Snug-Vegas and Snug-Reno: efficient mechanisms for performance improvement of TCP over heterogeneous networks

C.-Y. Ho and Y.-C. Chen

Abstract: Improving the performance of the traditional TCP in wireless IP communications has been an active research area. The significant cause of packet losses in such heterogenous networks is no longer limited to network congestion. The performance degradation of TCP in wireless and wired-wireless hybrid networks is mainly due to its lack of ability to differentiate the packet losses caused by network congestions from the losses caused by wireless link errors. New variants of TCP Vegas and TCP Reno named Snug-Vegas and Snug-Reno, respectively, are proposed. By using random-loss indications marked by base stations, Snug-Vegas and Snug-Reno may detect random packet losses precisely. Through the packet loss differentiation, Snug-Vegas and Snug-Reno react appropriately to the losses, and based on the simulation results, it is seen that the throughput of connection over heterogeneous networks can be significantly improved.

1 Introduction

It is increasingly important to provide ubiquitous mobile Internet access because of the great advancement in wireless networking technology and new emerging applications. The well-known problem in providing transmission control protocol/Internet protocol (TCP/IP) [1, 2] implementations over heterogeneous networks (wired/wireless environment) is how to depart from its original wired network oriented design and evolve to meet the challenges introduced by the wireless portion of the network. In the wired networks, a congestion is indeed a likely reason of packet loss. Alternatively, a noisy, mobile, and fading radio channel is the most likely cause of loss in the wireless networks. The effective bit error rates in wireless networks are significantly higher than that in wired networks. Since TCP does not have any mechanism to differentiate between congestion losses and wireless random losses, the latter may cause a severe throughput degradation.

The purpose of congestion control is to dynamically adapt the end-to-end transmission rate of a connection to the currently available capacity. TCP performs at an acceptable efficiency over the traditional wired networks where packet losses are caused by network congestion. However, when TCP observes random losses, it misinterprets such losses and reduces its window size, this causes the reduction of throughput unnecessarily. Therefore, TCP's performance drops rapidly in the presence of frequent random losses [3, 4].

TCP has several implementation versions (i.e., Tahoe, Reno, Vegas) that aim to improve network utilisation. Among these TCP variants, there are two notable approaches. One is Reno [5], which has been widely deployed on the Internet; the other is Vegas [6] with a claim of 37 to 71 percent throughput improvement over Reno being achieved. The throughput deterioration problem of TCP over wireless networks has been addressed in [7–18]. However, parts of the solutions are designed especially for either Reno [7, 8] or Vegas [18], no TCP-based mechanism is designed for both TCP versions.

In this work, we propose a base station-based random loss detection mechanism for both TCP Vegas and TCP Reno (abbreviated as Snug-Vegas and Snug-Reno hereafter, respectively). By using the random loss indications marked by a base station, Snug-Vegas and Snug-Reno may detect random losses more precisely than TCP Vegas and TCP Reno do. Through the packet loss differentiation, Snug-Vegas and Snug-Reno react appropriately to the losses, and based on the simulation results, we show that the throughput of connection over heterogeneous networks can be significantly improved.

2 Related work

In the all-IP heterogeneous networks, congestion is no longer the only cause of packet loss. Therefore, conventional TCP schemes may suffer from a severe degradation in performance in mixed wired and wireless environment [3, 19]. For this reason, several approaches have been proposed to optimise TCP for wireless networks. The solutions can be categorised into the link layer mechanisms, end-to-end TCP modifications, and base station schemes.

Link layer mechanisms: Link layer mechanisms try to improve the quality of the lossy wireless link. They hide the characteristics of the wireless link from the transport layer and try to rectify wireless link errors at the second layer. The intuition behind link layer mechanisms is to treat the problem as local, and to solve it locally. Techniques such as forward error correction (FEC), automatic repeat request

IEE Proceedings online no. 20050184

doi:10.1049/ip-com:20050184

Paper first received 27th April and in final revised form 11th October 2005

The authors are with Department of Computer Science and Information Engineering, National Chiao Tung University, No. 1001, Ta Hsueh Road, Hsinchu City, 30050, Taiwan

E-mail: cyho@csie.nctu.edu.tw

(ARQ), and explicit loss notification (ELN) have been proposed for this class of solutions [9, 10, 16]. However, such an approach requires protocols at different layers to interact and coordinate closely, which increases the complexity of protocol implementation.

End-to-end modifications: In the end-to-end modifications [7, 8, 11], TCP senders and receivers are responsible for the flow control; hence, the end-to-end semantics of TCP are preserved. Moreover, the TCP source attempts to handle the losses in a way that improves the performance of a connection that runs on wireless networks. In a wireless environment, the major cause of packet losses is not limited to network congestion, thus some rules are used to infer the cause of packet losses. Based on the inferences, a source may react appropriately to the losses. These approaches do not need any extra support from the intermediate hops along the path.

RedVegas [18] uses the innate nature of TCP Vegas and congestion indications marked by routers to distinguish between congestion loss and random loss. If the router detects congestion and marks a congestion indication bit on all packets in the current buffer, the consecutive packets of the same connection prior to the dropped packet will very likely be marked. Based on the acknowledgments (ACKs) of the marked packets, a RedVegas source may infer the cause of packet loss accordingly.

