

## **EJTCP: Enhanced Jitter-based TCP for Wireless Broadband Networks\***

ERIC HSIAO-KUANG WU, YU-CHEN HUANG AND GUI-KUI CHANG

*Department of Computer Science and Information Engineering*

*National Central University*

*Chungli, Taoyuan, 320 Taiwan*

*E-mail: hsiao@csie.ncu.edu.tw*

*E-mail: {yuchen; slayer}@wmlab.csie.ncu.edu.tw*

TCP is one of the most important and widely adopted transport protocols at present. With the advent of advanced wireless broadband technologies, TCP must be properly tuned and enhanced from traditional networks made up of purely wired links to wired-cum-wireless networks. As a result of high bit error rate (BER) in wireless links, TCP halves down its congestion window unnecessarily caused by random packet loss event continuously, and the performance is significantly degraded in wireless networks. The problem about differentiating congestion loss and random loss has been investigated recently by a number of wireless TCP solutions, such as JTCP (jitter-based TCP). While the existing solutions are accomplishing fundamental efforts to distinguish packet losses, some unavoidable events may occur to make TCP timeout frequently. Hence, we amend JTCP scheme to design a smooth transmission scheme to increase the correctness of loss distinction and avoid burst transmission. Besides, a novel channel aware scheme is designed to deal with dynamic modulation changing in 802.16 Wireless MAN. The experimental results show that our scheme, enhanced JTCP (EJTCP), unrolls significantly better performance than other algorithms over wired-cum-wireless network.

**Keywords:** congestion control, end-to-end control, jitter, transmission control protocol (TCP), wireless

### **1. INTRODUCTION**

#### **1.1 TCP Development and Congestion Control**

Transmission Control Protocol (TCP) [1] was designed as a reliable transport protocol to handle error conditions in an end to end manner when network hardware fails, or when networks become too heavily loaded. Through several modifications and improvements, it has become the most widely adopted transmission protocol to function well in traditional networks purely composed of wired links and stationary hosts. A dominating ratio (more than 90%) of Internet applications and services are built on top of TCP nowadays.

The congestion avoidance algorithm of TCP [2, 3] is designed for shunning congestion based on the additive increase and multiplicative decrease (AIMD) scheme. Fig. 1 illustrates the typical behavior of Reno version TCP with show start and congestion

---

Received August 22, 2005; revised November 21, 2005, February 6 & April 19, 2006; accepted June 14, 2006. Communicated by Yu-Chee Tseng.

\* This work was supported by Institute for Information Industry under the project "Advanced Mobile Context Aware Application & Service Technology Development Project" (96-EC-17-A-02-R7-0327) and Chungshan Institute of Science and Technology.

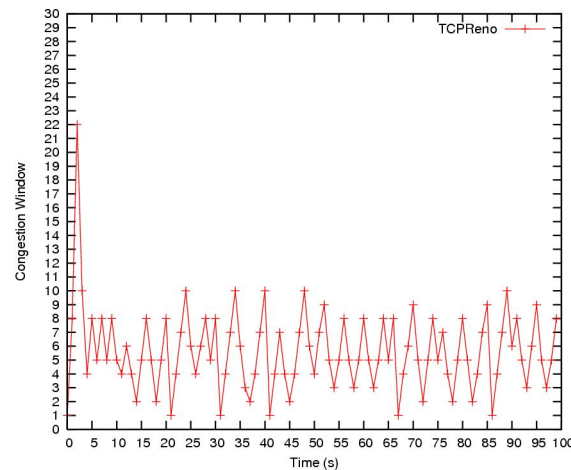


Fig. 1. Slow start and congestion avoidance of Reno.

avoidance phases for different objectives. After a connection is established, TCP starts up the slow start (SS) procedure for the initial transmission. In slow start phase, the congestion window of the sender will be incremented by one after receiving one acknowledgement (ACK) message. In other words, the congestion window (*cwnd*) increases exponentially by each round trip time (RTT). On the contrary, the congestion window in congestion avoidance phase is increased by at most one segment each round trip time. Whenever the sending rate is over the bottleneck capacity, some packets may be dropped in the bottleneck router. Whenever congestion occurs, the source may receive triple duplicate acknowledgements (TDACKs) to trigger the fast retransmit mechanism and halve down its *cwnd* to slow down the transmitted rate. As the network gets overloaded seriously, the sender may not receive any ACK for a certain periods and the sender will start up timeout procedure. The *ssthreshold* will be set to half of *cwnd* and *cwnd* be set to 1, and the slow start (SS) mechanism will be initiated again.

## 1.2 Wireless Network and 802.16 WirelessMAN

Wireless networks have been widely deployed and rapidly evolved in recently years. As wireless broadband technologies evolve from traditional GSM, 3G and IEEE 802.11 to IEEE 802.16, the wireless MAN (Wireless Metropolitan Area Networks) could bring the broadband services to the building or directly to the individual users. Emerging pervasive and ubiquitous computing technologies contribute the potential to make our lives easier. IEEE 802.16 network [4] one of the most challenging segments of wireless revolutions provide high transmitted rate and long transmitted range for users to access network, a viable alternative to the cable modem and DSL technologies.

The transmission mechanism of 802.16 MAC layer is similar to Data-Over-Cable Service Interface Specifications (DOCSIS) [5, 6]. Access and bandwidth allocation algorithms must accommodate hundreds of end-users with different QoS requirements. SSS issue the transmission requests at the contention period. Upon collecting all the requests, BS allocates the upload slots and download slots to that SSS. BS notes the information of

transmitted/received time in MAP and sends MAP to SSs. SSs start to transmit and receive data at the right time according MAP description.

