TCP-Jersey for Wireless IP Communications

Kai Xu, Ye Tian, and Nirwan Ansari, *Senior Member, IEEE*

***Abstract*—Improving the performance of the transmission control protocol (TCP) in wireless Internet protocol (IP) communications has been an active research area. The performance degradation of TCP in wireless and wired-wireless hybrid networks is mainly due to its lack of the ability to differentiate the packet losses caused by network congestions from the losses caused by wireless link errors. In this paper, we propose a new TCP scheme, called TCP-Jersey, which is capable of distinguishing the wireless packet losses from the congestion packet losses, and reacting accordingly. TCP-Jersey consists of two key components, the available bandwidth estimation (ABE) algorithm and the congestion warning (CW) router configuration. ABE is a TCP sender side addition that continuously estimates the bandwidth available to the connection and guides the sender to adjust its transmission rate when the network becomes congested. CW is a configuration of network routers such that routers alert end stations by marking all packets when there is a sign of an incipient congestion. The marking of packets by the CW configured routers helps the sender of the TCP connection to effectively differentiate packet losses caused by network congestion from those caused by wireless link errors. This paper describes the design of TCP-Jersey, and presents results from experiments using the NS-2 network simulator. Results from simulations show that in a congestion free network with 1% of random wireless packet loss rate, TCP-Jersey achieves 17% and 85% improvements in goodput over TCP-Westwood and TCP-Reno, respectively; in a congested network where TCP flow competes with VoIP flows, with 1% of random wireless packet loss rate, TCP-Jersey achieves 9% and 76% improvements in goodput over TCP-Westwood and TCP-Reno, respectively. Our experiments of multiple TCP flows show that TCP-Jersey maintains the fair and friendly behavior with respect to other TCP flows.**

***Index Terms*—Bandwidth estimation, explicit congestion notification, loss differentiation, wireless transmission control protocol (TCP).**

I. Introduction

COMMUNICATION networks have evolved greatly in the last decade. Packet switching technologies have eventually merged the traditional voice networks and data networks together into a converged and integrated multimedia network. The horizon of the converged integrated network is extending further to incorporate wired, wireless, and satellite technologies. All-Internet protocol (IP) wired and wireless hybrid network is becoming a reality. Transmission control protocol (TCP) has become the dominant communication protocol suite in today's multimedia applications. Nowadays, about 90% of the Internet traffics are carried by TCP. TCP needs to depart from its original wired network oriented design and evolve to meet the challenges introduced by the wireless portion of the network. IP [1] is a connectionless, best-effort-based variable length packet delivery network layer protocol that does not guarantee the reliable, timely and in-order delivery of packets between end stations. TCP [2] is a layer 4 transport protocol that uses the basic IP services to provide applications with an end-to-end connection-oriented packet transport mechanism that ensures the reliable and ordered delivery of data.

TCP was originally designed primarily for the wired networks. In the wired networks, random bit-error rate (BER) is negligible and congestion is the main cause of packet loss. TCP and many of its variants implement flow control and congestion control algorithms [3] based on the sliding window and additive increase multiplicative decrease (AIMD) [4] algorithms. A sliding window-based flow control mechanism allows the sender to advance the transmission window upon the reception of an acknowledgment (ACK) that indicates the last in-order packet has been received successfully by the receiver. When packet loss occurs at a congested link due to buffer overflow at the intermediate router, either the sender receives duplicate ACKs (DUPACK) or the sender's retransmission timeout (RTO) timer expires. These events activate the sender's congestion control mechanism by which the sender reduces the size of its transmission window, or congestion window ($cwnd$) in TCP terms, resulting in a lower transmission rate to relieve the link congestion. The original TCP algorithm, TCP-Tahoe, has three transmission phases, namely slow-start (SS), congestion avoidance (CA), and fast retransmit. TCP-Reno [5] extends TCP-Tahoe to include a fast recovery phase in conjunction with the fast retransmit. TCP maintains two variables, the congestion window size ($cwnd$), which is initially set to be 1 maximum segment size (MSS), and SS threshold ($ssthersh$). At the beginning of the TCP connection, the sender enters the SS phase, in which it increases the $cwnd$ by 1 MSS for every ACK it receives. Therefore, in the SS phase, TCP sender's $cwnd$ grows exponentially. When the $cwnd$ reaches the $ssthresh$, the TCP sender enters the CA phase. During this phase, the sender increases the $cwnd$ by $1/cwnd$ for every ACK it receives, which is equivalent to an increment of $cwnd$ by 1 MSS for every round-trip time (RTT). This additive increase leads to the linear growth of the transmission rate that helps the sender to slowly probe the available network bandwidth. The congestion window is reduced by 1/2 of the current value when the sender has received n DUPACKs (n is usually 3). At this point, TCP infers that packets were lost due to link congestion; it sets the $ssthresh$ to be the same as $cwnd$ and starts retransmission of the lost packet. This marks the beginning of the fast retransmit phase. During the

Manuscript received February 7, 2003; revised September 18, 2003. This work was supported in part by the New Jersey Commission on Higher Education via the NJI-TOWER project and in part by the the New Jersey Commission on Science and Technology via NJWINS.

The authors are with the Advanced Networking Laboratory, Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, NJ 07102 USA (e-mail: Nirwan.Ansari@njit.edu).