Base station schemes: If only the last hop connecting to a mobile host is a wireless link, a TCP-aware agent, which tries to shield the wireless network from the wired network, can be run on the base station to improve the performance of connections.

Split connection approaches [12–14, 17] isolate mobility and wireless related problems from the existing network protocols. This is done by splitting the TCP connection between the mobile host and the fixed host into two separate connections: a wired connection between the fixed host and the base station, and a wireless connection between the base station and the mobile host. In this way the wired connection does not need any change in existing software on the fixed hosts, and the wireless connection can use a specialised mobile protocol to provide better performance.

The snoop protocol [15] introduces a snoop module at a base station to monitor every packet transmitted through the connection in either direction. By caching recently transmitted TCP packets sent to a mobile host and monitoring the associated acknowledgment packets returning to the source, the snoop module can quickly resend a cached copy of the lost packet to the mobile host. The snoop protocol hides the packet loss from the fixed host and hence avoids the unnecessary invocation of congestion control mechanism.

In short, the intermediate node of base station schemes, usually the base station, sets up the connection with the fixed host and the mobile host, and is responsible for the recovery of the packet losses caused by wireless links. However, these approaches require large buffers at base stations and the end-to-end TCP semantics are sometimes not preserved.

3 TCP schemes and proposed mechanisms

With the fast growth of Internet traffic, how to efficiently utilise network resources is essential to a successful congestion control. In this Section, we first briefly review the design principles of TCP schemes, and then describe proposed mechanisms in detail.

3.1 TCP schemes

TCP Reno and TCP Vegas adopt an end-to-end closed-loop adaptive window congestion control. It is based on five fundamental mechanisms: slow start, congestion avoidance, retransmission timeout, fast retransmit, and fast recovery.

TCP Reno and TCP Vegas use slow start at the beginning of the connection and whenever a packet loss is detected via timeout. When the congestion window reaches a threshold value, called slow start threshold, TCP Reno and TCP Vegas leave slow start and enter congestion avoidance. Both protocols can detect packet losses by means of two mechanisms. If the coarse-grained timeout (set when the packet is sent) expires, TCP Reno reduces its congestion window to one packet size [Note 1] and TCP Vegas decreases its congestion window to two packet size, then they start again in slow start mode [Note 2]. Otherwise, if three duplicated acknowledgments arrive back to the sender before the timeout expiration, the protocols perform fast retransmit and fast recovery. During the connection, the TCP receiver can limit the sender congestion window by advertising the receiver window value. Because space is limited, please refer to [5, 6] for details.

TCP NewReno [20] is a modification to the fast retransmit and recovery. In TCP NewReno, a sender can recover from multiple packet losses without having to time out. TCP with selective acknowledgment (SACK) options [21] also has been proposed to efficiently recover from multiple packets loss. However, the additive increase and multiplicative decrease approach (AIMD) of Reno leads to periodic oscillations in the congestion window size, round-trip delay, and queue length of the bottleneck node. Recent studies have shown that the oscillation may induce chaotic behaviour into the network thus adversely affecting overall network performance [22, 23].

3.2 Proposed mechanisms

The issue of packet losses differentiation can be divided into two parts: (1) how to distinguish between congestion loss and random loss, and (2) how to make use of the information to refine the congestion window adjustment process. The success of our proposed mechanisms relies on the cooperation of the end-hosts and base station. It assumes that the base station is capable of marking packets when random packet loss occurs.

The key idea of our proposed mechanisms is described as follows. In the base station part, the base station watches every packet that passes through the connection in either direction. In the direction of data packet (from the base station to the destination), the base station gives a sequence number [Note 3], which is added in an IP option field (named SNIP), to every data packet of the same flow. For example, if source A sends data packets to destination B through a base station C, C will mark number 1 to SNIP field of the first data packet, number 2 to the second data

Note 1: From now on, we measure the window size in number of packets.

Note 2: Actually, the initial slow start congestion window value depends on the specific implementation for both TCP Reno and TCP Vegas.

Note 3: This way is like the sequence number of TCP. Further, for a practical TCP implementation, a sequence number identifies the byte in the stream of data. In this paper, to ease the description of the proposed mechanism, we use a sequence number to represent the packet.

packet, and so on. In the direction of ACK (backward path), when the base station receives an ACK, it will do two steps: (1) checks the sequence number, and (2) removes the SNIP field of every packet. If the sequence number of an ACK is not continuing with that of last ACK of the same flow, it means that there is at least one random packet loss. Then, the base station will mark a reserved bit, called random-loss indication flag (RI), of 'Flags field' in IP header of this ACK in order to notify the source that random loss occurred. In the destination part, destination just copies the sequence number of the SNIP field from the data packet to the SNIP field of the ACK packet when acknowledging a received packet. In the source part, the source checks the RI flag as it receives a duplicate ACK. If the flag is set, the source will record this message by setting a boolean number, named random loss value (RLV), to true, not extend the retransmission timeout (backoff time) and not reduce the sending rate.

