm after Receiving a New ACK

```
if (a new ACK is received)
  // SARBE algorithm
  SARBE();
  // loss differentiation estimate algorithm
  isIncipientCongestion = LDEA();
  if( (cwnd < ssthresh) &&
      (isIncipientCongestion == True))
     ssthresh = oCwnd;
  endif
endif
```

Whenever the sender receives a new ACK, it calls the available bandwidth estimate algorithm and estimates the network state. If the bottleneck link of TCP connection is becoming congested, the congestion control updates *ssthresh* to *oCwnd* during the slow start phase. By precisely setting *ssthresh* to the available bandwidth of bottleneck link leads the sender to enter the congestion avoidance phase opportunely before router queue overflows.

3.2.2 Algorithm after Receiving Duplicate ACKs

```
if ( n DupACKs are received)
   ssthresh = oCwnd;
   // the packet loss is caused by congestion
   if (isIncipientCongestion == true)
     if (cwnd > ssthresh )
       cwnd = ssthresh;
     endif
   else // the packet loss is not caused
         // by congestion
        // keeping the current cwnd
   endelse
endif
```

When Duplicate ACKs are received, *ssthresh* is set to *oCwnd*. If the packet loss is caused by the network congestion, the congestion control should restart the CA phase during the CA phase. Otherwise, it keeps the current *cwnd*.

3.2.3 Algorithm after Timer Timeout Expiration

```
if ( retransmission timeout timer expires)
   ssthresh = oCwnd;
```

```
   cwnd = 1;
endif
```

If the sender is triggered by a retransmission timeout event due to the heavy network congestion or very high bit-error rate of wireless link, the congestion control sets *ssthresh* to *oCwnd* and then sets *cwnd* to one for restarting the SS phase.

4. Simulation Results

We investigate TCP BaLDE in terms of the performance, fairness and friendliness in the mixed wired-wireless networks, under the presence or the absence of congestion and random error of wireless link. TCP Reno and TCP Westwood were used for comparison in our experiments. Our simulations were run by the NS-2, network simulation tool [12]. We used the recent Westwood module of NS-2 [3] in all comparisons.

4.1 Accuracy and Stability of the Bandwidth Estimator

We first evaluated the stability, accurateness and rapidity of SARBE. The simulation network scenario is depicted in Fig. 3 where total link bandwidth is 1.5 Mbps and is shared by both TCP and UDP flows. We used a FTP over TCP and a UDP-based CBR background load with the same packet size of 1000 bytes. Available bandwidth for TCP flow with respect to time is given as the dotted line in Fig. 4.

The result is shown in Fig. 4; TCP Westwood is very slow to obtain the available bandwidth changes. By contrary to TCP Westwood, TCP BaLDE can reach 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.

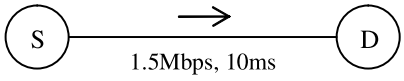

Fig. 3 Single bottleneck link.

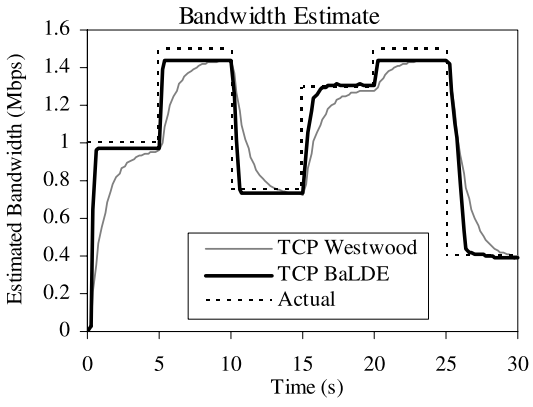

Fig. 4 Comparison of bandwidth estimators.

4.2 Impact of ACK Compression on the Bandwidth Estimator

To investigate the impact of ACK compression on estimate, we used the network scenario as Fig. 3 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, TCP Westwood estimates over 2 Mbps more than TCP BaLDE, which is close to the actual available bandwidth, as in Fig. 5.

4.3 Performance Evaluation

4.3.1 Performance over Wireless Networks

The simulation was run in a simple hybrid environment, shown in Fig. 6. The topology includes the bottleneck capacity of 5 Mbps, one-way propagation delay of 50 ms, the buffer capacity of routers equal to the pipe size, and a wireless link.