Digital Object Identifier 10.1109/JSAC.2004.825989

fast retransmit phase, TCP-Reno invokes the fast recovery algorithm to speed up the recovery process, by which the sender treats the DUPACKs received during the fast retransmit phase as normal ACKs and artificially inflates the $cwnd$. This inflated portion of $cwnd$ is later deducted at the end of the fast retransmit phase. The fast retransmit phase ends when the sender receives a normal ACK that acknowledges that the receiver has successfully received an ordered packet whose sequence number passes beyond the previous lost packet. Therefore, TCP decreases its $cwnd$ multiplicatively in the presence of packet loss.

The above TCP sender behavior works fairly well in wired network where packet losses are almost always caused by link congestions, and packet losses due to bit errors are usually negligible or, if any, not exceeding one loss per transmission window. However, in the wired/wireless heterogeneous networks, high BER, fading, and blackout become nonnegligible factors to packet losses. TCP's reactive congestion control and avoidance mechanism based on the assumption that all packet losses are due to congestions become incapable of handling the mixed packet losses. TCP without modification exhibits throughput degradations when used in wired/wireless heterogeneous networks [6], [7].

In this paper, we propose a new TCP scheme, TCP-Jersey, for the mixed wire and wireless networks. The rest of the paper is organized as follows. Section II briefly reviews the related works in improving the TCP performance for wireless IP communications. Section III presents in detail the two components of our proposed TCP-Jersey scheme, as well as its overall operation. We then present in Section IV various simulation results under different network conditions. We conclude the paper in Section V with a summary of the results and highlights of the future works.

II. Related Works

In the all-IP heterogeneous networks, where wired network interconnects with other wireless networks, e.g., cellular, satellite, and Ad Hoc networks, congestion is no longer the only cause of packet loss. Transmission errors, such as, random bit error on the wireless link, signal fading, and mobile handoff process, also contribute a significant amount to packet loss. The challenges of wireless networks [8] include high BER, limited bandwidth, and unexpected disconnections.

A. Wireless TCP Modifications

Conventional TCP schemes may suffer from a severe degradation in performance in mixed wired and wireless environment [6], [7]. Extensive research works have been done to remedy this. The solutions can be categorized into the split mode approach, link layer approach, and end-to-end TCP modifications.

The split mode approach [9]–[11] tries to shield the wireless network from the wired network. TCP connections of the wired and wireless networks are separated. The intermediate node, 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. This approach requires large buffers at base stations and the end-to-end TCP semantics are sometimes not preserved.

The link layer approach [12]–[15] rectifies wireless link errors at the second layer. Techniques such as forward error correction (FEC), automatic repeat request (ARQ), and explicit loss notification (ELN) have been proposed for this class of solutions. However, such approach requires protocols at different layers to interact and coordinate closely, which increases the complexity of protocol implementation.

In the end-to-end approach, TCP senders and receivers are responsible for the flow control; hence, the end-to-end semantics are preserved. TCP-Peach [16] is particularly designed for the satellite communication environment, where large bandwidth-delay product is the nature. Amongst the four phases of TCP, TCP-Peach leaves CA and fast retransmit intact. It replaces SS and fast recovery with sudden start and rapid recovery, respectively. In sudden start and rapid recovery, the sender probes the available network bandwidth in only one RTT with the help of low-priority dummy packets. An important assumption made by TCP-Peach is that the routers must support priority queueing. In TCP-Peach Plus [17], the actual data packets with lower priority replace the low-priority dummy packets as the probing packets to further improve the throughput. TCP-Westwood [18] is a rate based end-to-end approach, in which the sender estimates the available network bandwidth dynamically by measuring and averaging the rate of returning ACKs. TCP-Westwood claims improved performance over TCP-Reno and -Sack, while achieving fairness and friendliness. The end-to-end approach maintains the network layer structure and requires minimum modification at end hosts and router.

B. RED and ECN

Routers can also take part in controlling TCPs transmission rates by means of active queue management (AQM). Random early detection (RED), proposed in [19], is an AQM scheme implemented in routers that informs the sender of incipient congestion by probabilistically dropping packets before the buffer overflows. Two thresholds, namely, $\min_{\rm th}$ and $\max_{\rm th}$, are maintained and the average queue length ($\rm avg$) is computed from the moving average of the instantaneous queue length at the router. A RED router drops packets according to a pre-configured probability if $\rm avg$ lies between the two thresholds. Packets are not dropped if $\rm avg$ is below $\min_{\rm th}$; all packets are dropped if $\rm avg$ is above $\max_{\rm th}$. The packet drop triggers the receiver to send DUPACKs to the sender. The sender is therefore able to adjust its window size in response to the packet drop by means of congestion control. The early packet drop prevents the router to enter the fully congested state. By doing so, the average queue length at the router can be kept small, hence reducing the queueing delay and improving the TCP throughput.