We take Snug-Vegas for example; Snug-Vegas has three ways to detect packet loss: a triple-duplicate ACK, a fine-grained timeout, and a coarse-grained timeout, as in Vegas. Whenever a packet loss is identified by a triple-duplicate ACK or a fine-grained timeout, Snug-Vegas will try to infer the cause of loss. When a packet loss is detected and no duplicate ACK's RI flag is marked, Snug-Vegas assumes that the loss is a congestion loss. Otherwise, random loss is inferred. If the losses are identified by a coarse-grained timeout, Snug-Vegas does not intend to infer the cause of losses. Since the losses are severe and the information carried by the received ACKs may be passed. Snug-Vegas leaves this part of algorithm to be intact. After the lost packet is recovered, Snug-Vegas will adjust the congestion window size according to the loss differentiation. A connection needs not to reduce the sending rate if the loss was not caused by congestion. Thus, if the lost packet is detected as a random loss, the current congestion window size (CWND) will not be changed, that is, the CWND will be equal to the congestion window size when the loss is detected. However, if a congestion loss is perceived, the same window-adjustment mechanism as that in Vegas will be adopted.

TCP congestion control is mainly based on the feedback of ACKs. The control procedure will be triggered whenever an ACK is received by the connection source. Figure 1 illustrates the detailed procedure of Vegas/Snug-Vegas as it receives an ACK. Shaded blocks are for Snug-Vegas especially. The description of variables used in Fig. 1 is shown in Table 1.

4 Performance evaluation

In this Section, we simulate our proposed mechanism in order to show the goodput performance, fairness, and friendliness in mixed wired and wireless networks, with or without the presence of link loss and congestion. In addition, we compare the performance among some TCP variants (Vegas, RedVegas, Reno, Snug-Reno, and Sung-Vegas) by using the network simulator *ns2*, version 2.26 [24].

4.1 Goodput performance

The first-in first-out (FIFO) service discipline is assumed. All parameter settings of Vegas, RedVegas, and Sung-Vegas, or of Reno and Snug-Reno are the same. Especially, $\alpha = 1$ and $\beta = 3$. The size of each FIFO queue used in routers is 16 packets, the size of data packets is 1 kbyte. To ease the comparison, we assume that the sources always have data to send.

The network configuration for the goodput simulations is shown in Fig. 2, in which the bandwidth and delay of each full duplex link are depicted. Sources, destinations, and routers are expressed as S_i, D_i, and R_i, respectively. The link between R_2 and D_1 is a wireless link on which we assume all random losses occur. Typically, wireless links are subject to fading phenomena, which results in random loss in bursty manner. However, if random losses appear in bursty manner, it is easy for Snug-Vegas and Snug-Reno to recognise them. It is because Snug-Vegas and Snug-reno take a packet loss to be random loss when there is a packet to be marked. Therefore, we use a more complicated case, uniformly distributed loss model, to evaluate our proposed mechanisms. In wireless environments, if the bit error is uniformly distributed, the larger a packet is, the more likely the packet will be corrupted. Keeping this in mind, when we apply a random loss rate to data packets, we always set the proportional random loss rate to ACKs.

A TCP connection is established from S_1 to D_1, and a variable bit rate (VBR) source is used to generate cross traffic from S_2 to D_2. Complying with the study of Internet simulating [25], the VBR source is a Pareto distribution ON-OFF source with shape parameter 1.5. During ON periods, the VBR source sends data at 1.6 Mbit/s. In the following simulations, unless stated otherwise, the execution time of each sample point is 12 hours.

4.1.1 Basic behaviour: The design goal of Snug-Vegas and Snug-Reno is to improve performance for TCP Vegas and TCP Reno over heterogeneous networks. It is obvious that if the packet losses are the result of congestion, the behaviour of Vegas and Snug-Vegas, or Reno and Snug-Reno should be the same. In this Section, we examine the average goodputs among the five TCP variants with different cross traffic loads. The difference between the throughput and goodput is that the latter only counts those packets effectively received once. Each TCP variant is examined separately and the results can be found in Fig. 3.

With the increasing cross traffic load, the average goodputs of all TCP variants degrade accordingly. Note that the goodputs of Vegas and Snug-Vegas, or Reno and Snug-Reno are always identical. The results demonstrate that the behaviour of Vegas and Snug-Vegas, or Reno and Snug-Reno is the same when there is no random loss. It implies that Sung-Vegas and Sung-Reno do not misinterpret congestion loss as random loss in such simulation scenarios. From the simulation results, Vegas always surpasses Reno in goodput with different cross traffic loads, this conforms to the previous studies [6, 18, 26–28].

4.1.2 Impact of random loss: In this Section, we compare the average goodputs among the five TCP variants with different random loss rates. No cross traffic is introduced in the simulations. By observing the results shown in Fig. 4, Vegas, RedVegas, and Snug-Vegas can fully utilise the bottleneck link when the random loss rate is zero. However, Reno cannot maintain such high goodput with the same condition. This is because Reno needs to create packet losses by itself to probe the available bandwidth along the path. Therefore, a certain amount of goodput is lost.