We evaluated TCP performance in the lossy link environment. 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 [16]. In Fig. 7, for any random error rate, the goodput of the proposed TCP is better than other versions. For example, at 1% wireless link packet loss rate, TCP BaDLE achieves better performance than TCP Reno and Westwood by 76.6% and 17.9%, respectively.

Outperforming of the proposal at any random error rate below 0.005% can be explained by the different behaviors of he three protocols shown in Fig. 8. At the beginning of the TCP connections, the initial *ssthresh* is set too high in the network scenario with small bandwidth-delay product. TCP

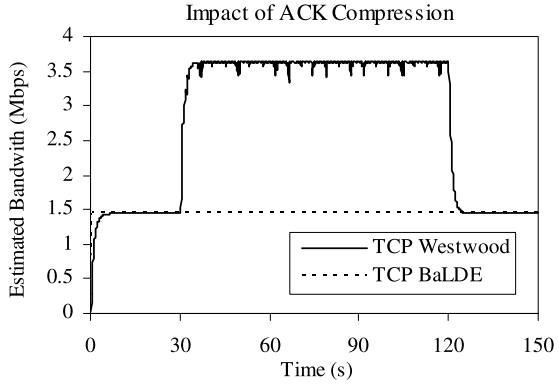

Fig. 5 Overestimated bandwidth of TCP Westwood in the presence of the ACK compression.

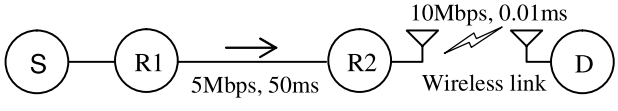

Fig. 6 The wired-wireless network scenario without cross-traffics.

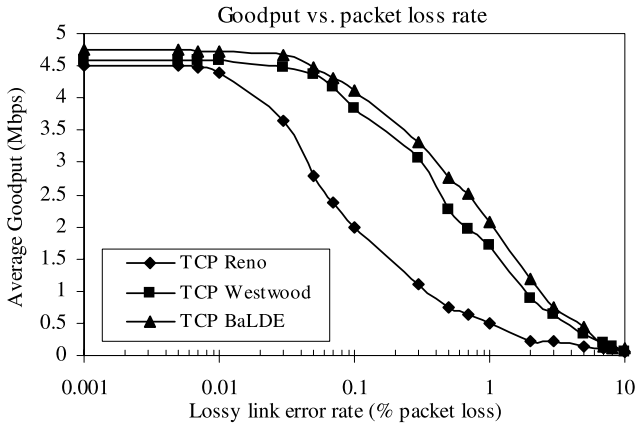

Fig. 7 TCP goodput vs. lossy link error rate without cross traffics.

Reno and TCP Westwood blindly probe the network capacity by exponentially increasing their *cwnd*. Upon their *cwnd* overshoots the bandwidth-delay product, this causes multiple packet losses and then a coarse timeout as in Figs. 8(a), and (b). The sender sets *ssthresh* to one-half of the current *cwnd* for TCP Reno, to the estimated bandwidth for TCP Westwood, and restarts the SS phase. In addition, TCP Westwood in Fig. 8(b) obtained a low available bandwidth. Thus, TCP Reno and TCP Westwood are poor in bandwidth utilization. By the contrary of TCP Reno and TCP Westwood, during the SS phase, TCP BaLDE detects the incipient congestion signal and sets *ssthresh* to the estimated bandwidth that close to the bandwidth-delay product. This leads the sender to opportunely enter into the CA phase to avoid packet loss caused by overflow at the router, shown in Fig. 8(c). Therefore, TCP BaLDE can utilize the available bandwidth of the bottleneck link more than TCP Reno and TCP Westwood.

4.3.2 Performance under Competing with Other Flows

To evaluate the TCP BaLDE performance in a more complex scenario, we simulated the heterogeneous network shown in Fig. 9 where one TCP connection is competing with 20 UDP connections. A traffic in the forward path of TCP connection is generated by 10 UDP connections on the bottleneck link of 5 Mbps. Other traffics of 10 UDP connections in the backward path of TCP connection share the bottleneck link between R1 and R2. Each UDP flow is modelled as an independent Pareto (shape parameter of 1.5) distributed on and off periods that set to 300 ms and 200 ms, respectively. During transmission, each flow carries packets with 1000-byte size at 500 kbps constant bit rate; the UDP flows do not transmit packet, when they are in off period.