In the physical layer of 802.16 networks, the transmission rate is not a constant value and it usually adapts to wireless propagation environments, *i.e.* interference, BER and power degradation. Currently, there exist three modulation options in 802.16, QPSK, 16-QAM and 64-QAM and the corresponding rates are 32MB, 64MB and 96MB. The adjusting mechanism is driven by the physical layer. As the environment of connection varies, the physical layer will adapt the modulation parameters accordingly based on BER, C/I and power degradation. Transmission parameters including modulation and coding schemes could be adjusted individually to each subscriber on a frame-by-frame basis.

### 1.3 TCP Over Wireless Network

TCP designed to deal with congestion problem usually performs well in traditionally wired environment. In a purely wired network, the packet losses usually are caused by congestion and TCP's congestion avoidance algorithm can handle this problem well in this environment. However, in wireless environment the non-congestion losses may occur due to random loss or collision loss. As a result of the higher bit error rate (BER) over wireless network, TCP will go through random loss event frequently [7]. According the design of TCP congestion algorithm, TCP will not be able to differentiate the loss caused by congestion or random loss. Once a wireless loss occurs, TCP diagnoses it as a congestion event and halves down its *cwnd*. The reduction or slow down reaction is unnecessary at this situation. Those unnecessary behaviors will diminish the performance of TCP and make the TCP's efficiency considerable degraded over wireless links. The distinction of packet losses has been a fundamental key issue for TCP solutions over wireless networks recently.

The fast growing deployment of wireless networks indicates that wireless links will play a key role for the new generation Internet. A new appropriate version of TCP will be designed and tuned for the wired-cum-wireless network. Furthermore, if random losses occur within a very short period, subsidiary events (wait to timeout; burst transmit) may take place immediately after the loss event. Those events will damage the performance of TCP significantly. Recently, a great amount of efforts have been proposed to distinguish the congestion loss and wireless loss. Very few algorithms have been able to judge the loss event exactly, and improve TCP's throughput effectively. However, in addition to distinguishing losses, a complete set of strategies for significant wireless losses are highly expected for the new generation wireless network. Last but not least, a good transport scheme could adjust TCP's transmission rate to avoid congestion when the bandwidth is reduced by adaptive modulation.

## 2. RELATED WORK

TCP performance problem over the wired-cum-wireless network has been investigated and addressed recently to alleviate poor end-to-end performance in wireless medium [8, 9]. They are roughly classified into two categories: (1) hiding non-congestion loss: link layer proposal and split-connection proposal (2) end-to-end approach [10, 11].

In this section, we are discussing the end-to-end approaches for the advantages of easy implementation and less overhead.

In order to maintain the TCP's end-to-end semantic, TCP can make good use of transmitted information obtained from propagation or embedded with other schemes to distinguish the lost events. Westwood TCP Westwood [12, 13] estimates the available bandwidth of bottleneck link by measuring the arrival rate of ACKs for discriminating the cause of packet losses. It uses the interval of continuous feedback ACKs and the packet size to hit the network capacity,  $SBW[k]$ .  $BWE[k]$ , a smoothed value is calculated by low-pass filtering the sequence of  $SBW[k]$ .

$$SBW[k] = \frac{pkt\_size}{this\_time - last\_ACK\_time},$$

$$BWE[k] = (1 - \alpha) \times \frac{SBW[k] + SBW[k-1]}{2} + \alpha \times BWE[k-1],$$

where  $pkt\_size$  denotes the packet size,  $this\_time$  denotes the received timestamp of latest ACK,  $last\_ACK\_time$  denotes the received timestamp of previous ACK and  $\alpha$  denotes the coefficient of low-pass filter. While TDACKs are detected, the source sets the  $ssthresh$  equal to the capacity of link and sets the  $cwnd$  equal to  $ssthresh$ . Therefore, it can work well than Reno by avoiding the blindly halving of the sending rate. Although Westwood tries to estimate the bandwidth of link, and passes through correction several times. However, it might overestimate the available bandwidth [14, 15]. The incorrect estimation could cause TCP Westwood to get the poor throughput over wireless environment at some times.

TCP Vegas and TCP Veno TCP Vegas [16] try to use BaseRTT that is the minimum RTT sample value observed over the duration of the connection to estimate the capacity of the TCP link. The sender can estimate the backlog at the queue denoted by  $N$  by the following equations:

$$Expected = \frac{cwnd}{BaseRTT}, \quad Actual = \frac{cwnd}{ActualRTT}, \quad Diff = Expected - Actual,$$

$$ActualRTT = BaseRTT + \frac{N}{Actual}, \quad N = Actual \times (ActualRTT - BaseRTT)$$

$$= Diff \times BaseRTT,$$

where  $Expected$  denotes the expected rate of link,  $Actual$  denotes the actual rate and  $ActualRTT$  denotes the real  $RTT$  we get at the moment.  $cwnd$  is the current TCP's congestion window size. Vegas attempts to keep  $N$  to a small range by adjusting the TCP window size proactively, thus avoiding packet loss caused by congestion. However, Vegas has serious problem for fairness [17] when there are coexisting other TCP version connections (*i.e.* Reno, Newreno) on its path. Since the algorithm of Vegas is less aggressive than other version, it will reduce its congestion window down to zero while competing with other TCP connection.

**TCP Veno** [18] enhances the mechanism of Vegas to judge the cause of loss. Veno sets a threshold, 3, to represent the status of network. If packet loss is detected and  $N > 3$ ,

that implies the connection is in bad status and the packet will be considered as a congestion loss; otherwise, it is regarded as a random loss. TCP VenO can reduce the damage from packet loss but it will misjudge when the loss occurs too closely or the random loss rate is high. Besides, the value of *BaseRTT* may not be the same for an on-going connection.

**JTCP** [19] was proposed to use the jitter ratio to predict the reason of packet loss, and further adapt its congestion control. One-way delay jitter ( $\Delta OWD$ ) [20] is defined as:

$$\Delta OWD^i = (R^i - S^i) - (R^{i-1} - S^{i-1}),$$

where  $i$  denotes the packet's index;  $S^i$  and  $R^i$  denote the sending time from sender and receiving time at receiver of  $i$ th packet. If  $\Delta OWD^i$  is greater than 0, then queuing delay of  $i$ th packet is larger than  $(i - 1)$ th packet with the same pack size and the same propagation path. In other word, bottleneck router processed cross traffic which arrived between  $i$ th and  $j$ th packets. In the following, we use one-way delay jitter to derive the jitter ratio.