When the random loss rate is increased, the goodput improvements of Sung-Vegas and Snug-Reno become obvious. When the random loss rate is 5%, the goodput of Snug-Vegas is about 1.51 times higher than that of Vegas, 1.17 times higher than that of RedVegas, and 3.31 times higher than that of Reno. Similarly, the goodput of Snug-Reno is about 2.07 times higher than that of Reno.

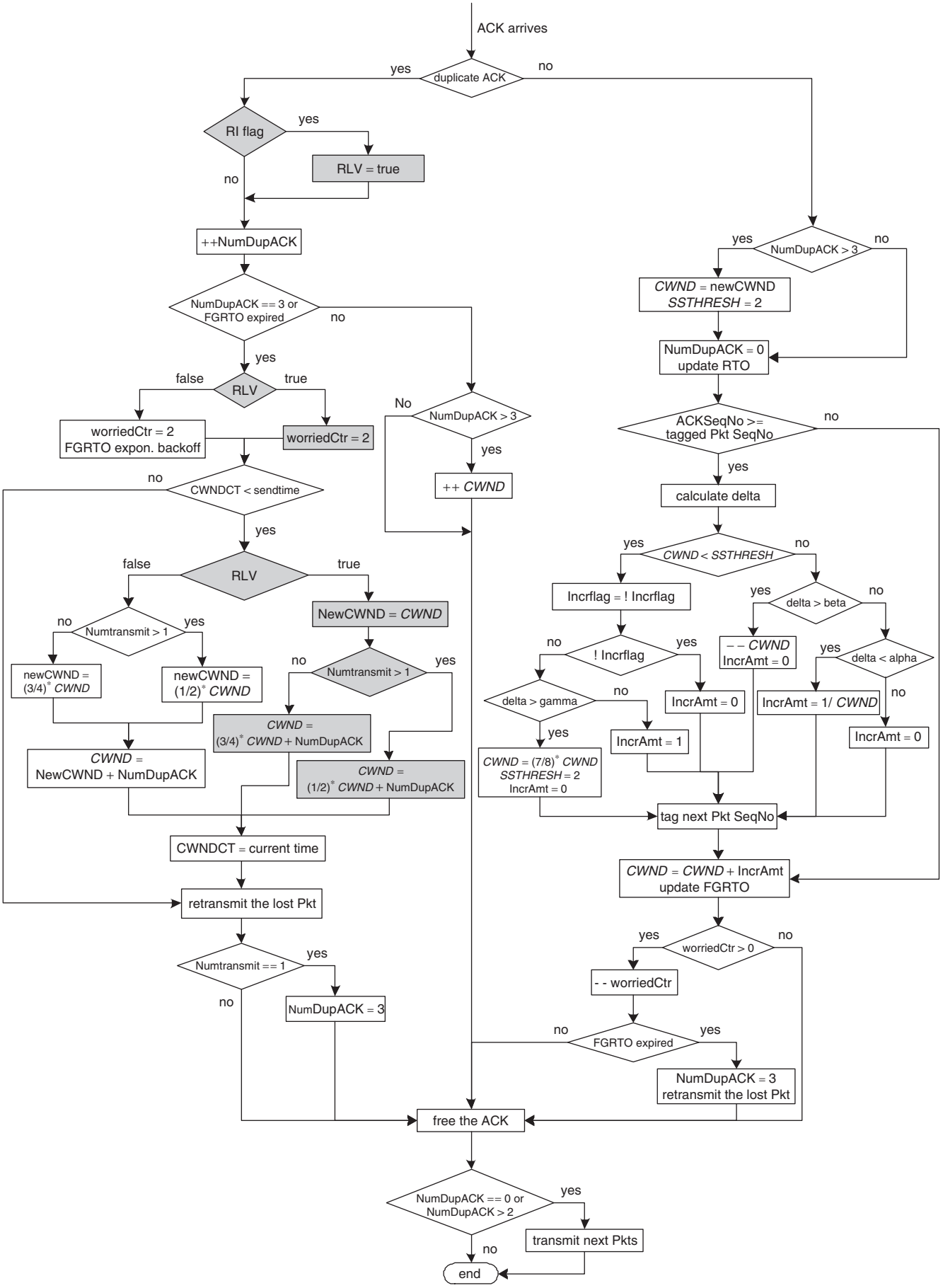

Fig. 1 *Flowchart to illustrate the procedure of Vegas/Snug-Vegas as it receives an ACK*

When the random loss rate is between 3 and 15%, Snug-Vegas always keeps a goodput improvement of larger than 25.6% in comparison with Vegas. Notably, with 12% random loss rate, the goodput improvement is up to 105.77%. With the same condition, Snug-Reno always keeps a goodput improvement of larger than 40% in

Table 1: Variables description

Variable	Description
NumDupACK	number of duplicate ACK
RTO	duration of the coarse-grained retransmission timer
FGRTO	duration of the fine-grained retransmission timer
CWNDCT	last congestion window adjustment time owing to a packet loss detection
SendTime	sending time of the lost packet
Delta	amount of extra data
NumTransmit	number of transmission times of the lost packet
NewCWND	congestion window size that will be used as a lost packet is recovered
IncrFlag	a flag used to adjust congestion window every other round-trip time
IncrAmt	increment amount of congestion window size for each new ACK is received
WorriedCtr	a counter used to check FGRTO after a lost packet is recovered