Explicit congestion notification (ECN) [20] is an extension to RED. Instead of randomly early dropping packets, an ECN router marks packets to alert the sender of incipient congestion. ECN is an explicit signaling mechanism designed to convey network congestion information from routers to end stations; however, since the signaling only uses one bit for such congestion information, the information conveyed is not quantitative. For TCP flow control, ECN works by configuring the intermediate router to mark packets with congestion experienced (CE) bit in the IP header when the router's average queue

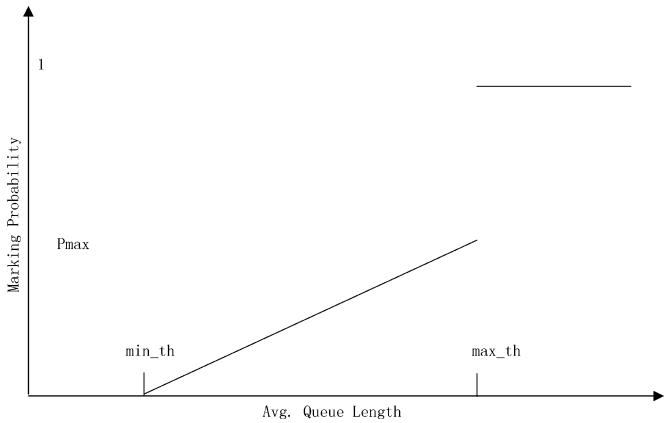


Fig. 1. Marking function of ECN.

occupancy exceeds a threshold, so that the TCP receiver can echo this information back to the sender via ACK by setting the explicit congestion echo (ECE) bit in the TCP header. The router's packet marking/dropping algorithm is described as follows. ECN is usually implemented by configuring the RED router to marking instead of dropping packets when the average queue length lies between the $\min_{\rm th}$ and $\max_{\rm th}$ thresholds. An ECN capable router is configured with three parameters, the minimum threshold $\min_{\rm th}$, the maximum threshold $\max_{\rm th}$, and the maximum marking probability $P_{\max}$. When a packet arrives at the router, if the average queue length is below $\min_{\rm th}$, the router will not mark the packet. If the average queue length exceeds the $\min_{\rm th}$ but below $\max_{\rm th}$, the router will mark the packet with CE bit in its IP header with probability $P = ((\text{avg} - \min_{\rm th})/(\max_{\rm th} - \min_{\rm th}))P_{\max}$. If the average queue length is above $\max_{\rm th}$, then the packet is marked with probability 1. Therefore, the marking probability at the router increases linearly as the average queue length builds up from the minimum threshold to the maximum threshold. The marking function is depicted in Fig. 1.

In the forwarding direction of the TCP flow, when the receiver receives a packet that has its CE bit set, the receiver sets the ECE bit in the TCP header for all subsequent packets it sends back to the sender until the sender signals the receiver of its action on the congestion notification. This is achieved by the sender setting the congestion window reduced (CWR) bit in the TCP header. In the reverse direction of the flow, upon reception of a packet with the ECE bit set, the TCP sender immediately cuts its sending rate to half and then invokes the CA algorithm to gradually increase the rate back to a sustainable level. The sender also sets the CWR bit to signal the receiver of its action.

III. TCP-JERSEY

In the heterogeneous networks, to explicitly differentiate the cause of packet loss becomes the leading goal for the TCP design. Both TCP-Peach and -Westwood, designed for wireless networks, do not support ECN at the moment. Since ECN implementation is becoming readily available at the routers and provides explicit network congestion information, it would be a plus to incorporate ECN in the wireless TCP design. In this section, we first describe two key components used by TCP-Jersey in Sections III-A and III-B, followed by the implementation in Section III-C.

A. Available Bandwidth Estimation

TCP-Tahoe, -Reno, and their variants detect the available bandwidth of the bottleneck link by continuously increasing the window size until the network is congested and then decrease the window size multiplicatively, e.g., the AIMD algorithm. However, the decrement of window size is rather heuristic and coarse, i.e., halving the current window size. This is because these TCP schemes lack the ability to quantitatively estimate the available bandwidth before the congestion happens. These TCP schemes' congestion control mechanisms are reactive rather than proactive and preventive. TCP-Westwood [18] proposes a remedy to this problem by employing its bandwidth estimator at the sender side based on the interval of the returning ACKs. In TCP-Westwood, the bandwidth estimator works as follows. When the sender receives an ACK at time t_k, it records a sample of the bandwidth as

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

where d_k is the amount of data acknowledged by the ACK and t_{k-1} is the time when the previous ACK was received. This sample bandwidth is further smoothed by a low-pass filter using Tustin approximation as

$$\hat{b_k} = \frac{\frac{2\tau}{t_k - t_{k-1}} - 1}{\frac{2\tau}{t_k - t_{k-1}} + 1}\hat{b}_{k-1} + \frac{b_k + b_{k-1}}{\frac{2\tau}{t_k - t_{k-1}} + 1} \tag{2}$$

where $\hat{b}_k$ is the filtered measure of the available bandwidth at time t_k, and $1/\tau$ is the cutoff frequency of the filter. In addition, TCP-Westwood also employs a timer in its bandwidth estimator such that if time τ/m ($m > 2$) has passed without receiving any new ACKs, the filter assumes the reception of a sample $b_k = 0$. When TCP-Westwood determines that the link is congested after receiving three DUPACKs, it sets the SS threshold to reflect its estimated bandwidth delay product as

$$ssthresh = \frac{\text{BWE} \times \text{RTT}_{\min}}{seg\_size} \tag{3}$$

where BWE is the estimated bandwidth, $\text{RTT}_{\min}$ is the minimum of TCPs estimation of round trip delay, and seg_size is the segment size. The congestion window, $cwnd$, is then set to be the same as $ssthresh$ if the connection is not in the SS phase, i.e., $cwnd > ssthresh$. However, [18] does not describe the practical rules of choosing parameters τ and m.