Jitter ratio ( $j_r$ ) [21] is defined as the ratio of queued packet. As we have known, packet will be queued if the packet arrival rate is larger than the bottleneck link's capacity. Here we define variables first; let  $t_A$  be the inter-departure time of  $i$ th packet-pair, which is constituted by two consecutive packets.  $t_D$  denote inter-arrival time of  $i$ th packet-pair when they arrived the receiver. When bottleneck is congested, the inter-arrival time,  $t_D$ , is approximate to  $1/B$ , where  $B$  is the bottleneck link's capacity, and jitter ratio, denotes the ratio of queuing speed to the sending rate, can be derived as follows

$$\begin{aligned} j_r &= \frac{1/t_A - B}{1/t_A} \approx \frac{1/t_A - 1/t_D}{1/t_A} = \frac{(t_D - t_A)/(t_A \times t_D)}{1/t_A} = \frac{t_D - t_A}{t_D} \\ &= \frac{(R^i - R^{i-1}) - (S^i - S^{i-1})}{(R^i - R^{i-1})} = \frac{\Delta OWD^i}{(R^i - R^{i-1})}. \end{aligned}$$

In corporate with the behavior of TCP, JTCP adjust its congestion window every round-trip time. Therefore, the jitter ratio of sending packets during each round-trip time under congestion window,  $w$ , can be rewritten as:

$$\sum_{i=n-w}^{n-1} \Delta OWD^i = \sum_{i=n-w}^{n-1} ((R^i - R^{i-1}) - (S^i - S^{i-1})) = (R^{n-1} - R^{n-w}) - (S^{n-1} - S^{n-w}).$$

And the average jitter ratio is defined as:

$$j_r = \frac{(R^{n-1} - R^{n-w}) - (S^{n-1} - S^{n-w})}{R^{n-1} - R^{n-w}}.$$

It's obviously that congestion will occur after a particular queue gets full. Furthermore, the value of jitter ratio is proportional to the queuing speed. Hence, JTCP uses jitter ratio to compare with the threshold,  $k/w$ , where  $k$  is set to 1 by experiment. When TDACKs is detected and jitter ratio is estimated to be lager than  $1/w$ , the sender regards that its sending rate is larger than the bottleneck link's capacity and packet loss was

caused by congestion but channel loss. Although JTCP outperforms other schemes in throughput performance, it may make false-determine lost event under bursty cross traffic. In this article, we improve the accuracy of estimation to enhance the performance and further adapt to various wireless loss events. JTCP uses jitter ratio ( $k/cwnd$ ) to determine the relationship between transmission rates of JTCP and utilization of bottleneck's queue. However, this threshold may be incorrect under bursty cross traffic. The reason is discussed in [23], and EJTCP uses RTT to further filter noise in jitter ratio.

### 3. ENHANCED JTCP

We propose a novel scheme to amend the mechanism of JTCP for lost events and avert spare timeout caused by burst transmission. Besides, for the new generation wireless broadband network, 802.16 WiMAX network, we design a physical layer-aware rate control scheme to adjust the transmitted rate when the modulation transmission parameters change.

#### 3.1 JTCP Error Judgment

Although JTCP could distinguish congestion loss and wireless loss usually, we find that JTCP will make error conclusions in some situations, especially two random losses occur closely. We can simplify the formula of JTCP as follows:

$$jr = \frac{(R^i - R^{i-1}) - (S^i - S^{i-1})}{R^i - R^{i-1}} = 1 - \frac{S^i - S^{i-1}}{R^i - R^{i-1}}.$$

Jitter ratio denotes the relationship between inter-departure and inter-arrival time of two consequent packets. In Fig. 2, we illustrate how JTCP might false-estimate this relation. When the first TDACK event is detected, the sender will retransmit the lost one. During this time, if the second TDACK event occurs at the time which is close to the first one, the retransmitted packet will cause incorrect jitter ratio measurement. As a consequence, jitter ratio will collect false relationship between different packet-pairs. Therefore, the information in jitter ratio will be not able to capture the real relationship of the inter-arrival time.

This problem is caused by that the sender can not differentiate TDACKs responses to which particular packets. To cope with this problem, we modify TCP header accordingly. Two extra bytes will be utilized to record the sending and the receiving times in the revised TCP header, and the revised header overhead compared to original TCP header is about 10%. For the improvement of throughputs, the slight overhead in revised TCP header may be reasonable. In our scheme, we also take the RTT into account for the network condition determination. RTT will grow and decline according to the queue length of routers and it reflects the state of network so we can use RTT to have better knowledge of on-going network status. For each transmission period before the loss event, (from slow start to timeout), we can get the max RTT ( $\max\_rtt$ ) and a condition threshold ( $rtt\_thresholds$ ) by the following equation:

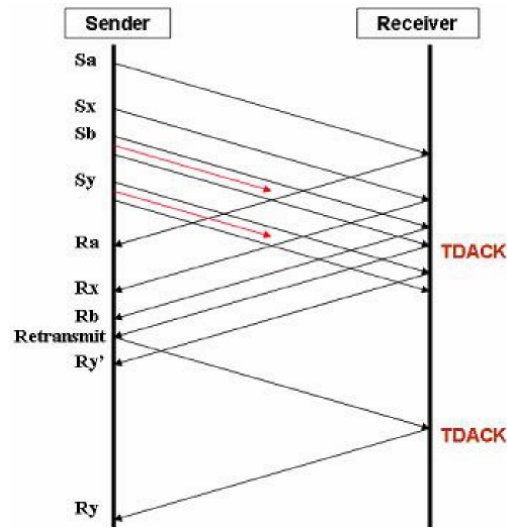


Fig. 2. JTCP error judgment.

