Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

TCP-Rab: a receiver advertisement based TCP protocol

TANG Xu-hong (汤旭红)[1], LIU Zheng-lan (刘正蓝)[†2], ZHU Miao-liang (朱淼良)[2]

([1]*Navigation Department, Shanghai Maritime University, Shanghai 200135, China*)

([2]*Computer College, Zhejiang University, Hangzhou 310027, China*)

[†]E-mail: lzlan@163.com; lzlan76@gmail.com

Received June 19, 2003; revision accepted July 15, 2003

Abstract: With the evolution of network technologies, the deficiencies of TCP protocol are becoming more and more distinct. The new TCP implementation, called Receiver Advertisement Based TCP (TCP-Rab) proposed here to eliminate these deficiencies, adopts two basic mechanisms: (1) Bandwidth Estimation and (2) Immediate Recovery. Bandwidth estimation is carried out at the receiver, and the result is sent back to the sender via the acknowledgments. Immediate Recovery guarantees high performance even in lossy link. Rab can distinguish the reason for packet loss, and thus adopt appropriate recovery strategy. For loss by network congestion, it will back off its congestion window, and for loss by link errors, it will recover the congestion window immediately. Simulations indicated that Rab has superiority over other TCP implementations.

Key words: Bandwidth Estimation, Immediate Recovery, TCP protocol
doi:10.1631/jzus.2004.1352 **Document code:** A **CLC number:** TP393.04

INTRODUCTION

The Transport Control Protocol (TCP) provides end-to-end reliable connection-oriented service over heterogeneous networks. After the famous 1986 network collapse due to network congestion, TCP congestion control mechanisms were introduced in TCP protocol. The congestion control mechanisms in early TCP, known as TCP-Tahoe, are discussed in Congestion Avoidance and Control (Jacobson, 1988). Tahoe introduces two basic mechanisms: slow start and congestion avoidance. However, with the emergence of some new network applications, these mechanisms bring some deficiencies, such as low link utilization, fluctuating throughput, etc. Later, there was a new implementation of TCP: Reno (Jacobson, 1990). Reno introduces two other basic mechanisms: fast retransmission and fast recovery, which are triggered by the source after receiving three dupACKs (duplicate acknowledgements). In Reno, the packet that is acknowledged three times is retransmitted before timeout, the congestion window (cwnd) is halved and the connection enters 'congestion avoidance' phase. This action is the key difference from that of Tahoe, which set the 'cwnd' to one packet after three dupACKs.

Presently, the network applications have changed a lot in many aspects, such as data transfer over paths with ever increasing bandwidth-delay product, quality of service (QoS) requirements for interactive traffic and communication over wireless links, which make the incompetence of the TCP stand out.

The congestion control mechanism of TCP is in fact a sliding window mechanism, in which packet loss means network congestion. When the TCP detects a packet loss, it deems that the network is congested, and therefore takes measures to recover its cwnd. Under the condition of wireless

links, packet loss may be caused by link error, not by network congestion. In this case, the link utilization will decrease if the congestion control algorithm is applied. In the links with high bandwidth-delay product, the "halving congestion window" mechanism of TCP will also decrease the link utilization greatly. Because of the bad performance of TCP in the wireless links, in the links with high bandwidth-delay product, and in the interactive traffic applications, it is necessary to develop a new TCP protocol, which is the goal of this paper.

RELATED WORKS

The evolution of wireless communication technologies led to increasing application of communication over wireless links. TCP research associated with wireless communication is also increasingly becoming a hotspot of research.

TCP protocol associated with wireless communication has improved in three key aspects (Tsaoussidis and Matta, 2002): (1) the protocol's error detection mechanism, which should distinguish different types of errors and be applicable for heterogeneous wired/wireless environments, (2) the error recovery mechanism, which should consider the distinctive characteristics of wireless networks such as transient or burst errors due to handoffs or fading channels, and (3) the protocol strategy, which adopts different tactics for different errors. Among them, the primary one is the error detection mechanism.