TCP-Jersey adopts the same idea of estimating the available bandwidth at the sender by observing the rate of the returning ACKs, but uses a rather simple estimator. The bandwidth estimator we propose in TCP-Jersey is derived from the time-sliding window (TSW) estimator proposed in [21]. In [21], the network router employs the following estimator to estimate the bandwidth occupied by individual flows

$$R_n = \frac{T_w \times R_{n-1} + L_n}{(t_n - t_{n-1}) + T_w} \tag{4}$$

where R_n is the estimated bandwidth when packet n arrives at time t_n , t_{n-1} is the previous packet arrival time, L_n is the size of packet n, and T_w is a constant time window.

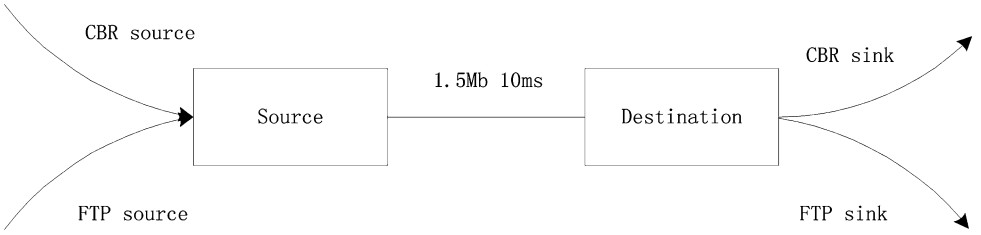


Fig. 2. Simulation setup for comparative study of the effectiveness of bandwidth estimators.

TCP-Jersey employs the available bandwidth estimator (ABE) at the sender. ABE monitors the rate of receiving ACKs to estimate the available bandwidth for the TCP connection and then calculates the optimum congestion window size based on this estimation. TCP-Jersey computes the optimum congestion window once every RTT. When window reduction is needed because of the congestion signaling from CW, TCP-Jersey sets the $cwnd$ and $ssthresh$ to the estimated optimum congestion window size. The ABE estimator is as follows:

$$R_n = \frac{\mathrm{RTT} \times R_{n-1} + L_n}{(t_n - t_{n-1}) + \mathrm{RTT}} \tag{5}$$

where R_n is the estimated bandwidth when the nth ACK arrives at time t_n, t_{n-1} is the previous ACK arrival time, L_n is the size of data that the nth ACK acknowledges, and RTT is the TCP's estimation of the end-to-end RTT delay at time t_n. The optimum congestion window ($ownd$) in units of segment is then calculated as

$$ownd_n = \frac{\mathrm{RTT} \times R_n}{seg\_size} \tag{6}$$

where seg_size is the segment size. It is worth noting that, first, our bandwidth estimator is extremely simple to implement and is not dependent on configuration parameters. Secondly, like the original TSW estimator, it decays with time, making it suitable for networks with nonstatic bandwidth delay product, a characteristic usually exhibited in the mixed wired and wireless networks.

To verify the effectiveness of our proposed ABE and provide a comparative result with the TCP-Westwood's bandwidth estimator, we conducted the following simulation using the NS-2 network simulator [22]. The network topology for the comparative study depicted in Fig. 2 consists of two network nodes, the source node and the destination node, and the bottleneck link between them. The bottleneck link is configured as a 1.5 Mb/s error free duplex link with one-way link delay of 10 ms. The bottleneck queue is a drop-tail queue with a large enough buffer space to ensure that there is no buffer overflow throughout the simulation. FTP traffic with a constant packet size of 1000 B is simulated between the source and destination nodes. Between the source and destination nodes, we also setup user datagram protocol (UDP)-based CBR background traffic with a packet size of 1000 B.

The rate of the CBR traffic varies throughout the simulation as follows. From time 0 to 5 s, the CBR source generates traffic at the rate of 500 kb/s and then stops for 5 s; from 10 to 15 s, the CBR generates traffic at the rate of 750 kb/s; from 15 to 20 s, the rate of the CBR is changed back to 500 kb/s, and then stops for another 5 s; the CBR traffic resumes from time 25 s at the rate of 500 kb/s till the end of the simulation time. The total simulation time is 30 s. We run the simulation twice; first we use the TCP-Westwood, whose NS-2 module is available from [23], to carry the FTP traffic; in the second run, we use TCP-Jersey for the FTP and we record the estimated bandwidths from TCP-Westwood and TCP-Jersey, respectively, as the simulation proceeds. Fig. 3 plots the bandwidth estimations from two TCPs versus the actual available bandwidth for the FTP traffic. From Fig. 3, we can see that both bandwidth estimators from TCP-Westwood and TCP-Jersey follow the changes of the available bandwidth fairly closely, but TCP-Jersey's ABE provides a more accurate estimation than TCP-Westwood.

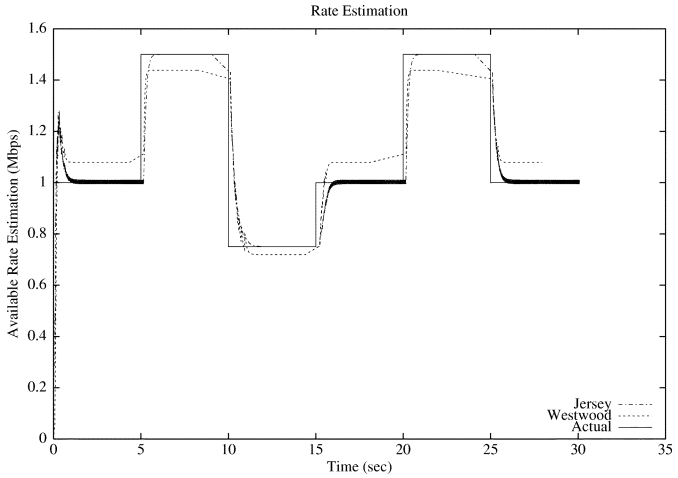


Fig. 3. Comparison of bandwidth estimators.