Figure 10 shows the goodput of each TCP suffers from degradation when different TCP versions compete with the UDP traffics in shared bottleneck link. TCP BaLDE offers higher goodput than TCP Reno and TCP Westwood. This result is achieved due to the network state-aware congestion control mechanism and the adaptive bandwidth estimate of

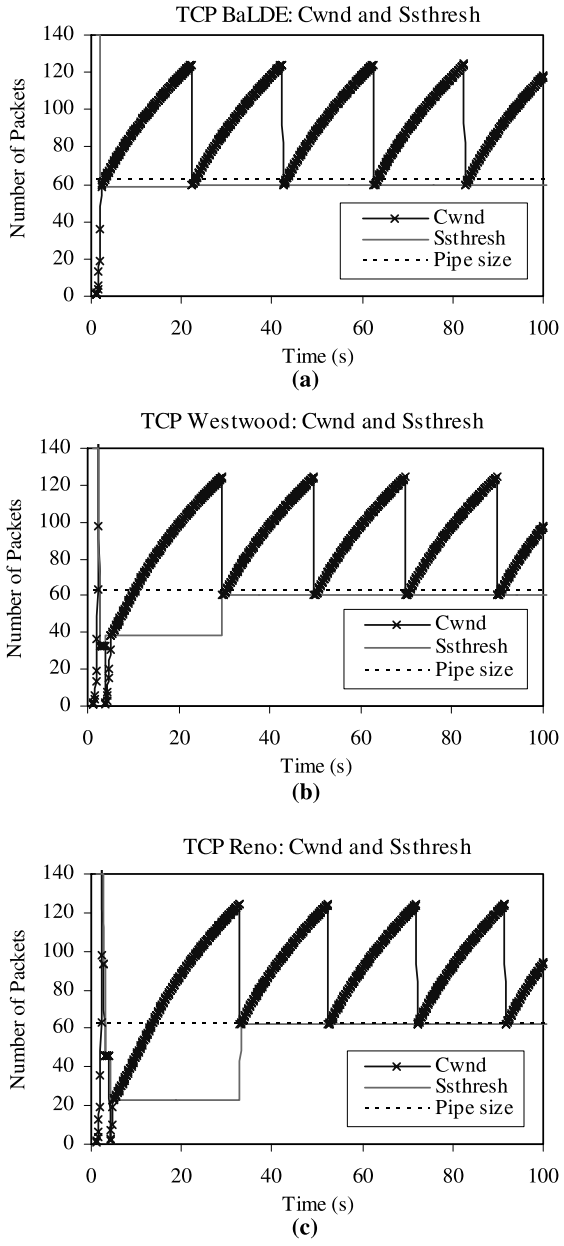

Fig. 8 *Cwnd* and *ssthresh* in the absence of random errors of (a) TCP Reno, (b) TCP Westwood and (c) TCP BaLDE.

TCP BaLDE. The UDP flows operate periodically by filling to the bottleneck link capacity in both directions and then silence. During no packet transmission of the UDP flows, if duplicate ACKs trigger the TCP sender to re-transmit a lost packet without incipient congestion signal given from the loss differentiation estimate algorithm, TCP BaLDE keeps the current *cwnd*, instead of reducing to one-half of the current *cwnd* and setting to *ssthresh* during SS phase as in TCP Reno and in TCP Westwood, respectively. Therefore, TCP BaLDE can utilize available bandwidth more than TCP Reno and TCP Westwood. In addition, because of the presence of the UDP traffics in the backward path during UDP flows transmit, TCP Westwood updates *ssthresh*

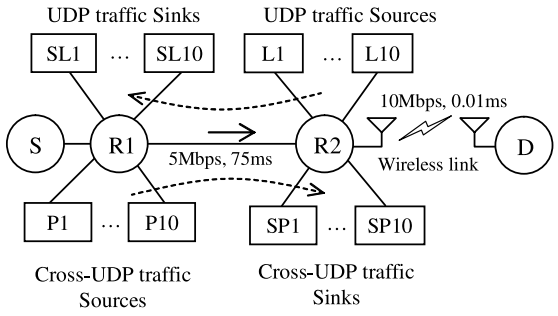

Fig. 9 The wired-wireless network scenario under competing with UDP traffics in the forward and backward path.