In order to improve the throughput and energy efficiency of TCP in mobile nodes, Tsaoussidis and Badr (2000) proposed a new TCP implementation named TCP-Probing, wherein a probe mechanism is grafted to the TCP and is coupled with an additional tactic called 'Immediate Recovery'. When a packet loss is detected, the sender initializes a probing cycle, in which the sender will not send any normal packets, but only some probing segments. After the probe cycle, the two RTTs measured in the probe cycle are compared. If both lie in the range of [best RTT, last RTT], 'Immediate Recovery' is applied. Otherwise, the sender enters a slow start phase as in Tahoe. We found that in the connections whose RTTs are small, and the link errors are comparatively dense, the throughput gained by this mechanism is outstanding. However, in other cases, such as in connections whose RTTs are large, or whose congestion is frequent, the link utilization obtained by TCP-Probing is poor, and so is the throughput. The reason is that two RTTs will be wasted in the probe cycle after every packet loss which may be induced by network congestion.

Other researchers proposed some schemes based on base station (Zhang and Tsaoussidis, 2001; Parsa and Aceves, 1999; Bakre and Badrinath, 1995; Brown and Singh, 1997). An example is WTCP (Ratnam and Matta, 1998). It can distinguish the types of errors in the border of the wired/wireless links (i.e. the base station). The base station buffers all the packets from the senders, so it can recover the loss in the wireless links locally. The prominent point of this mechanism is that it can definitely distinguish the types of errors (i.e. the packet loss that happened in the wired or wireless links), so it can adopt different recovery strategies. However, there are negative effects nearly in all the mechanisms based on base station: (1) the TCP connection is split into two connections, one from the sender to the base station, the other from the base station to the receiver, so the syntax of TCP is destroyed; (2) the base station needs huge buffers to store the packets from all the senders, which causes the problem of scalability.

Casetti *et al*.(2002) proposed a new TCP implementation TCP-Westwood, in which the sender estimates the available bandwidth by measuring and low-pass filtering the returning rate of acknowledgments. The estimated bandwidth is then used to set the 'cwnd' and the 'ssthresh' (slow start threshold) after a congestion epoch. Our research indicated that TCP-Westwood achieves good performance in certain conditions. Its prominent point is it requires only modification in the sender, so it can be gradually transited from the present TCP. However, there is also some shortcoming in Westwood: (1) Westwood assumes that the sizes of all the packets are identical, however, in the cases

that the sizes are not identical, when three dupACKs are received, the sender cannot know which packet triggers the dupACKs, nor the size of the packet; so the sender cannot estimate the bandwidth accurately; (2) Delayed ACKs and dupACKs have severely negative effect on the estimation result; (3) In the links whose RTTs are comparatively large, Westwood cannot react to the instantaneous dynamics of the bandwidth; (4) In the links where errors happen often, especially in the backward path, the loss of ACKs will cause the sender have no information for Bandwidth Estimation, and thus cannot estimate the bandwidth accurately.

In this paper, we proposed another new TCP implementation named Receiver Advertisement Based TCP (TCP-Rab). It introduces two basic mechanisms: Bandwidth Estimation and Immediate Recovery, and can distinguish the types of errors by comparing the RTT of the dupACKs with the one that is used by the present RTO timer, and then adopts different recovery strategies for different loss types. In this scheme, the Bandwidth Estimation is carried out in the receiver, and the estimated bandwidth can be delivered back to the source via ACKs by setting the AdvertisedWindow field equal to min(AdvertisedWindow, RTTmin*BWE). Qualitative analysis and simulation experiments indicated that TCP-Rab can be applied not only in traditional network environments, but can also achieve outstanding performance even in links with high delay-bandwidth product and wireless links.

ALGORITHM OF TCP-RAB

In this section, we will discuss the implementation of TCP-Rab. It can track the dynamics of the network, estimate the available bandwidth, and further adjust the congestion window of the TCP connection according to the estimation result.