B. Congestion Warning

The current ECN scheme marks packets probabilistically, while the average queue length lies between $\min_{\mathrm{th}}$ and $\max_{\mathrm{th}}$. The router thereby not only informs the sender of the congestion, but also influences on which TCP connection the congestion window size would be adjusted due to its randomness in packet marking.

Echoing back ECN information from the receiver to the sender takes time, whereas network situation is constantly changing. Although ECN provides valuable congestion information, this information may not be timely enough for the sender to make the right decision suitable for the current network status under all circumstances. Moreover, both RED and ECN are sensitive to parameter settings [24]. Improper parameter settings may lead to unsatisfactory TCP performance [25]. We, therefore, propose a simpler congestion notification scheme, namely congestion warning (CW), with fewer parameter settings, yet still provides essential and accurate congestion information to the sender. We propose that the router shall mark all the packets when the average queue length exceeds a threshold (thresh) and leave the TCP sender who receives marks

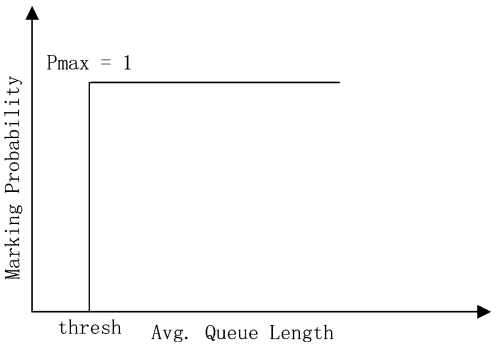


Fig. 4. Packet marking probability of the CW scheme.

to decide its window adjustment strategy. The router's marking scheme is depicted in Fig. 4. CW inherits the same information bits used in the original ECN implementation, i.e., the CE bit in the IP header and the ECE and CWR bits in the TCP header to convey the congestion warning information. We assume that routers along the connection path implement the original ECN mechanism and its parameters can be configured to reflect the CW scheme without the need to change the router's software. We believe this is a fairly realistic assumption because RED and ECN have gained their popularity since they were first introduced in early 1990s.

The calculation of the average queue length in CW also differs from that of the original ECN. In the original ECN, the average queue length is largely dependent on a long-term averaging value of the instantaneous queue length [19]. However, in our case, a larger queue weight is preferred since we expect the average queue length to closely track the instantaneous queue length and at the same time smooth out small spikes in the instantaneous queue length. A close tracking of the instantaneous queue length provides the sender with more accurate buffer information of the router. The original queue weight suggested in [19] is rather small, e.g., 0.002. Experiments show that a queue weight of 0.2 would be good enough to track and smooth the instantaneous queue length. The effect of the queue weight on the average queue length is illustrated in Fig. 5.

Fig. 5(a) shows that the average queue length does not follow the instantaneous queue length when the queue weight is set to 0.002. Fig. 5(b) shows that the average queue length closely follows the instantaneous queue length as expected for queue weight 0.2. Note that such settings make sense in our design. Since in our scheme, we do not rely on the router's AQM for TCP congestion control, rather, we only require the router to alert, in a timely manner, about the incipient congestion and let the TCP sender to carry out the control mechanism.

The nonprobabilistic packet marking of CW routers helps the TCP sender to differentiate the cause of packet losses. When the TCP sender receives DUPACK with the CW mark, it knows for sure that the network is in the congested state and assumes that the packet loss indicated by DUPACK is more likely to be caused by congestion. On the other hand, if a DUPACK without the CW mark is received, the sender could assume with higher confidence that the loss has occurred due to transmission errors. In the wired-wireless heterogeneous network, this packet loss is more likely to be from the wireless link. Suppose that the packet loss is caused by a random bit error in the wireless