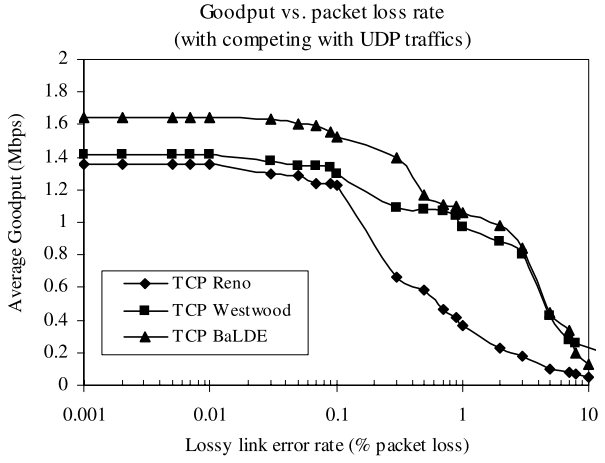

Fig. 10 TCP goodput vs. lossy link error rate under competing with UDP traffics in the forward and backward path.

to an estimated value greater than that value corresponds to the bandwidth-delay product. Thus, TCP Westwood is prone to loss packet due to over-buffer at the router. For instance at 0.001% loss rate [16], TCP Westwood goodput is worse 13.62% than TCP BaLDE, while worse 3.33% than that TCP BaLDE in the absence of traffic competing as in Fig. 7.

4.3.3 Performance under Bottleneck Link Capacity Changes

Simulation results in Fig. 11 show that TCP BaLDE and TCP Westwood performances are improved significantly as the bottleneck link capacity increases under condition of 0.1% packet loss rate and one way propagation delay of 50 ms. As the window size is determined by bandwidth-delay product, TCP BaLDE and TCP Westwood utilize the large bandwidth more effectively than TCP Reno. When the bottleneck link capacity is low, the protocols are not much different. Because the window size is small and optimization of the window size is not more effective.

4.3.4 Performance under Different Delay Times

As shown in Fig. 12, we investigated the performance of various TCP protocols when applied in some network environments under different RTTs. One way propagation delay varies from 20 ms to 180 ms, and the capacity of the bottleneck link is assumed to be 5 Mbps. The TCP protocols at 0.1% packet loss rate suffer the performance degradation when RTT increases. Because the sender increases *cwnd* in every RTT, when RTT increases, the sender grows *cwnd* slower (i.e. the bandwidth utilization is ineffective). When the packet loss does not concern the network congestion, TCP Westwood adjusts its sending rate to hit the estimated

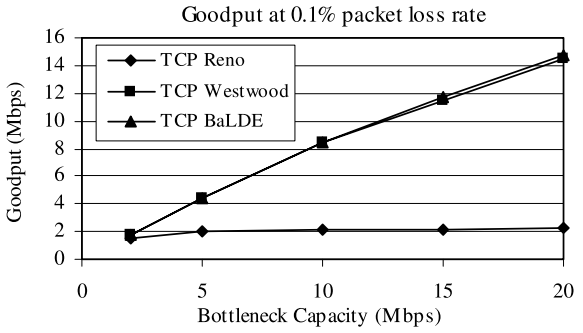

Fig. 11 TCP goodput under varying bottleneck link capacity.

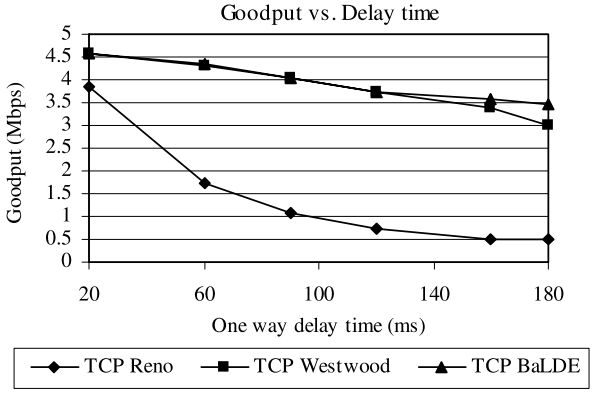

Fig. 12 TCP goodput with different delay times.