Design issues

The number of packets the TCP can send in one RTT is decided by the connection's sending window, which is the minimum of congestion window (cwnd) and advertised window (awnd). A key design aspect of TCP-Rab is that it dynamically adjusts the value of AdvertisedWindow field of the ACKs, carries out Bandwidth Estimation in the receiver, and delivers the estimation result back to the source via ACKs by setting the AdvertisedWindow field equal to min(AdvertisedWindow, RTTmin*BWE). The sender then properly sets the 'cwnd' and 'ssthresh' according to the AdvertisedWindow field of the ACKs. Rab maintains the syntax of TCP, which has huge difference from that of schemes based on base station.

TCP-Rab has two basic mechanisms: receiver based Bandwidth Estimation and Immediate Recovery. The receiver based Bandwidth Estimation overcomes the shortcomings of Westwood, and thus can be applied not only in the traditional links, but also in the links with high bandwidth-delay product or high error rate. In the case of packet loss due to link error, the Immediate Recovery will recover the sending window immediately, thus achieves high link utilization.

Congestion control mechanism of receiver

TCP-Rab carries out Bandwidth Estimation in the receiver. This mechanism is more accurate and easier to implement compared with Bandwidth Estimation in the sender as in Westwood.

The cost of TCP-Rab is that it needs two additional variables: lastEstTime and recvdBytes, which denote the time of the last estimation and the received bytes since the last estimation respectively. Whenever the receiver sends an ACK according to some strategy (for example, delayed ACK), it will execute its congestion control algorithm, as depicted by the pseudocode in Fig.1.

```
When an ack is to be sent
BWE=recvdBytes/(now−lastEstTime);
recvdBytes=0;
lastEstTime=now;
awnd=min(awnd′, BWE*RTTmin);

Where:
    BWE: the estimated bandwidth;
    now: the present time;
    awnd′: the awnd as in Reno;
    awnd: advertised window in ACK;
    RTTmin: the minimum of RTT detected till present
```

Fig.1 Bandwidth estimation of the receiver

Congestion control mechanism of sender

According to Fig.1, the receiver implements only a coarse-granularity congestion control algorithm, which is not sufficient for effective congestion control. So the sender should do fine-granularity congestion control. The congestion control mechanism of sender includes two components: Bandwidth Estimation and Recovery Strategy.

1. Bandwidth estimation

In order to prevent throughput fluctuation caused by burst traffic, it is important to use a low-pass filter to obtain the low-frequency components of the available bandwidth. It is known that congestion occurs whenever the low-frequency input traffic rate exceeds the link capacity (Hoe, 1996). Therefore it is useful to track only low-frequency components of the available bandwidth. In this scheme, we use a simple low-pass filter. Meanwhile, this mechanism also distinguishes the types of packet loss (by link errors or by network congestion). The pseudocode is shown in Fig.2. Each time an ACK is received, the sender executes these codes, and 'ssthresh' may be updated, which assures the instantaneity of bandwidth changes.

First, the sender checks if the ACK is dupACK and if the RTT detected by this packet is less than the one that is used by the current RTO timer. If so, Rab deems that the possible packet loss is caused by link error, not by network congestion, the Bandwidth Estimation procedure will terminate, and the estimated bandwidth and 'ssthresh' will not be updated. If not, the source will update the 'ssthresh'. Therefore, in the case of link errors, the sending window will recover immediately. In other cases, the Estimated Bandwidth and 'ssthresh' will be updated properly.

```
When an ack is received
if (the ack is dupACK AND rtt<t_rtt_)
      return;
endif
sample_WND=ack->awnd();
WND=0.9*WND+0.05*
        (sample_WND+last_sample_WND);
ssthresh=WND/pktsize;
last_sample_WND=sample_WND;

where:
      rtt: the round trip time of this pkt;
      t_rrt_: the round trip time used by present RTO timer;
      sample_WND: the advertised window from the re-
            ceiver;
      last_sample_WND: the last sample_WND;
      WND: the estimated sending window of sender.
```

Fig.2 Bandwidth estimation of the sender

2. Recovery strategy

For the sake of simplification, Rab assumes the number of packet as the unit of 'cwnd' and 'ssthresh', and assumes that all the packets have the same size. This presumption is essentially different from that of Westwood. In Westwood, if the size of packets is not identical, the source cannot know which packet triggers the dupACK, and will not be informed of the packet size, and thus cannot estimate the bandwidth accurately. In Rab, the Bandwidth Estimation mechanism does not require that all the packets should have the same size. Here we make this assumption just for simplification.