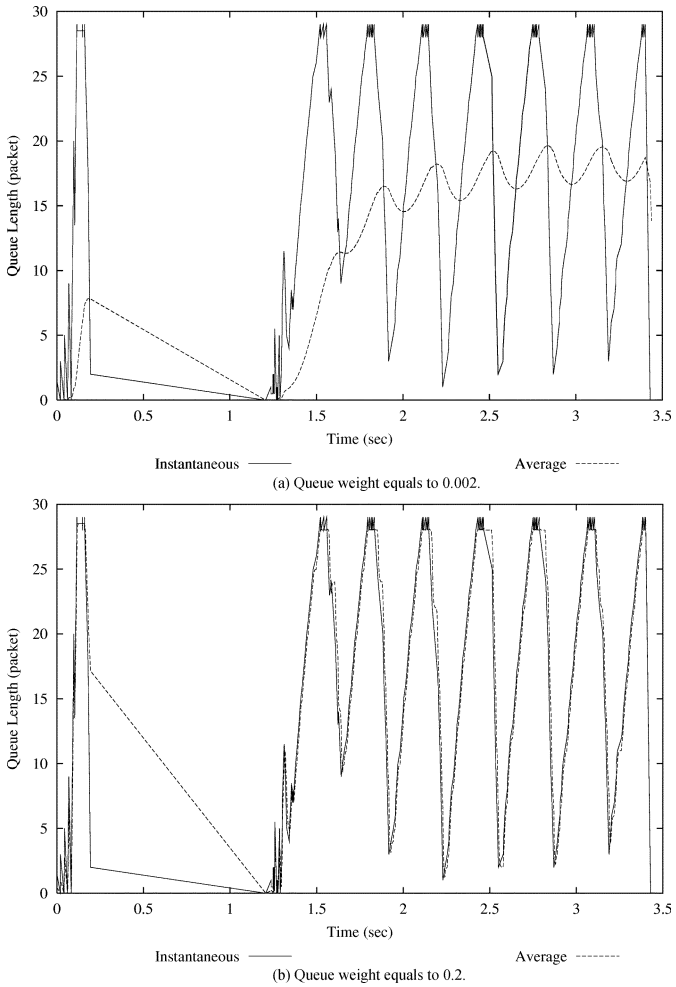


Fig. 5. Average queue length versus instantaneous queue length with different queue weights.

channel,[1] the wireless error recovery procedure at the TCP sender would not lower its current window size, but retransmit the lost packets using the current window size since there is no sign of congestion, and the DUPACK without mark actually indicates that the network has the capability to deliver the packets with the current window size. We will discuss the procedure in more detail in Section III-C. Accurately differentiating wireless packet losses from congestion packet losses is one of the key issues in improving TCP performance for IP communications in the wireless environment. Erroneously interpreting a packet loss due to the wireless error as congestion does not hurt too much. However, misinterpreting a packet loss triggered by congestion is more severe because the wireless error recovery will react as if there were no congestion. Suppose a packet loss is interpreted as a link error, but was actually caused by congestion, retransmission without reducing window size only aggravates the network congestion level. Therefore, we would rather set the threshold at the router aggressively to

[1]A proper wireless recovery function is needed for different types of wireless errors. TCP-Jersey focuses on illustrating its capability of distinguishing wireless losses from congestion losses. The random link errors are considered for this algorithm. Further differentiation of various types of wireless losses such as those caused by fading and handoff will be investigated in our future improvement of TCP-Jersey.

avoid missing any packet loss caused by buffer overflow. Experiments show that setting the threshold at 1/3 of the link buffer capacity is satisfactory. Almost 100% of congestion losses and most link losses are differentiated successfully.

C. Operation of TCP-Jersey

The rationale of proposing a joint approach with the combination of CW and ABE is described as follows. Intermediate routers generate ECN markings when there is a sign of congestion judged by comparing the average queue length with the thresholds. In the heterogeneous network with the presence of wireless links, the bandwidth-delay-products of the links no longer stay static during a session. The congestion indications created by CW may not always be as reliable as we would hope. Blindly dropping the sender's congestion window by a constant upon receiving marked ACKs is certainly not an optimum solution for better TCP performance. For instance, in an all-IP network, many traditional non-IP communications are carried upon the IP network, most noticeably the voice-over-IP (VoIP) services . Because of the ON-OFF nature of the voice conversation, by the time a TCP flow sharing bandwidth with VoIP flows receives congestion notification via CW, the VoIP stream may happen to be in the OFF state, thus making the notification invalid. We disagree with traditional TCP's blindly halving its sending rate upon receiving indications of link congestion. There is a proposal of congestion response in TCP that implements the idea of reducing the sending rate in smaller steps, i.e., 1/8 or 1/16 of the current congestion window [26]. However, these fixed window decrease factors are still somewhat heuristic and do not adapt to the level of the link congestion. We believe that the 1 bit of information used in today's congestion notification mechanism is not enough for TCP to make intelligent adjustment to deal with link congestion. We either need to extend the current congestion notification scheme so that routers can notify TCP end stations of the link congestion in a quantitative way, or use another mechanism that can interact with the binary congestion signals. In this paper, we propose a joint approach, namely TCP-Jersey that combines ABE and CW, which is a minimally modified version of ECN at the router, so that the TCP sender could set its congestion window to a more sensible value when congestion is detected. Also, as we have stated before, such a combination improves TCP's ability to differentiate random wireless packet losses from losses caused by congestion. We will discuss the details of our proposed algorithm in the next section.

The proposed TCP scheme, TCP-Jersey, incorporates CW and ABE introduced in Section II. The pseudocode of the ABE algorithm is presented in Fig. 6. The parameter k (line 3) is a scaling factor that controls the weight of the past estimation. It is usually set to 1. The available bandwidth estimation is computed once every RTT. Procedure ABE is invoked at the sender upon receiving an ACK or DUPACK.

TCP-Jersey adopts SS, CA, and fast recovery from Reno but replaces Reno's fast retransmit with explicit retransmit and introduces the rate control procedure. The only difference between Reno's fast retransmit procedure and Jersey's explicit retransmit procedure is that unlike Reno's retransmit procedure that halves the current congestion window before starting the retransmission, explicit retransmit keeps the current $cwnd$. It

```
Initialization:
  ABE = 0
  T_prev = 0
  T_last = 0

(1)  ABE( )
(2)  {
(3)    TW = RTT * k
(4)    Delta = now - T_prev
(5)    T_prev = now
(6)    if (now - T_last) >= RTT
(7)      ABE = (TW * ABE + L) / (Delta + TW)
(8)      T_last = now
(9)    end if
(10)   return ABE
(11) }
```