$$rtt\_thresholds = \max\_rtt \times \alpha.$$

The major objective of this paper is to mend the weakness of JTCP, which is the false-estimation between congestion and random when cross traffic is bursty not fluid. In [23], the analysis shows us that one-way delay jitter of TCP packets will depend on cross traffic volume (on going connections), TCP traffic under the fluid cross traffic model but the noise will not be evaluated under bursty cross traffic. Under bursty cross traffic, JTCP will not be able to judge the relationship between the sending rate of TCP flow and residual bandwidth correctly and false-estimate the signal between congestion and random loss. This is caused by the noise which can be eliminated by larger congestion window at the risk of flooding the bottleneck. EJTCP utilizes RTTs to further filter the noise signal in the JTCP. Besides, the timeout problem of JTCP will be mitigated under bursty cross traffic model. Here, the  $\max\_rtt$  is initialized as zero, and  $\max\_rtt$  will be updated when jitter ratio is larger than  $k/cwnd$ , which is similar to JTCP. In other words, EJTCP could accumulate the knowledge of  $\max\_rtt$  by jitter ratio. When network background traffic is stationary, the  $\max\_rtt$  should be stationary as well.

As the current RTT value is over  $rtt\_thresholds$  and  $jr$  is greater than  $k/cwnd$ , we regard the loss as a congestion event. Then we can amend the determining conditions as:

```

If ( $jr > k/cwnd$  &&  $rtt > rtt\_thresholds$ )
     $ssthreshold = (1/2) \times cwnd$ ;
     $cwnd = ssthreshold$ 
Else
     $fast\_retransmission()$ ;
endif

```

The state transition diagram of the proposed scheme is illustrated as Fig. 3. [23] shows that one-way delay jitter values of TCP packets are determined by cross traffic;

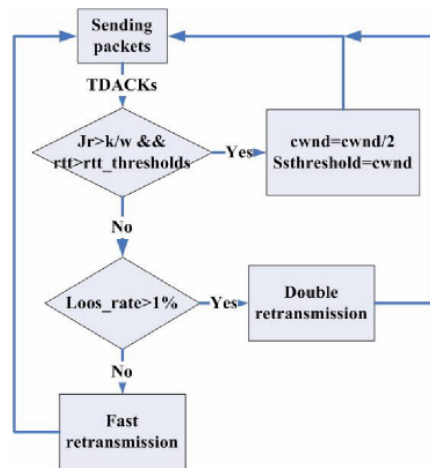


Fig. 3. TDACKs is longer than one RTT.

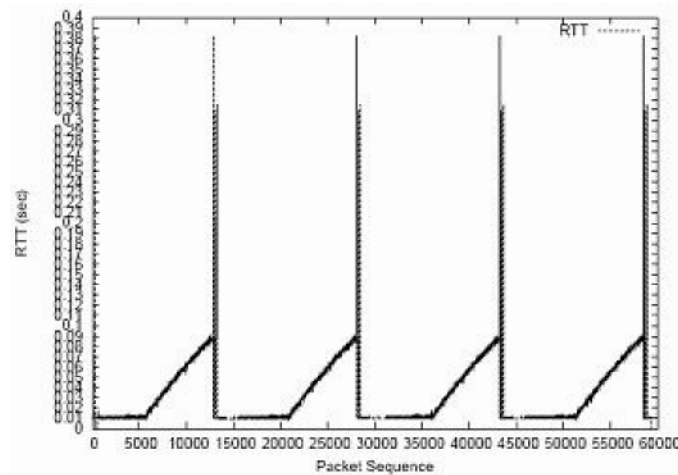


Fig. 4. RTT vs. packet sequence.

TCP traffic is under the fluid cross traffic model but the noise will not be evaluated under bursty cross traffic. However, RTTs only depend on the sum of propagation and queuing delay. Therefore, the RTTs can help us to further filter out the impact of noise. We initially investigate a simplified scenario to evaluate the relationship between RTTs and packet timeout, caused by congestion loss, in fluid cross traffic model. As the simulation result in Fig. 4, we set the threshold of RTTs to 0.2 to filter the noise in the following simulations.

In our methods,  $c$  plays a crucial role in determining the threshold. However, the value of  $\alpha$  is determined by the fluctuation of background traffic. The detailed investigation to analyze relationship between  $c$  and background traffic will be carried in our future work.

### 3.2 Burst Transmit and Smooth Transmit Scheme

According to experimental data, we found that even if TCP sender can judge the reasons that lead to cause packet loss, the throughput of TCP is still inefficient. By comparing the growth and decline of congestion window in wireless environment with 0% loss rate and 1% loss rate, shown in Figs. 5 and 6, we find that under the 1% loss condition, TCP sender reduces the congestion window at the wrong time continuously. By observing the curve of *cwnd*, TCP sender goes through slow start several times but the value of *cwnd* is still small.

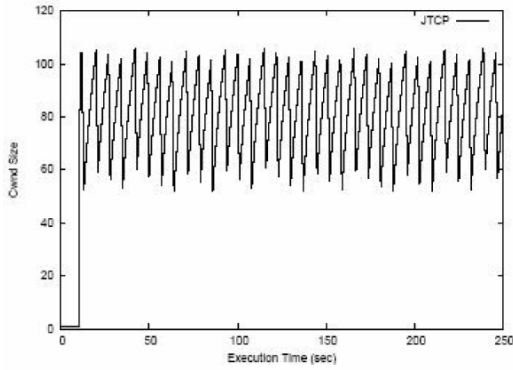


Fig. 5. *Cwnd* of JTCP with 0% loss rate.