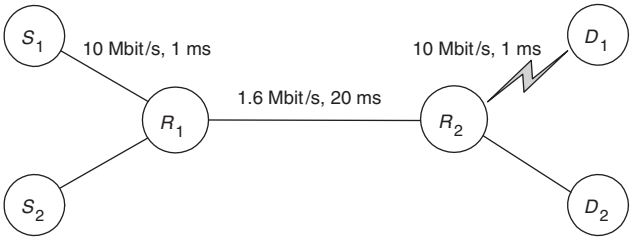

Fig. 2 *Network configuration for the goodput simulations*

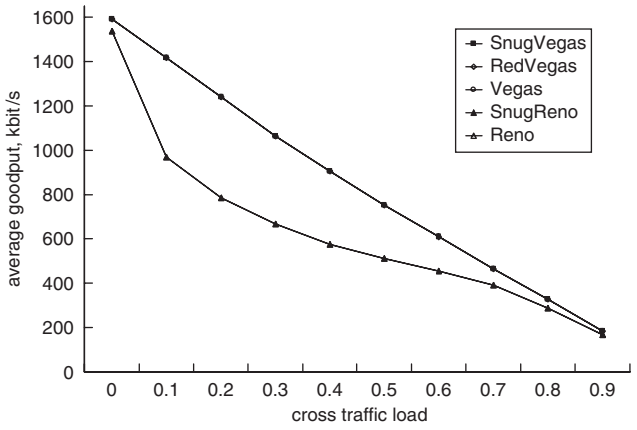

Fig. 3 *Average goodput against cross traffic load for TCP variants*

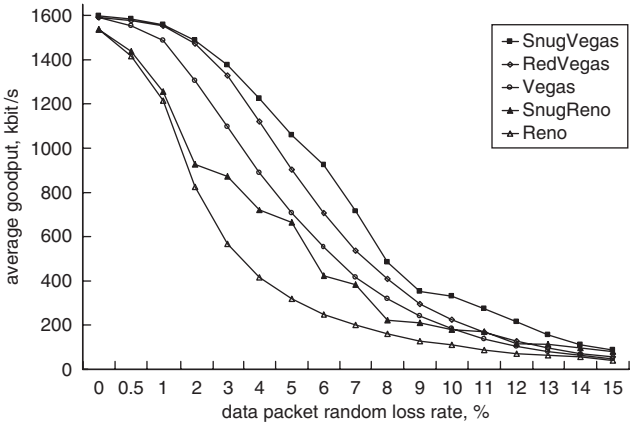

Fig. 4 *Average goodput against random loss rate for TCP variants*

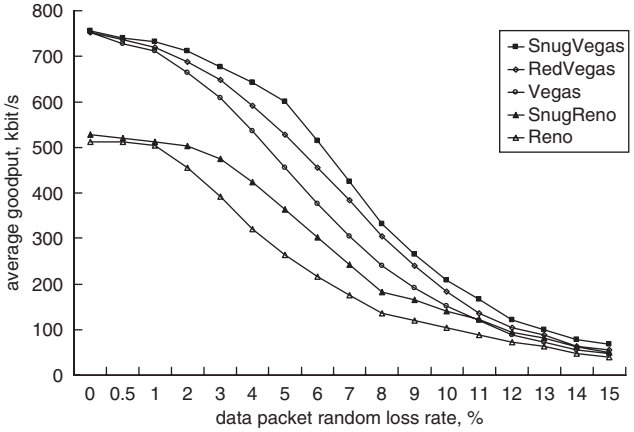

Fig. 5 *Average goodput against random loss rate for TCP variants with cross traffic load of 0.5*

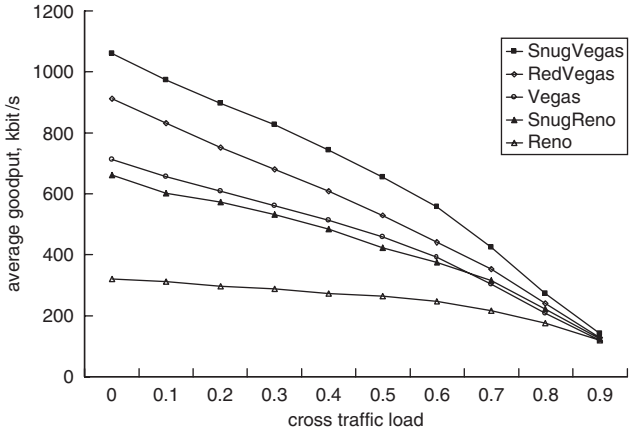

Fig. 6 *Average goodput against cross traffic load for TCP variants with random loss rate of 0.05*

comparison with Reno. The goodput improvement is up to 102.5%, especially, when there is 15% random loss rate in the wireless link. Additionally, in a severe random loss rate, the goodputs of Snug-Vegas and Snug-Reno are still not bad. For example, the goodput improvement of Snug-Vegas is 83.33% as compared with Vegas, and the goodput of Snug-Reno is about 2.03 times higher than that of Reno.