Fig. 6. Pseudocode of the ABE algorithm.

```
(1)  recv( )
(2)  {
(3)    ABE()

(4)    if ACK and CW=0
(5)      SS() or CA()
(6)    end if

(7)    if ACK and CW=1
(8)      rate_control()
(9)      SS() or CA()
(10)   end if

(11)   if nDUPACK and CW=1
(12)     rate_control()
(13)     explicit_retransmit()
(14)     fast_recovery( )
(15)   end if

(16)   if nDUPACK and CW=0
(17)     explicit_retransmit()
(18)     fast_recovery( )
(19)   end if
(20) }
```

Fig. 7. Pseudocode of TCP-Jersey's sender receiving procedure.

leaves the adjustment of the congestion window to the rate control procedure. The operation of the rate control procedure is also quite simple. The procedure sets the $ssthresh$ to $ownd$, the optimum congestion window size computed using (6), and sets the $cwnd$ to be the same as $ssthresh$ if the connection is in the CA phase. The pseudocode of TCP-Jersey's sender receiving module is depicted in Fig. 7. It operates as follows. Upon entry, it invokes the ABE procedure (line 3). If an ACK is received without the CW mark, it proceeds as Reno, i.e., invoking SS or CA depending on whether or not the $cwnd$ is below the $ssthresh$ (line 4–6). If the received ACK or the nth DUPACK is marked with the CW bit, it calls the rate control procedure to adjust the window size and proceeds with SS or CA if it is an ACK (line 7–10); or enters the explicit retransmit if it is the nth DUPACK (line 11–15). When the nth DUPACK is received without the CW mark, TCP-Jersey renders the packet drop is caused by a random error, and therefore it enters the explicit retransmit without adjusting the window size (line 16–19).

The flowchart of TCP-Jersey sender's response to DUPACK is illustrated in Fig. 8, and the sender's response to the normal ACK is illustrated in Fig. 9.

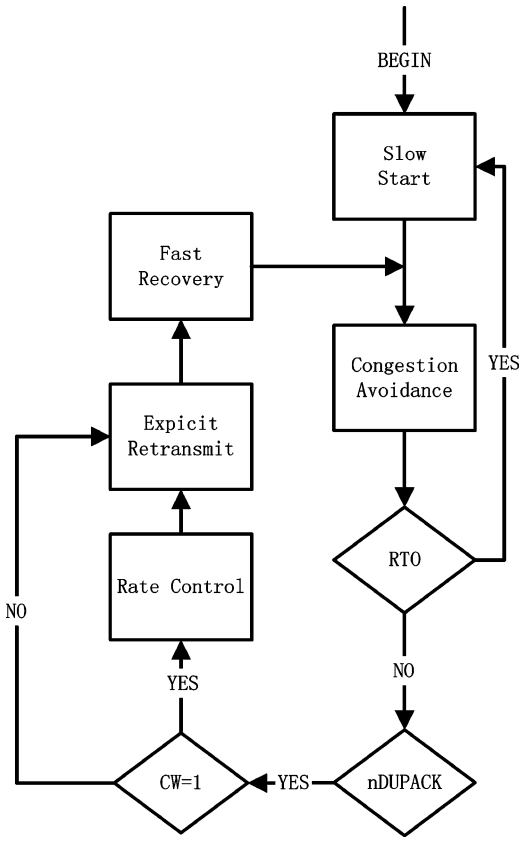


Fig. 8. Flowchart of TCP-Jersey sender's response to DUPACK.

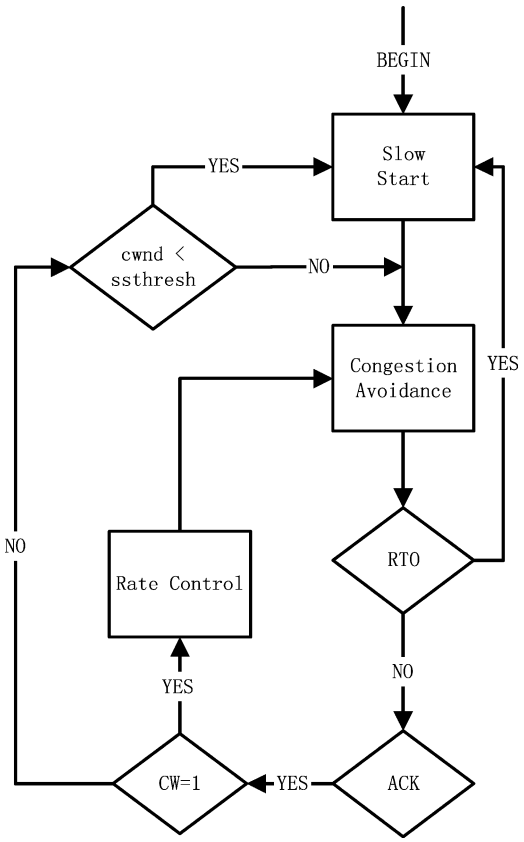


Fig. 9. Flowchart of TCP-Jersey sender's response to ACK.

IV. Simulation Results

We simulate TCP-Jersey in order to show its goodput performance, friendliness, and fairness in mixed wired and wireless networks, with or without the presence of link loss and congestion. The simulation tool we use is the NS-2 network simulator [22]. We have done the necessary code modification in NS-2 to undertake the experiments.

A. Goodput Performance

Goodput is the effective amount of data delivered through the network. It is a direct indicator of network performance. We expect that a good TCP scheme transmits as many data as possible, while behaving friendly to other TCP flows in terms of consuming the network resource, e.g., bandwidth. The simulation environment is depicted in Fig. 10. The source connects to a wireless base station via a 10 Mb/s error free link with 45 ms one-way delay. The base station is linked to the destination, a wireless mobile node, via a 2 Mb/s lossy channel with 1 ms delay. A single TCP connection running a long-live FTP application delivers data from the source to the destination.