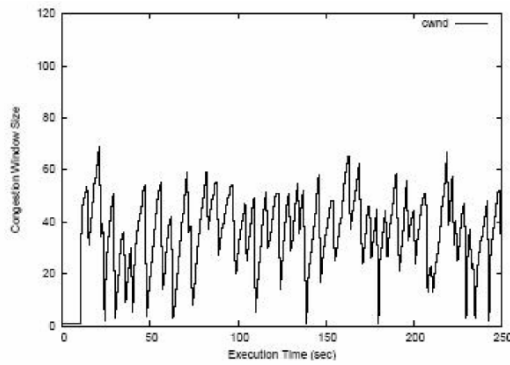


Fig. 6. *Cwnd* of JTCP with 1% loss rate.

As Fig. 6, upon tracing the TCP transmission events, we can find that TCP sender undergoes a great number of timeouts. The occasions of TCP timeout can be classified into two types roughly, retransmission loss and burst transmission. For the first case, while TDACKs are being sent back to the source to point out the lost packet, those TDACKs will trigger fast retransmission in the sender side. However, if the retransmitted packet gets lost again, the sender will wait the ACK until retransmission timeout.

For the second case, if TCP random losses occur too closely, it may cause TCP sender to transmit a large number of packets, those packets will fill up the queue of router and some of the packets may be dropped. For better understanding, we introduce the TCP transmission mechanism first. It follows the equation as:

$$\text{Max\_seq.} = \text{ACK\_seq.} + \text{dupwnd} + \text{cwnd.}$$

*Max\_seq.* denotes the max data sequence number that TCP can send out; *ACK\_seq.* is the received ACK sequence and *dupwnd* is the counter of duplicate ACK. *Data\_seq.* in table denotes the sequence of sent data. In order to mitigate the impact of double *fast\_retransmission* in EJTCP, we use the cumulative sequence number to filter the retransmission event. Therefore, the  $\text{Max\_seq.} = \text{ACK\_seq.} + \text{dupwnd} + \text{cwnd.}$

The case is summarized in Figs. 8 and 7. D denotes data; R represents retransmitted data and A represents ACK. There are three lost packets, 4758, 4807 and 4814. After sending out D-4817, the sender receives 56 duplicate ACKs to point out the lost packet,

ACK_seq	dupwnd	cwnd	Max_seq	sendout packet
4757	0	60	4817	4817
4757	56	60	4873	4873
4806	0	60	4866	X
4806	8	60	4874	4874
4806	53	60	4919	4919
4813	0	60	4873	X
4813	44	60	4917	X
4919	0	60	4979	4920-4979

Fig. 7. Time caused by burst transmission.

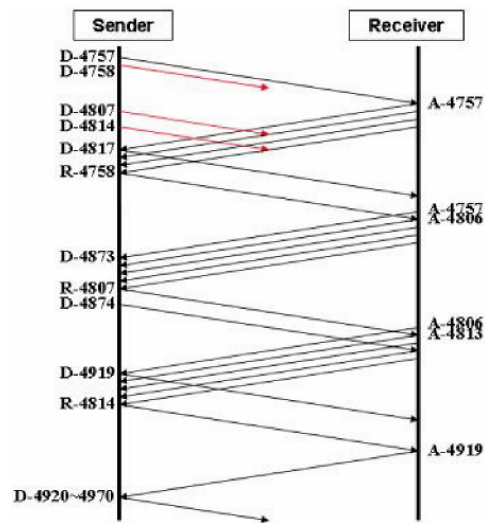


Fig. 8. Timeout caused by burst transmission.

D-4758. Upon receiving R-4758, the destination sends A-4806 to indicate the next lost packet. However, as the sender receives A-4806, *Max\_seq* is equal to 4866 which is smaller than the packet it has sent. The sender will stop sending until *dupwnd* = 8 and the same situation repeats later. Upon receiving A-4813, the sender stops sending again and waits for *dupwnd* steps up. However, in the waiting duration, TCP sender receives A-4919. At that moment, it finds that *Max\_seq* is equal to 4979 and last *Data\_seq* is 4919. In other words, the source can send out 60 packets, so it sends out 60 data packets at the same time. However, the burst data will overload the bottleneck router and generate a packet loss burst.

In order to address this timeout problem, we design two policies to manage two conditions described above respectively. For the retransmitted loss problem, when TDACK triggers loss retransmission, the sender judges the causes of loss first and calculates the packet loss rate (*loss\_rate*) based on its record. If it is caused by transmitted error, and *loss\_rate* > 1%, the source retransmits two duplicate packets at the same time, or the source retransmits lost packet once *loss\_rate* < 1%. Otherwise, it lowers the transmitted rate and resends the lost packet once because the network is congested. We can elaborate the conditions as follows:

```

If (jr > k/cwnd && rtt > rtt_thresholds)
    ssthreshold = (1/2) x cwnd;
    cwnd = ssthreshold;
Else
    If (loss_rate > 1%)
        double_retransmission();
    Else
        fast_retransmission();
    Endif
Endif

```

For the burst transmission problem, we use a variable,  $N$ , to control transmitted mechanism. When the number of transmitting packet is larger than  $N$ , the sender can transmit  $N$  packets. In other words, the mechanism is like slow start, but it can send out  $N$  packets at most rather than two at the same time. The value of  $N$  can not be too large to affect transmission but this mechanism must be faster than slow start. Therefore, we set  $N = 3$  in our simulation. This mechanism is like slow start but it does not wait to timeout and set  $cwnd = 0$ . Therefore, the performance will be better than the original scheme.

### 3.3 Physical Layer-Aware Scheme for 802.16

For IEEE 802.16 WiMAX network, whenever the wireless characteristics of environment (*i.e.* bit error rate, power degradation and interference power) change, the modulation parameters may be adjusted to the suitable one in the physical layer to increase efficiency. However, this adjust mechanism may cause congestion in the BS if the modulation parameters are adjusted from a higher transmitted rate to lower and there is no bottleneck in the wired router along the path. The packets will be queued in BS and the queue will be filled up if the sender is not aware to slow down the transmitted rate.