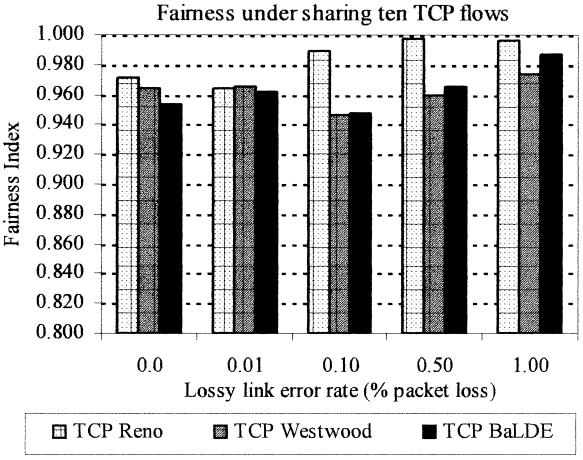

Fig. 13 Fairness of TCP under sharing 10 TCP flows.

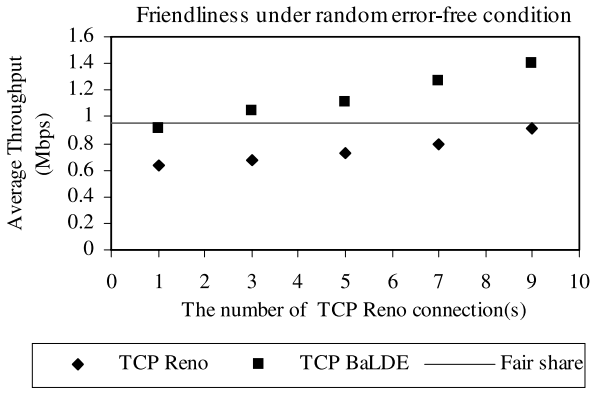

Fig. 14 Friendliness of TCP BaLDE and TCP Reno under random error-free condition.

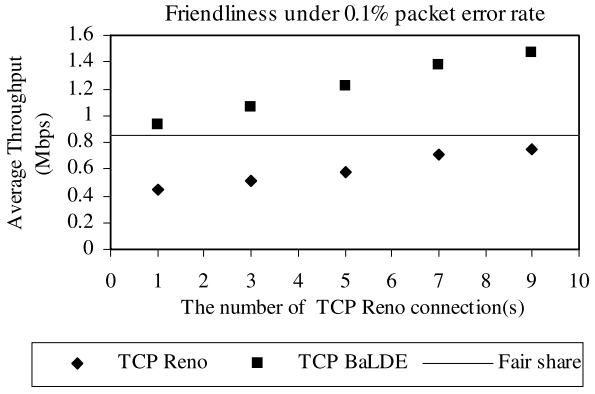

Fig. 15 Friendliness of TCP BaLDE and TCP Reno under 0.1% packet error rate.

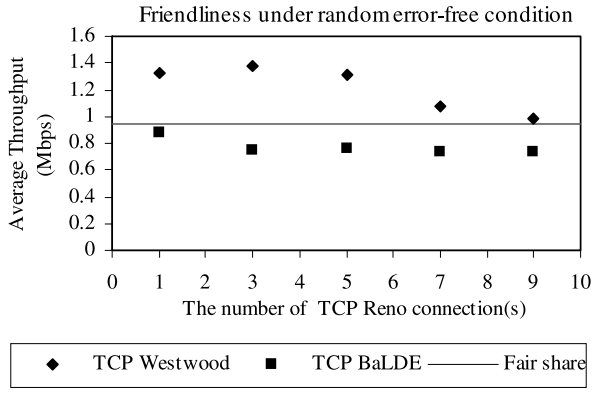

Fig. 16 Friendliness of TCP BaLDE and TCP Westwood under random error-free condition.

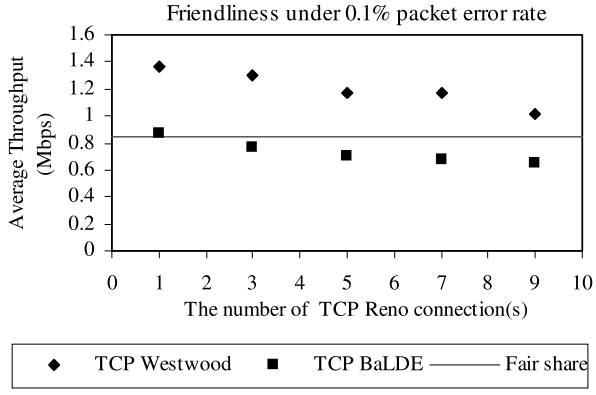

Fig. 17 Friendliness of TCP BaLDE and TCP Westwood under 0.1% packet error rate.

bandwidth whenever receiving duplicate ACKs. While TCP BaLDE classifies the packet loss caused by the wireless link, it does not reduce its sending rate unnecessarily. Therefore, TCP BaLDE performs better than TCP Reno and TCP Westwood as the propagation delay increases.