Now, the source knows the available bandwidth for the TCP connection (by means of Bandwidth Estimation). Let us describe in the following paragraphs how the estimated bandwidth is used to properly set 'cwnd' and 'ssthresh' after a packet loss.

The results of some researches (Hoe, 1996) indicated that the optimal value for 'ssthresh' is the number of segments in flight in a pipe when TCP rate equals the available bandwidth. That is, when its transmission window is equal to the available bandwidth-delay product. The recovery strategies of Rab are depicted in Fig.3 and Fig.4.

In the case of receiving three dupACKs, Rab will retransmit the lost packet immediately as in Reno, and then set 'ssthresh' to its available band-

```
if (3 dupacks are received)
   ssthresh=WND/pktsize;
   if (ssthresh<2)
         ssthresh=2;
   endif
   if (cwnd>ssthresh)
         cwnd=ssthresh;
   endif
endif
```

Fig.3 Three dupACKs action

```
if (RTO expires)
    ssthresh=WND/pktsize;
    if (ssthresh<2)
        ssthresh=2;
    endif
    cwnd=1;
endif
```

Fig.4 Timeout action

width, and 'cwnd' to 'ssthresh'. From Fig.2, we know that the 'ssthresh' has already been set properly (will not be changed in the case of link errors, and will be updated in other cases). So in the case of packet loss by link errors, the sending rate of the source can be recovered immediately. In the case of packet loss by network congestion, the sending rate of the source can be set to the right value corresponding to its available bandwidth.

In the case of timeout, Rab deems the network has been seriously congested, so it sets 'cwnd' to one packet.

SIMULATION EXPERIMENTS

We verified Rab in ns2, and compared the result with the other four TCP implementations [Westwood, Reno, SACK (Floyd *et al*., 2000) and Tahoe]. To avoid the influence of each other, each time, we only configured one TCP implementation in the source, and delayed-ACK was always configured in the receiver. The simulation topology is shown in Fig.5, the bandwidth of the link from the sender to the router was 200 Mbps, and its one-way delay was 5 ms. The bandwidth and one-way delay of another link, from the router to the receiver, were set differently in each simulation experiment for different purposes. The TCP flow in all the experiments was FTP data flow.

Through simulation experiments, we studied the performance of Rab under different conditions, such as throughput in different bottleneck bandwidth, different RTT, and different link error rate. We also verified the accuracy of bandwidth estimation, and observed the adjustment of 'cwnd' and 'ssthresh'.

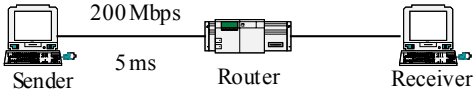

Fig.5 Simulation topology

RESULT AND ANALYSIS

Bandwidth Estimation accuracy

In this simulation experiment, we configured the source with Rab and Westwood respectively to verify the accuracy of bandwidth estimation. For the connections whose bottleneck bandwidth and RTT are small, 'cwnd' and 'ssthresh' of the TCP are comparatively small, so, even if packet loss occurs, they can resume soon. Therefore, we only inspected Bandwidth Estimation accuracy of the two implementations in the case of links with high bandwidth-delay product. The bottleneck bandwidth of the connection was set to 100 Mbps, and the RTT was set to 500 ms.

To simulate the bandwidth dynamics, this experiment introduced two ON/OFF UDP flows, whose sending rates were 20 Mbps and 30 Mbps respectively while ON. The TCP connection sent data throughout the simulation. Both UDP flows started in OFF state, after 50 s, the first UDP connection was turned ON, joined by the second one at 100 s; the second UDP connection followed an OFF-ON-OFF pattern at times 150 s, 250 s and 350 s; at 400 s the first UDP connection was turned OFF. The simulation experiment lasted 500 seconds.