We can avoid this kind of congestion by cross-layer design between the physical layer and the transport layer. We implement this mechanism by embedding modulation information, modulation bits, into TCP header in the physical layer. The TCP sender can modulate its transmission mechanism according to the modulation bits. When the modulation changes in the process of transmission, because the transmitted rate of each modulation is proportioned, we can regulate the related parameters of TCP corresponding by the modulation bits.

## 4. SIMULATION EXPERIMENTS

We experiment several scenarios to validate that the proposed strategies for EJTCP do offer better performance than other wireless TCP solutions for various wireless events. Simulations are performed with network simulator 2 (NS-2) [22]. Fig. 9 shows the network topology that is a wired-cum-wireless mixed network in the first network scenario. S is the source node, R1 and R2 are routers, and D is the destination node. The transmitted rate between S and R1 is 100Mbps with 5ms propagation time. The bottleneck is between two routers, the transmitted rate is 5Mbps with 40ms propagation time. The last hop is wireless link between R2 and D. The transmitted rate of wireless link is 10Mbps for 802.11 network and 64Mbps for 802.16 network with 0.01ms propagation time. The packet size is 1,000 bytes and the size of queue is 100.

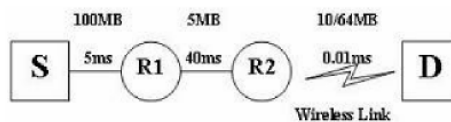


Fig. 9. Simulation network topology.

#### 4.1 Throughput

The first experiment compares the throughput of EJTCP with other TCP schemes (Reno, Newreno, Westwood, JTCP). The losses may be random or congestion drop over wireless link. We run the simulation with 5Mbps bottleneck capacity and the total time is 250 seconds. Fig. 10 shows the simulation result.

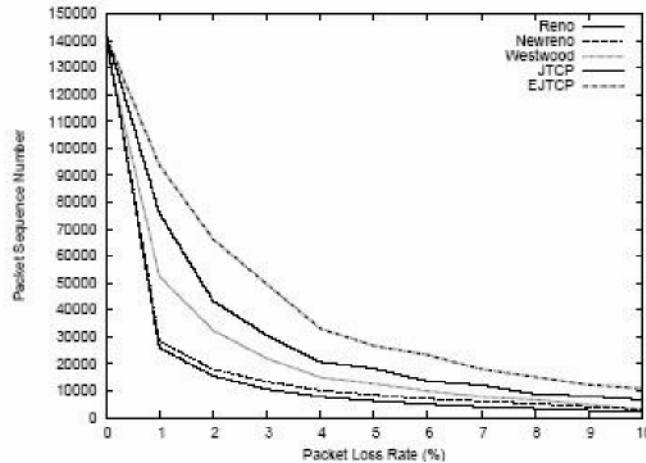


Fig. 10. Average throughput with 5MB bottleneck.

Initially, the throughputs of all versions of TCP are about the same for 0% loss rate. However, when the loss rate increases, the divergence is more and more conspicuous. Comparing with other TCP scheme, EJTCP has more than 20% improvement for the 1% to 5% packet loss rate and more than 50% improvement for the 6% to 10% packet loss rate. The major reason of the improvement is that EJTCP can determine the cause of loss more precisely and avoid retransmission loss and bursting transmission to overload the bottleneck router. If we change the bottleneck capacity as 2Mbps and perform the same simulation, we can get the experimental results as Fig. 11, which is alike with above.

#### 4.2 Fairness

Similar to the utilization scenario study, we change the connection number of TCP from 1 to 10 to calculate fairness with other connection. The definition of fairness is:

$$Fairness = \frac{(\sum b_i)^2}{n \times (\sum b_i^2)}$$

$n$  denotes the total connection number and  $i$  denotes the index of connection. From the fairness equation, we can see that the fairness with 1 corresponding to best allocation among all connections. We run this simulation with 1% loss. Fig. 12 shows the simulation result.

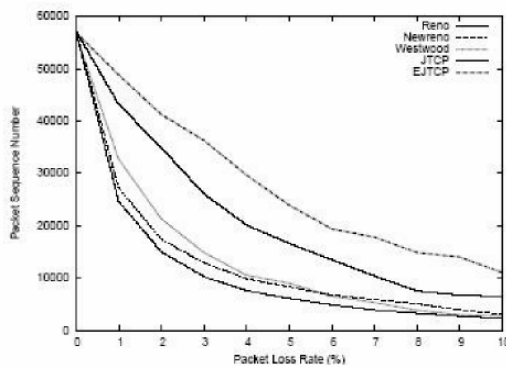


Fig. 11. Average throughput with 2MB bottleneck.

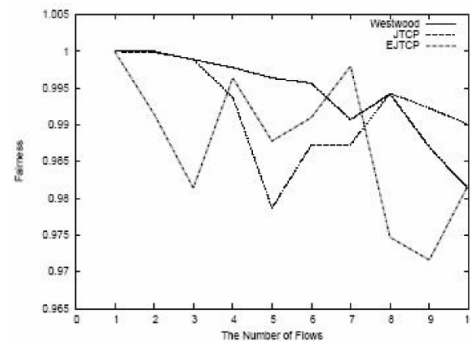


Fig. 12. Fairness with 1% loss rate.

EJTCP's fairness is better with Westwood's and JTCP's. The average value of fairness is about 0.985 of EJTCP. That means that EJTCP can achieve intra-friendly with other connections. Those connections can share the bottleneck bandwidth impartial. When the bursty error of wireless channel is high, related approach – westwood, vegas and reno will false-determine the congestion signal and further reduce their transmission rate wrongly. Under this circumstance, TCP can't utilize the residual bandwidth, and our method can use the residual bandwidth left by TCP. This is also our goal, and TCP friendliness cannot be hold.