4.4 Fairness

Another evaluation for TCP is fairness that a set of connections of the same TCP version, which can share fairly with each other when they are sharing the same bottleneck link. The index of fairness was defined in literature [1] as

$$fi = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\left(\sum_{i=1}^{n} x_i^2\right)}, \quad \frac{1}{n} \le fi \le 1.0 \tag{5}$$

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, the value of 1.0 indicates the perfectly fair allocation.

Using the same scenario as Fig. 6 with ten same TCP flows, we simulated the different TCP versions individually, as shown in Fig. 13. Because TCP Reno connections regulate the same behaviors, Additive Increase Multiplicative Decrease (AIMD), when the packet loss probability is increased, the fairness can be improved since the average window sizes of TCP Reno connections obtain smaller. In contrast with TCP Reno, since TCP BaLDE and TCP Westwood connections can be more effective in utilization of the bandwidth (their average window sizes are larger), TCP connections window size gaps may be larger. Therefore, TCP BaLDE and TCP Westwood are worse in fair sharing than TCP Reno, nevertheless they can achieve an acceptable fairness index.

4.5 Friendliness

The friendliness of a TCP protocol implies fair bandwidth sharing with competing with other TCP version. Our experiments were run on the scenario of Fig. 6 with bottleneck link capacity of 10 Mbps and one way propagation delay of 35 ms; and 100 Mbps wireless link capacity. We have considered a total of ten flows mixing TCP BaLDE with TCP Reno at 0% and 0.1% packet loss rate of the wireless link. The x-axis of Figs. 14 and 15 represents the number of TCP Reno flows; the remaining connections used in TCP BaLDE. The dotted line is the fair share. The results in Fig. 14 show that the throughput under random error-

free condition achieved by TCP BaLDE is not far to the fair share. As the results in Fig. 7 and Fig. 10, the TCP Reno shows that the bandwidth utilization is ineffective under 0.1% packet error rate. Accordingly, there is more bandwidth unused for fewer TCP BaLDE flows as the number of TCP Reno flows in Fig. 15 increase. Therefore, TCP BaLDE flows will obtain more average throughput than TCP Reno.

TCP Westwood in Fig. 16 and Fig. 17 occupies more bandwidth than TCP BaLDE at 0% and 0.1% packet loss rate. This is because TCP Westwood overestimates shared bandwidth when there is some traffic in its backward path.

5. Conclusion

In this paper, we propose an end-to-end TCP mechanism to enhance the TCP performance over heterogeneous networks. TCP BaLDE is incorporating the stable accurate rapid bandwidth estimator and the loss differentiation estimate algorithm. The proposal can interact appropriately to the packet loss in heterogeneous networks, where the packet loss is caused by either network congestion or random errors of wireless links. LDEA detects the network becoming incipient congestion to help the sender to enter the CA phase opportunely before router queue overflows. By relying on distinguishing ability the causes of the packet loss, TCP BaLDE adjusts the packet transmission rate precisely according to the estimated bandwidth after receiving a new ACK, fast retransmit or retransmission timeout event occurs.

The simulation results show the stability, accuracy and rapidity of TCP BaLDE, as well as tolerating impact of ACK compression on the backward path. TCP BaLDE can improve better performance than TCP Reno, TCP Westwood under several different network conditions. Furthermore, it is fair on bottleneck sharing to multiple TCP flows of the same TCP versions, and friendly to existing TCP version. However, TCP BaLDE requires change in both the TCP sender and receiver sides.

Accuracy of loss differentiation algorithm and transforming window size-equivalent is mainly based on estimating the minimum RTT. During a TCP connection, if current routing changes to a new path with a longer propagation delay, the connection is not able to update the minimum RTT correctly. Therefore, our future research would focus on improving accuracy of the minimum RTT.