The simulation results of Bandwidth Estimation are depicted in Fig.6.

The scenario above is intended to demonstrate the accuracy of Bandwidth Estimation in Rab and Westwood when subjected to "step" and "impulse" stimuli. The simulation result shown in Fig.6 and Fig.7 indicates that in the case of links with high bandwidth-delay product, the estimation result of Westwood has huge difference from the actual bandwidth, so Westwood cannot estimate the available bandwidth accurately, and we conclude that Westwood cannot be applied in the links with high bandwidth-delay product. However, the estimation result of Rab is almost the same as the actual

one. The reason for this fact is described in detail in the following section.

Setting 'cwnd' and 'ssthresh'

The setting of 'cwnd' and 'ssthresh' is very critical in the TCP congestion control mechanism. If 'cwnd' and 'ssthresh' are too small, the sending rate of TCP connection will be smaller than what the network can supply, so the link utilization decreases. If 'cwnd' and 'ssthresh' are too large, the sending rate of TCP connection will surpass what the network can supply, which may induce frequent packet loss, and thus result in low link utilization too.

In the aforementioned simulation experiments, we also observed the dynamics of 'cwnd' and 'ssthresh' in Rab, Westwood and Reno, as depicted in Fig.7.

We found that the setting of 'cwnd' and 'ssthresh' in Westwood was almost the same as in Reno, when the available bandwidth reduced, they responded rapidly, so, 'cwnd' and 'ssthresh' were tuned downwards immediately, which is the law of Multiplicative Decrease of TCP; however, when the available bandwidth increased, due to the law of Additive Increase of TCP, 'cwnd' increased slowly, and 'ssthresh' remained unchanged until the next time of packet loss, as depicted in Fig.7b and 7c. So, Westwood and Reno cannot track the instantaneous