#### 4.3 Single-hop and Multi-hop Topology

In this section, we use two more topologies to demonstrate EJTCP's performance under Poisson cross traffic. The first topology is single-hop environment, which is illustrated in Fig. 13. The bottleneck is located in the wireless link and its bandwidth is 11Mbps. Wireless TCP's flow was sent from S to D as well as the cross traffic was sent from Cs to Cr with CBR traffic. The link's bandwidth except the bottleneck link is 100Mbps.

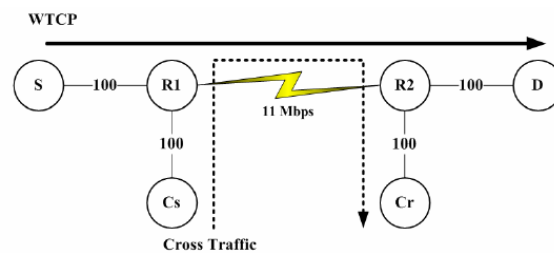


Fig. 13. Single-hop environment, bottleneck link is wireless link.

As we can see, the simulated result in Fig. 14 shows that both JTCP and EJTCP can achieve more than 50% utilization when the random loss rate is less than 7%. Compared to the Westwood, JTCP can provide more than 40% improvement and EJTCP's improvement in bandwidth utilization close to 50%.

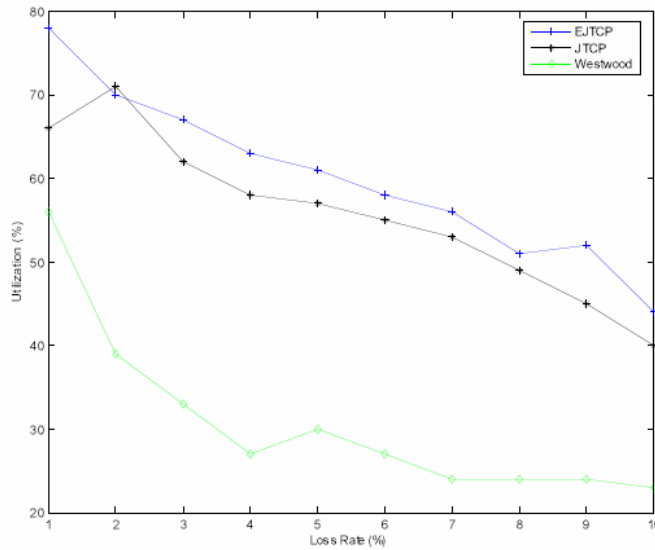


Fig. 14. Single-hop environment and poisson cross traffic.

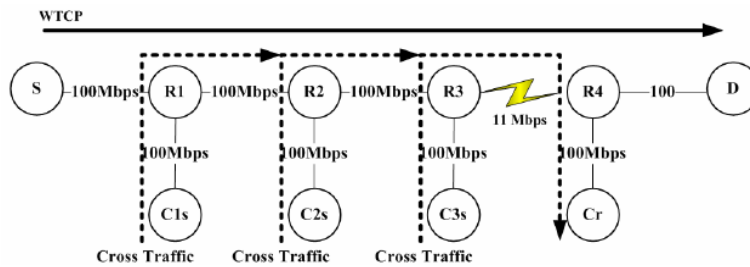


Fig. 15. multi-hop environment, bottleneck link is wireless link.

The second topology is single-hop environment, which is illustrated in Fig. 15. The bottleneck is located in the wireless link also and its bandwidth is 11Mbps. Wireless TCP’s flow was sent from S to D as well as three pairs of cross traffic were sent from C1s to Cr, C2s to Cr and C3s to Cr with CBR traffic. The link’s bandwidth except the bottleneck link is 100Mbps.

As we can see, the simulated result in Fig. 16 shows that both JTCP and EJTCP can achieve more than 25% utilization when the random loss rate is less than 5%. Compared to the Westwood, JTCP can provide more than 15% improvement and EJTCP’s improvement in bandwidth utilization close to 20%.

In Fig. 16, EJTCP performs better than JTCP in some instance but worse in others. This stranger behavior is caused by the cross traffic in backward path. In multi-hop scenario, the relationship between round-trip time and queuing delay in the bottleneck may be disrupted by aggregated background traffic in the backward path. Under this circumstance, delay jitter, which is depended on the one-way delay only, will perform better accuracy in loss determination.

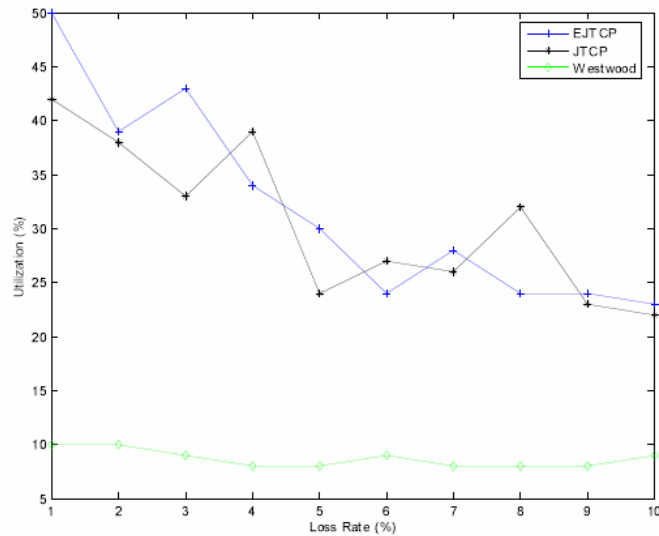


Fig. 16. Multi-hop environment and CBR cross traffic.

## 5. CONCLUSION

This article presented an enhanced protocol, EJTCP, to improve the performance of TCP over the wired-cumwireless network. This protocol maintains the end-to-end semantics of TCP and achieves fairness. Simulation results show that EJTCP has better throughput than other wireless TCP algorithms. We correct the lost judgment mechanism of JTCP to reduce error judgment for lost events. We add ACKed packet sequence into TCP header to rectify packet received time. Furthermore, since RTT can reflect the state of network, and we use this information to help us to distinct the cause of packet lost. We set a variable to calculate the *rtt\_threshold*. When losses occur, the sender receiver TDACKs or retransmit timeout, we compare the value of *jr* with  $1/w$  and compare the value of RTT with *rtt\_threshold* to distinguish the causes of losses. The simulation results prove that EJTCP can judge more accurately and get better performance in wireless link.