4.1.3 Impact of random loss and cross traffic: TCP connections over heterogeneous networks may experience both random losses and congestion losses. In this Section we introduce random losses and cross traffic into the simulation to examine the goodputs of five TCP variants. The results are shown in Figs. 5 and 6.

Figure 5 depicts the goodputs of the five TCP variants with the random loss rate varying from 0 to 15% and the VBR source with 800 kbit/s averaged sending rate to generate cross traffic. The simulation results demonstrate that both Snug-Vegas and Snug-Reno can always maintain higher goodputs than their original version (i.e., Vegas and Reno), respectively. As compared with Vegas, when the random loss rate is greater than 2%, Snug-Vegas always

achieves more than 11% goodput improvement. In particular, when the random loss rate is 7%, the goodput improvement of Snug-Vegas reaches 39.8%. Similarly, Snug-Reno always keeps a goodput improvement of larger than 21.17% in comparison with Reno when the random loss rate is between 3 and 15%. Notably, with 6% random loss rate, the goodput improvement is up to 39.35%.

As the random loss rate is fixed at 5% and the cross traffic load varies from 0 to 0.9, the simulation results also illustrate that the goodputs of Snug-Vegas and Snug-Reno are respectively higher than Vegas and Reno as shown in Fig. 6. Compared with Vegas, the goodput improvement of Snug-Vegas is kept between 17.5 and 48.9%; while it is kept between 17.5 and 231.3% compared with Reno. Moreover, the goodputs of Snug-Reno are from 1.04 to 2.07 times higher than those of Reno when the cross traffic load is between 0.9 and 0.

4.2 Fairness

Another important issue of TCP is the fairness. Multiple connections of the same TCP scheme must interoperate nicely and converge to their fair shares. We use the fairness index fuction (1), proposed in [29], to justify the fairness of TCP schemes. The fairness index function is expressed as

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)} \quad (1)$$

where x_i is the throughput of the ith connection, and n is the number of connections, $F(x)$ ranges from $1/n$ to 1.0. A perfectly fair bandwidth allocation would result in a fairness index of 1.0. On the contrary, if all bandwidth are consumed by one connection, (1) would yield $1/n$.

We set up the simulation as shown in Fig. 7, where a total of $20(m+n)$ same TCP flows share a 20 Mb bottleneck link, and the R_2 connecting to each destination (from D_1 to D_{20}) is a wireless link. We run the simulation for different TCP schemes and compare their fairness index; the results are summarised in Table 2. All TCP variants including our proposed mechanisms achieve fairly satisfactory fairness index. The fairness index only demonstrates the whole situation, so we show the detailed average throughtput of all TCP variants in Table 3. From Table 3, we find that the average throughputs of Snug-Vegas (or Snug-Reno) are higher than those of Vegas (or Reno), especially in a severe random loss rate.

4.3 Friendliness

A friendly TCP scheme should be able to coexist with other TCP variants and not cause them starvation. To verify the friendliness of our proposed mechanisms, we construct a mixed wired and wireless network, where Snug-Vegas coexists with Vegas, or Snug-Reno coexists with Reno. The reason is that when a Vegas user competes with other Reno users, it does not receive a fair share of bandwidth

Table 2: Fairness comparison

Error Rate	Vegas	Snug-Vegas	Reno	Snug-Reno
0.0	0.98	0.98	0.99	0.99
0.1	0.99	0.99	0.99	0.99
0.5	0.99	1	0.99	1
1.0	1	1	0.99	0.99
5.0	0.99	0.99	0.99	0.99
10.0	0.97	0.98	0.97	0.98

The fairness index is calculated based on a total of 20 TCP connections running the FTP application. Error rate is in units of percentage of packet drops. Link error rate varies from 0 to 10%

Table 3: Average throughput (kbits/s) comparison

Error Rate	Vegas	Snug-Vegas	Reno	Snug-Reno
0.0	999.33	999.33	951.58	951.58
0.1	998.20	998.75	950.85	950.90
0.5	994.14	996.46	946.71	949.23
1.0	988.72	992.34	942.32	945.14
5.0	767.92	801.22	354.67	403.64
10.0	192.49	251.13	123.02	143.72

The average throughput is calculated based on a total of 20 TCP connections running the FTP application. Error rate is in units of percentage of packet drops. Link error rate varies from 0 to 10%