References

[1] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairnessand discrimination for resource allocation in shared computer systems," DEC, Res. 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. Netw., vol.5, no.6, pp.756–769, 1997. TCP Westwood - Modules for NS-2 [Online].

[3] TCP Westwood—Modules for NS-2 [Online]. Available: http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.html, 2004.

[4] T. Goff, J. Moronski, D.S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," Proc. IEEE INFOCOM 2000, pp.1537–1545, Tel-Aviv, Israel, March 2000.

[5] R. Yavatkar and N. Bhagawat, "Improving end-to-end performance of TCP over mobile internetworks," Proc. IEEE WMCSA 94, Santa Cruz, California, U.S., 1994.

[6] E. Ayanoglu, S. Paul, T.F. LaPorta, K.K. Sabnani, and R.D. Gitlin, "AIRMAIL: A link-layer protocol for wireless networks," ACM Wireless Networks, vol.1, no.1, pp.47–69, 1995.

[7] A. Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," ICDCS'95, pp.136–143, Oct. 1995.

[8] S. Biaz and N.H. Vaidya, "Distinguishing congestion losses from wireless transmission losses: A negative result," Seventh International Conference on Computer Communications and Networks (ICCCN), pp.722–731, New Orleans, Oct. 1998.

[9] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," RFC 2018, Oct. 1996.

[10] H. Balakrishnan, S. Seshan, and R.H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," Wirel. Netw., vol.1, no.4, pp.469–481, 1995.

[11] L.T. Anh and C.S. Hong, "Enhanced TCP with end-to-end bandwidth and loss differentiation estimate over heterogeneous networks," Lecture Notes in Computer Science, vol.3515, pp.436–443, April 2005.

[12] NS-2 network simulator [Online]. Available: http://www.isi.edu/nsnam/, 2004.

[13] D. Barman and I. Matta, "Effectiveness of loss labeling in improving TCP performance in wired/wireless networks," Boston University Technical Report, BUCS-TR-2002-016, 2002.

[14] A. Chockalingam and M. Zorzi, "Wireless TCP performance with link layer FEC/ARQ," Proc. IEEE Intl Conf. Commun., vol.2, pp.1212–1216, June 1999.

[15] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "Forward error correction (FEC) building block," RFC 3452, Dec. 2002.

[16] S. Mascolo, M.Y. Sanadidi, C. Casetti, M. Gerla, and R. Wang, "TCP westwood: End-to-end congestion control for wired/wireless networks," Wirel. Netw. J., vol.8, pp.467–479, 2002.

[17] 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, Aug. 2000.

[18] 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.

[19] S. Bregni, D. Caratti, and F. Martigon, "Enhanced loss differentiation algorithms for use in TCP sources over heterogeneous wireless networks," IEEE Global Communications Conference, Globecom 2003, pp.666–670, Dec. 2003.

[20] L.T. Anh and C.S. Hong, "Stable accurate rapid bandwidth estimate for improving TCP over wireless networks," Lecture Notes Comput. Sci., vol.3421, pp.141–148, March 2005.

Tuan Anh Le received the College Diploma in Telecommunications from Posts and Telecoms Training Center II, Ho Chi Minh city, Vietnam in 1996, and his B.S. degree in Information Technology from Natural Sciences University, Ho Chi Minh city in 2002. During 1997–2003, he worked as research assistance in R&D section of Faculty of Information Technology, Posts and Telecommunications Institute of Technology (PTIT), Ho Chi Minh city Campus, Vietnam. He received MS in Computer Engineering, Kyung Hee University, Korea. He is now a research associate in Faculty of Information Technology at PTIT, Ho Chi Minh city.

Choong Seon Hong received his B.S. and M.S. degrees in electronics engineering from Kyung Hee University, Seoul, Korea, in 1983, 1985, respectively. In 1988 he joined KT, where he worked on N-ISDN and Broadband Networks as a member of the technical staff. From Sept. 1993, he joined Keio University, Japan. He received the Ph.D. degree at Keio University in March 1997. He had worked for the Telecommunications Network Lab, KT as a senior member of technical staff and as a director of the networking research team until August 1999. Since September 1999, he has worked as a professor of the School of Electronics and Information, Kyung Hee University. His research interests include Ad hoc Networks, Network Security and Network Management. He is a member of IEEE, IPSJ, and Korean Institute of Communication Sciences (KICS).