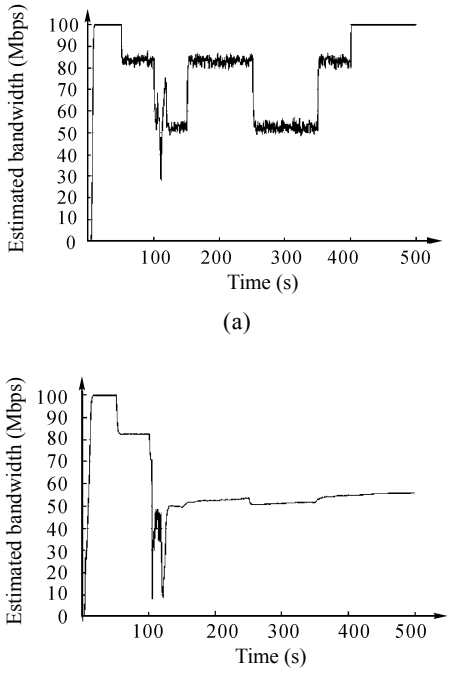

Fig.6 Result of Bandwidth Estimation
(a) Rab; (b) Westwood

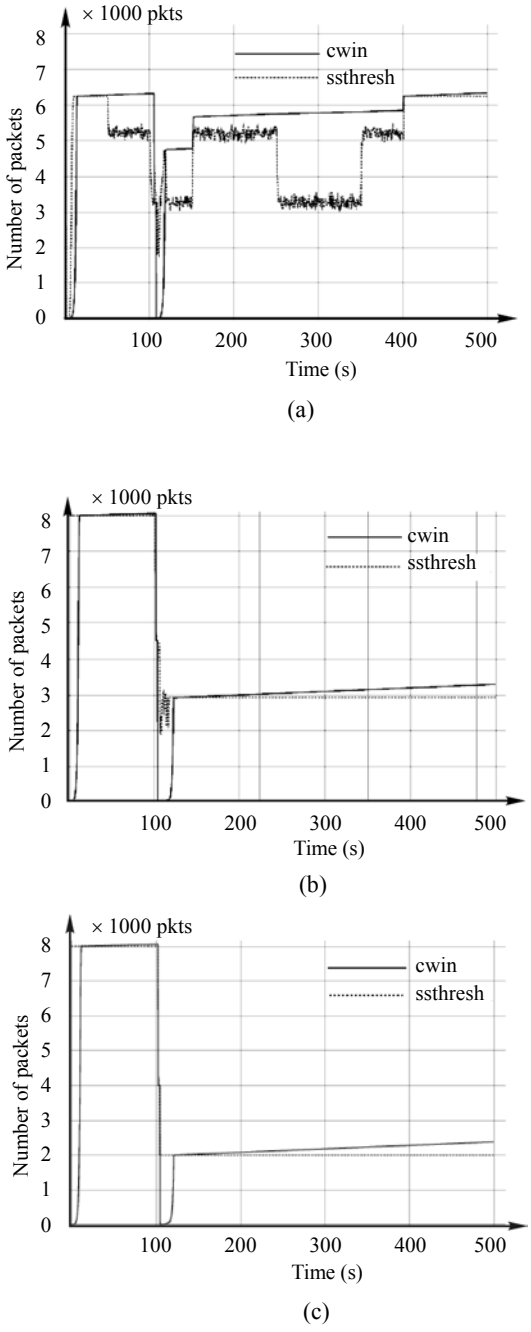

Fig.7 Variety of cwnd and ssthresh
(a) Rab; (b) Westwood; (c) Reno

dynamics of the available bandwidth, and thus cannot utilize the available bandwidth efficiently.

However, in Rab, the scenario is totally different. Upon receiving an acknowledgment, the source will possibly update its 'ssthresh' (refer to the algorithm in Fig.2) immediately, and thus affect the mode of adjustment of 'cwnd' (additively or exponentially[1]). So, when the available bandwidth varies, increases or decreases, Rab can respond rapidly (through the setting of 'ssthresh'). Due to the Immediate Recovery mechanism of Rab, 'ssthresh' is always set to the value corresponding to the available bandwidth, as shown in Fig.7. The difference in strategies of updating 'cwnd' and 'ssthresh' is also the main reason for the disparity in Bandwidth Estimation accuracy depicted in Fig.6.

Impact of bottleneck bandwidth

Nowadays, the bandwidth that the link can supply is becoming increasingly larger, so, a question arises: can TCP fully utilize the bandwidth? As we know, due to its congestion control mechanism, TCP is very sensitive to packet loss, for example, in Reno, the "halving congestion window" mechanism results in much lower link utilization than what the link can provide, especially in the case of links with high bandwidth-delay product or links with dense error rate.

We designed an experiment to compare the throughput gained by the five TCP implementations in the topology shown in Fig.5. We experimented with each implementation separately. The RTT of the connection was fixed at 100 ms; the bottleneck bandwidth was varied from 10 Mbps to 200 Mbps. The simulation result is shown in Fig.8.

The simulation result indicated that in Tahoe, Reno and SACK, when the bottleneck bandwidth is more than 40 Mbps, the throughput remains almost unchanged, which is about 30 Mbps. Most of the bandwidth is wasted. In Westwood, when the bottleneck bandwidth is less than 130 Mbps, TCP can achieve good performance. But, when the bottleneck bandwidth increases further, the achieved performance drops sharply. This fact also proves that Westwood cannot be applied in the links with high bandwidth-delay product. However, in Rab, the result is thoroughly different. From 10 Mbps to 200 Mbps, however the bottleneck bandwidth varies, the throughput the connection gains is almost identical to what the network can supply.

Impact of RTT

Undoubtedly, RTT affects the throughput of TCP connection greatly, especially in the case of network congestion or link error. Packet loss will cause the source to spend some RTTs to recover to the normal sending rate, the larger the RTT is, the more time is spent in the recovery.