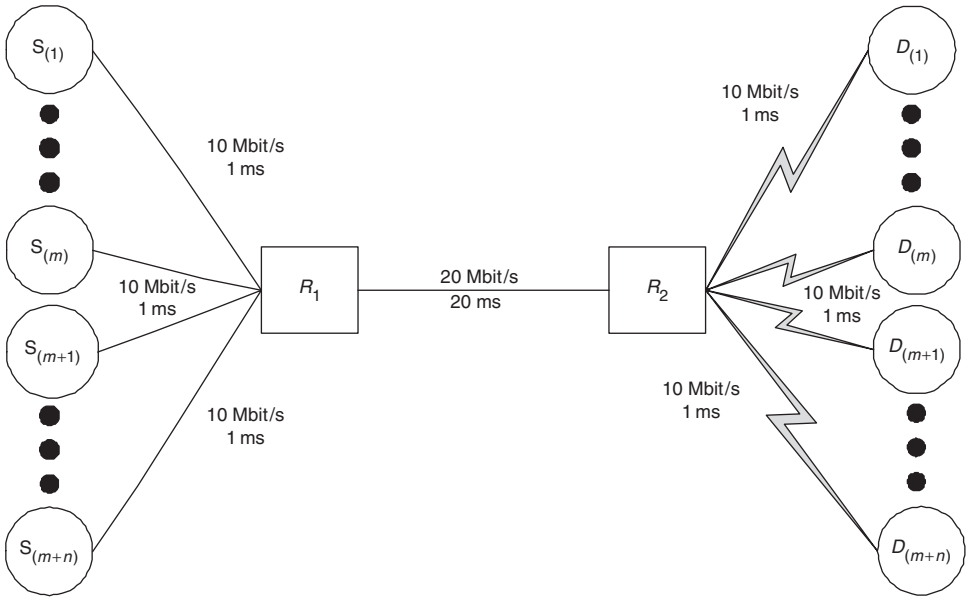

Fig. 7 *Simulation network for obtaining fairness index and verifying friendliness*

owing to the conservative congestion avoidance mechanism used by Vegas. It is because Reno continues to increase the window size until a packet is lost. This would occur mainly owing to buffer overflow (if the queue management algorithm is drop tail). This bandwidth estimation mechanism results in a periodic oscillation of window size and buffer-filling behaviour of Reno. Thus, while Vegas tries to maintain a smaller queue size, Reno keeps inserting many more packets into the buffer, and stealing more bandwidth [26–28].

The simulation network is shown in Fig. 7. The bandwidth of access links is 10 Mbit/s, and propagation delays are 1ms. In addition, the access link from R_2 to D_i, $i = 1, 2, \ldots, (m+n)$, is a wireless link. The bandwidth of connection link is 20 Mbit/s, and propagation delay is 20 ms. There are 20 paris of connections, of which m are Snug-Vegas (or Snug-Reno) connections and n are Vegas (or Reno) connections. We vary the proportion of these two TCP schemes in the network by adjusting the variables m and n. Without the presence of congestion and link loss, all 20 connections are expected to share the bottleneck bandwidth equally, i.e., roughly 1 Mbit/s per connection.

From Section 4.1.1, we know the behaviour of Vegas and Snug-Vegas, or Reno and Snug-Reno is the same when there is no random loss. Therefore, we omit the simulation with 0% link error rate at the wireless links. We only set the link error rate to 0.1% at the wireless links in our simulation environment. The throughput results are listed in Table 4 and Table 5. Our proposed mechanisms achieve a slightly higher throughput than original TCP versions (i.e., Vegas and Reno) when a lossy link exists, but within a tolerable range. The mean throughput of both TCP schemes is still close to the fair share.

Table 4: Throughput comparison between Vegas and Snug-Vegas over lossy links

Vegas source	Snug-Vegas source	Vegas mean throughput	Snug-Vegas mean throughput
3	17	996.60	999.23
5	15	996.99	998.81
10	10	997.27	999.42
15	5	997.93	998.91
17	3	996.58	999.54

The number of Vegas and Snug-Vegas sources varies. The total number of all sources is 20. Mean throughput is in units of kbit/s. Wireless links have 0.1% random error rate.

Table 5: Throughput comparison between Reno and Snug-Reno over lossy links

Reno source	Snug-Reno source	Reno mean throughput	Snug-Reno mean throughput
3	17	950.25	951.78
5	15	950.71	951.67
10	10	949.97	950.94
15	5	949.31	951.77
17	3	950.32	951.80

The number of Reno and Snug-Reno sources varies. The total number of all sources is 20. Mean throughput is in units of kbits. Wireless links have 0.1% random error rate.

5 Conclusions

In this study, we have proposed a base station-based scheme for both TCP Vegas and TCP Reno, called Snug-Vegas and Snug-Reno, respectively, to improve the TCP performance in the heterogeneous network consisting of wired and wireless links. With the ability of random loss detection, Snug-Vegas and Snug-Reno react appropriately to the loss that is either caused by network congestion or transmission error, and consequently enhances the goodput of a connection over heterogeneous networks. Our simulations show that our proposed mechanism is a viable solution to the TCP performance degradation in wireless IP communications. However, there is still room for improvement. The bandwidth of the bottleneck link is under-utilised when the random loss rate is high; therefore, a new approach of the fast recovery mechanism would be one of our future works. Also, our future research in this direction would be to improve Snug-Vegas and Snug-Reno so that the sender could further differentiate various types of wireless packet losses such as losses caused by random errors, fading, and mobile handoff processes.