The article also point out a new problem that caused by burst transmission after judging the lost event correctly. This burst transmission event may produce overload to router and cause burst packet loss. We propose a smooth transmit scheme to solve this problem. Our scheme scatters the large number of data like the mechanism of slow start and avoids unnecessary timeout effectively. We propose the rate adjustment scheme of TCP to deal with the modulation changing in 802.16 networks. Each modulation parameter represents a specific transmitted rate, and the transmitted rate will change for applying different modulation parameters. When the transmitted rate is changed, we adjust the parameters of TCP correspondingly. This mechanism can avoid congestion at BS when the transmitted rate is reduced. Simulation results prove that EJTCP can judge more accurately and get better performance in wireless links.

## REFERENCES

1. R. Stewart and C. Metz, "SCTP: new transport protocol for TCP/IP," *IEEE Internet Computing*, Vol. 5, 2001, pp. 64-69.
2. V. Jacobson and M. J. Karels, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM*, 1988, pp. 314-329.
3. M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, 1999.
4. C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "IEEE standard 802.16: a technical overview of the wirelessMANTM air interface for broadband wireless access," *IEEE Communications Magazine*, Vol. 40, 2002, pp. 98-107.
5. R. Fish, "DOCSIS cable modem service overview," in *Proceedings of IEEE International Conference on Consumer Electronics*, 2000, pp. 76-77.
6. W. J. Liao and H. J. Ju, "Adaptive slot allocation in DOCSIS-based CATV networks," *IEEE Transactions on Multimedia*, Vol. 6, 2004, pp. 479-488.
7. G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, "TCP performance issues over wireless links," *IEEE Communications Magazine*, Vol. 39, 2001, pp. 52-59.
8. K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE Journal on Selected Areas in Communications*, Vol. 22, 2004, pp. 747-756.
9. S. Vangala and M. A. Labrador, "Performance of TCP over wireless networks with the snoop protocol," in *Proceedings of the 27th IEEE Local Computer Networks*, 2002, pp. 600-601.
10. H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, Vol. 5, 1997, pp. 756-769.
11. Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *IEEE Communications Magazine*, Vol. 43, 2005, pp. S27-S32.
12. C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP westwood: bandwidth estimation for enhanced transport over wireless links," in *Proceedings of ACM Mobicom*, 2001, pp. 287-297.
13. M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo, "TCP westwood: congestion window control using bandwidth estimation," in *Proceedings of IEEE Globecom*, Vol. 3, 2001, pp. 1698-1702.
14. N. Parvez and E. Hossain, "Improving TCP performance in wired-wireless networks by using a novel adaptive bandwidth estimation mechanism," in *Proceedings of IEEE Globecom*, Vol. 5, 2004, pp. 2760-2764.
15. A. Capone, L. Fratta, and F. Martignon, "Bandwidth estimation schemes for TCP over wireless networks," *IEEE Transactions on Mobile Computing*, Vol. 3, 2004, pp. 129-143.
16. L. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, Vol. 13, 1995, pp. 1465-1480.
17. K. Takagaki, H. Ohsaki, and M. Murata, "Analysis of a window-based flow control mechanism based on TCP Vegas in heterogeneous network environment," in *Proceedings of IEEE International Conference on Communications*, Vol. 10, 2001, pp. 3224-3228.
18. P. F. Cheng and C. L. Soung, "TCP Veno: TCP enhancement for transmission over

- wireless access networks,” *IEEE Journal on Selected Areas in Communications*, Vol. 21, 2003, pp. 216-228.
19. E. H. K. Wu and M. Z. Chen, “JTCP: jitter-based TCP for heterogeneous wireless networks,” *IEEE Journal on Selected Areas in Communications*, Vol. 22, 2004, pp. 757-766.
  20. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: a transport protocol for real-time application,” RFC 1889, 1996.
  21. S. Y. Chen, E. H. K. Wu, and M. Z. Chen, “A new approach using time-base model for TCP-friendly rate estimation,” in *Proceedings of IEEE International Conference on Communications*, Vol. 1, 2003, pp. 679-683.
  22. NS-2 network simulator, <http://www.isi.edu/nsnam/ns>.
  23. X. Liu, K. Ravindran, B. Liu, and D. Loguinov, “Single-hop asymptotics in available bandwidth estimation: sample-path analysis,” in *Proceeding of ACM SIGCOMM Internet Measurement Conference*, 2004, pp. 300-313.



**Eric Hsiao-Kuang Wu (吳曉光)** received his B.S. degree in Computer Science and Information Engineering from National Taiwan University in 1989. He received his Master and Ph.D. in Computer Science from the University of California, Los Angeles (UCLA) in 1993 and 1997. He is a full professor of computer science and information engineering at National Central University, Taiwan. His primary research interests include wireless networks, mobile computing, and broadband networks. He is a member of Institute of Information and Computing Machinery (IICM) and IEEE.



**Yu-Chen Huang (黃玉成)** was born in Taiwan, on January 4, 1974. He received his B.S. degree and Master in Computer Science and Information Engineering from National Chung Cheng University, Taiwan in 1996 and 1998. He is currently a Ph.D. student in Computer Science and Information Engineering of National Central University, Chungli, Taiwan. His research interests are in network measurement, networking protocols, and active network services.



**Gui-Kui Chang (張桂魁)** received his B.S. degree and Master in Computer Science and Information Engineering from National Central University, Taiwan in 2003 and 2005. His primary research interests include TCP/IP, congestion control, and reliable multicast for wireless networks.