In the simulation topology shown in Fig.5, the bottleneck bandwidth was fixed at 50 Mbps, RTT was varied from 50 ms to 600 ms, and the link error rate (packet error rate, PER) was set to 10^{-6}. The

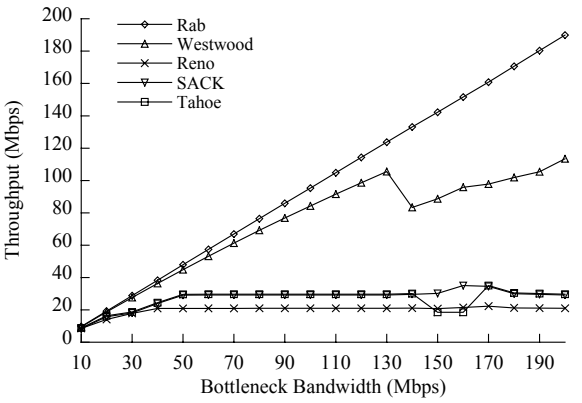

Fig.8 Throughput vs bottleneck link speed

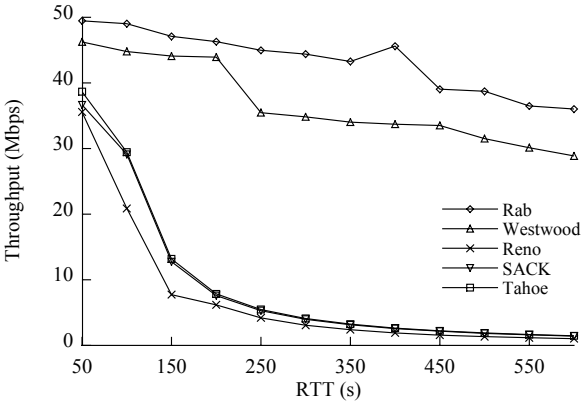

Fig.9 Throughput vs RTT

[1]The update of 'ssthresh' may change the phase of the TCP connection: slow start or congestion avoidance, and thus change the increment mode of 'cwnd': additively or exponentially.

simulation result is depicted in Fig.9.

As the RTT increases, the throughput gained in Tahoe, Reno and SACK declines drastically due to the effect of congestion control mechanism in the case of packet loss, where the frequent packet loss results in shrank 'cwnd' and cannot be recovered rapidly. Although the performance gained by Westwood is comparatively fine here, through simulation, we found that it also declines drastically in the case of denser link error, especially in the backward path. The reason is as mentioned before, the loss of packet, especially the loss of ACK, result in inaccuracy of Bandwidth Estimation in Westwood, which is much less than the actual one. Unlike in Westwood, packet loss cannot affect the accuracy of Bandwidth Estimation in Rab, so, even in the links with dense link error rate, the throughput it gains also corresponds to the available bandwidth.

Impact of link error rate

For TCP, the network is much like a "black box", the information about the network the TCP can get is very limited, what it get almost all comes from the ACK. Tahoe, Reno and SACK cannot distinguish the reasons for packet loss (network congestion or link errors). They only consider packet loss when three dupACKs are received or timeout occurs. So their recovery strategies have no difference, and are all "back off congestion window".

One of the key reasons why Rab is suitable for communications over wireless links is Rab's ability to distinguish the packet loss reasons (please refer to the algorithm in Fig.2), so Rab can adopt different recovery strategies for different types of packet losses. For packet losses due to link error, 'cwnd' and 'ssthresh' will not shrink, which result in high link utilization even in the links with dense error rate.

We examined the performance of Rab under condition of different link error rate. The bottleneck bandwidth of the connection was fixed at 50 Mbps, and RTT was fixed at 200 ms. We modeled the wireless links by using a two-state Markov channel (Bhagwat *et al*., 1996), which was in either good state or bad state. In the good state, the packet error rate (PER) was zero (i.e. no packet loss due to link error will occur), and in the bad state, the packet error rate varied from 10^{-6} to 0.9. The time spent in each state was modeled by an exponential distribution with mean good state duration fixed at 10 s, and the mean bad state duration was fixed at 1 s. The simulation duration was 100 seconds, and the result is depicted in Fig.10.