6 References

1 Postel, J.: 'Internet protocol'. IETF RFC 791, Sep. 1981
2 Postel, J.: 'Transmission control protocol'. IETF RFC 793, Sep. 1981
3 Lakshman, T.V., and Madhow, U.: 'The performance of TCP/IP for networks with high bandwidth-delay products and random loss', *IEEE/ACM Trans. Netw.*, 1997, **5**, (3), pp. 336–350
4 Balakrishnan, H., Padmanabhan, V.N., Sechan, S., and Katz, R.H.: 'A comparison of mechanisms for improving TCP performance over wireless links', *IEEE/ACM Trans. Netw.*, 1997, **5**, (6), pp. 756–769
5 Jacobson, V.: 'Modified TCP congestion avoidance algorithm'. Tech. Rep., Apr. 1990
6 Brakmo, L.S., and Peterson, L.L.: 'TCP Vegas: end-to-end congestion avoidance on a global internet', *IEEE J. Sel. Areas Commun.*, 1995, **13**, (8), pp. 1465–1480
7 Fu, C.P., and Liew, S.C.: 'TCP Veno: TCP enhancement for transmission over wireless access networks', *IEEE J. Sel. Areas Commun.*, 2003, **21**, (2), pp. 216–228
8 Lee, C.L., Liu, C.F., and Chen, Y.C.: 'On the use of loss history for performance improvement of TCP over wireless networks', *IEICE Trans. Commun.*, 2002, **E85-B**, (11), pp. 2457–2467
9 Namda, S., Ejzak, R., and Doshi, B.T.: 'A retransmission scheme for circuit-mode data on wireless links', *IEEE J. Sel. Areas Commun.*, 1994, **12**, (8), pp. 1338–1352
10 Ayanoglu, E., Paul, S., LaPorta, T.F., Sabnani, K.K., and Gitlin, R.D.: 'AIRMAIL: A link-layer protocol for wireless networks', *Wirel. Netw.*, 1995, **1**, pp. 47–60
11 Bansal, D., Chandra, A., and Shorey, R.: 'An extension of the TCP flow control algorithm for wireless networks'. Proc. IEEE ICPWC'99, Jaipur, India, Feb. 1999, pp. 207–210
12 Bakre, A.V., and Badrinath, B.R.: 'Implementation and performance evaluation of indirect TCP', *IEEE Trans. Comput.*, 1997, **46**, (3), pp. 260–278
13 Yavatkar, R., and Bhagawat, N.: 'Improving end-to-end performance of TCP over mobile internetworks'. Proc. IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, Dec. 1995, pp. 146–152
14 Brown, K., and Singh, S.: 'M-TCP: TCP for mobile cellular networks', *ACM SIGCOMM Comput. Commun. Rev.*, 1997, **27**, pp. 19–43
15 Balakrishnan, H., Sechan, S., Amir, E., and Katz, R.H.: 'Improving TCP/IP performance over wireless networks'. ACM MOBICOM'95, Berkeley, CA, Nov. 1995, pp. 2–11
16 Samaraweera, N., and Fairhurst, G.: 'Reinforcement of TCP error recovery for wireless communcation', *ACM SIGCOMM Comput. Commun. Rev.*, 1998, **28**, pp. 30–38
17 Bakre, A., and Badrinath, B.R.: 'I-TCP: indirect TCP for mobile hosts'. Proc. ICDCS'95, Vancover, BC, Canada, May 1995, pp. 136–143
18 Chan, Y.C., Chan, C.T., and Chen, Y.C.: 'RedVegas: performance improvement of TCP Vegas over heterogeneous networks', *IEICE Trans. Fundam.*, 2004, **E87-A**, (7), pp. 1672–1679
19 Lefevre, F., and Vivier, G.: 'Understanding TCP's behavior over wireless links'. Proc. Communications Vehicular Technology, SCVT-2000, Lewer, Belgium, 2000, pp. 123–130
20 Hoe, J.C.: 'Start-up dynamics of TCP's congestion control and avoidance schemes'. Master's thesis, MIT, 1995
21 Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: 'TCP selective acknowledgement options', *Internet Draft, Apr. 1996*
22 Feng, W., and Tinnakornsrisuphap, P.: 'The failure of TCP in high-performance computational grids'. SC 2000: High-performance Networking and Computing Conf., Dallas, Tx, USA, Nov. 2000

23 Veres, A., and Boda, M.: 'The chaotic nature of TCP congestion control'. IEEE INFOCOM'2000, Tel Aviv, Israel, Mar. 2000, Vol. 3, pp. 1715–1723
24 http://www.isi.edu/nsnam/ns/
25 Floyd, S., and Paxson, V.: 'Difficulties in simulating the internet', *IEEE/ACM Trans. Netw.*, 2001, **9**, (4), pp. 392–403
26 Mo, J., La, R.J., Anantharam, V., and Walrand, J.: 'Analysis and comparison of TCP Reno and Vegas'. IEEE INFOCOM'99, New York, USA, Mar. 1999, Vol. 3, pp. 1556–1563
27 Lai, Y.C., and Yao, C.L.: 'Performance comparison between TCP Reno and TCP Vegas'. IEEE ICPADS'2000, Iwate, Japan, Jul. 2000, pp. 61–66
28 De Vendictis, A., Baiocchi, A., and Bonacci, M.: 'Analysis and enhancement of TCP Vegas congestion control in a mixed TCP Vegas and TCP Reno network scenario', *Perf. Eval.*, 2003, **53**, pp. 225–253
29 Jain, R., Chiu, D., and Hawe, W.: 'A quantitative measure of fairness and discrimination for resource allocation in shared computer systems'. DEC, Research Report TR-301, Sep. 1984