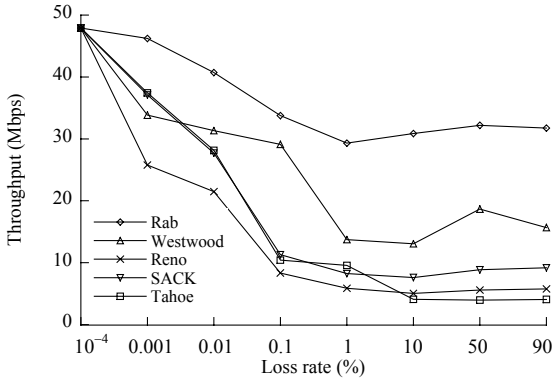

Fig.10 Throughput vs link loss rate

In the case of link error, the performance of Westwood is much like the one of Tahoe, Reno and SACK, which also proves the judgment that Westwood is not suitable for communications over wireless links. Due to the virtue of the algorithm, Rab is not sensitive to link errors, so even in the links with dense error, it can still achieve acceptable performance. Rab is applicable for communications over wireless links.

CONCLUSION AND FUTURE WORK

This paper introduced a new implementation of TCP, Rab. It can achieve satisfactory performance not only in traditional network environments, but also in the links with high band-delay product or links with dense link error. The outstanding performance that Rab achieves mostly comes from two basic mechanisms: Bandwidth Estimation and Immediate Recovery. After packet loss, due to network congestion or link error, Rab immediately sets 'cwnd' and 'ssthresh' to the values corre-

sponding to the available bandwidth. So satisfactory performance is assured. Our simulation showed that Rab could achieve better performance than the other four implementations under various conditions.

In Rab, we adopt a simple low-pass filter which is easy to implement but may not be effective enough, more effective filters may be worth further research in future work. Furthermore, as QoS is increasingly required, how to guarantee QoS in the TCP implementation is also worth further research.

References

Bakre, A., Badrinath, B., 1995. I-TCP: Indirect TCP for Mobile Hosts. Proceedings of the IEEE ICDCS'95, Vancouver, Canada, p.136-143.

Bhagwat, P., Bhattacharya, P., Krishna, A., Tripathi, S., 1996. Enhancing throughput over Wireless LANs Using Channel State Dependent Packet Scheduling. INFOCOM'96, San Francisco, California, p.1133-1140.

Brown, K., Singh, S., 1997. M-TCP: TCP for mobile cellular networks. *Computer Communication Review*, **27**(5):19-43.

Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M.Y., Wang, R., 2002. TCP Westwood: end-to-end congestion control for wired/wireless Networks. *Wireless Networks Journal*, **8**(1):467-479.

Floyd, S., Mahdavi, J., Mathis, M., Podolsky, M., 2000. An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC2883.

Hoe, J.C., 1996. Improving the Start-up Behavior of A Congestion Control Scheme for TCP. Proceedings of SIGCOMM'96. Standford, p.270-280.

Jacobson, V., 1988. Congestion avoidance and control. *ACM Computer Communications Review*, **18**(4):314-329.

Jacobson, V., 1990. Berkeley TCP Evolution from 4.3-Tahoe to 4.3 Reno. Proceedings of the 18th Internet Engineering Task Force, University of British Colombia, Vancouver, p.365-376.

Parsa, C., Aceves, G., 1999. Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media. IEEE International Conference on Network Protocols. Toronto, Canada, p.213-221.

Ratnam, K., Matta, I., 1998. WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links. Proceedings of the Third IEEE Symposium on Computer and Communications, Athens, Greece, p.74-78.

Tsaoussidis, V., Badr, H., 2000. TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains. IEEE International Conference on Network Protocols, Osaka, Japan, p.12-21.

Tsaoussidis, V., Matta, I., 2002. Open issues on TCP for mobile computing. *Wireless Communication Mobile Computers*, **2**(2):3-20.

Zhang, C., Tsaoussidis, V., 2001. TCP-real: Improving Real-time Capabilities of TCP over Heterogeneous Networks. IEEE/ACM NOSSDAV, New York, p.189-198.