We run the simulation for TCP-Tahoe, -Reno, -Westwood, and -Jersey, respectively. Under such network configuration, there is almost no congestion. The random link error rate[2] at the wireless bottleneck link varies from 0.001% to 10%. The simulation time is 100 s. The goodput is calculated based on the received ACKs at the sender side. The goodput result is shown in Fig. 11. The error rate is plotted in the logarithmic scale.

[2]The link-error rate used in our simulation is the packet loss rate instead of the BER.

Fig. 10. Goodput simulation environment.

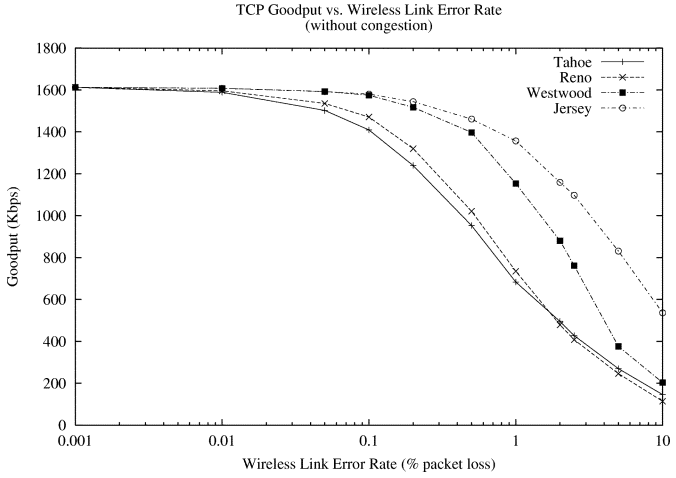


Fig. 11. Comparison of goodput with TCP-Tahoe, -Reno, -Westwood, and -Jersey without presence of congestion.

For link-error rate smaller than 0.01%, all TCP schemes perform closely to each other. Beyond that point, TCP-Jersey starts to outperform the other TCP variants. The closest competitor is TCP-Westwood. The goodput of Westwood does not fall behind Jersey very much until the error rate increases to 0.1%. At a very practical error rate for wireless loss, i.e., 1%, Jersey outperforms Westwood by 17% and Reno by 85%. Especially, at the high error rate side, Jersey still has satisfactory goodput,

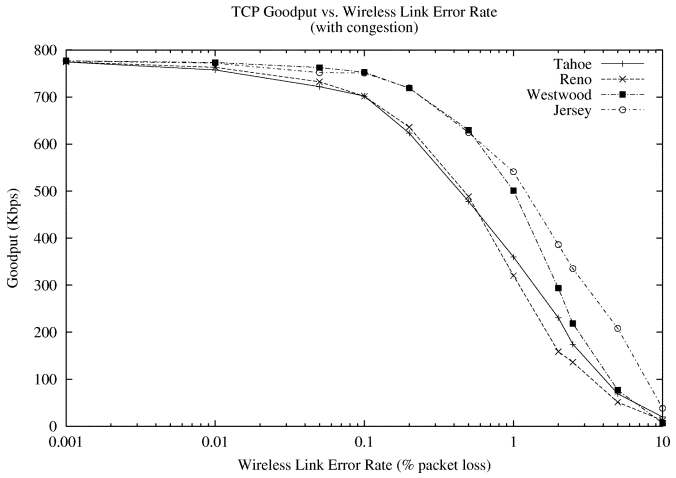


Fig. 12. Comparison of goodput of TCP-Tahoe, -Reno, -Westwood, and -Jersey with presence of congestion.

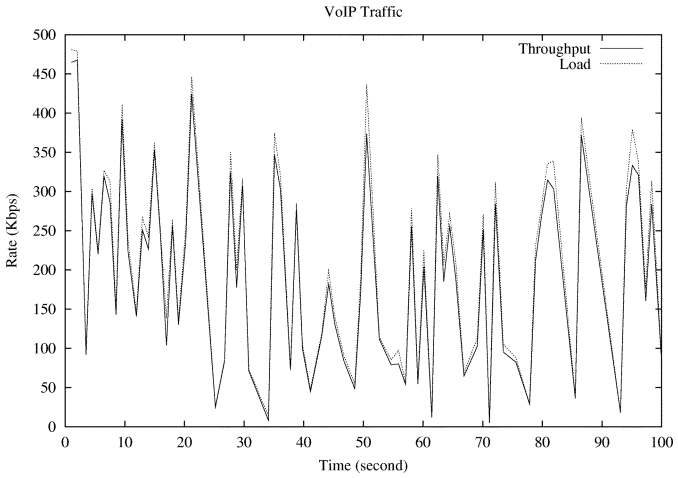


Fig. 13. Network load of VoIP traffics under TCP-Jersey.

whereas other TCPs experience a severe degradation in performance. This phenomenon is anticipated since Jersey handles wireless losses better than the other TCP variants because it can distinguish wireless losses from congestion losses and act accordingly.

We next introduce VoIP traffics to the same network in Fig. 10 to create congestions. The VoIP traffics consist of five VoIP flows carried by the UDP protocol. Each VoIP flow is modeled as an independent exponentially distributed on-off source that has 50% talk-spurt and 50% silent period. The average call duration of each VoIP calls is 8 s of simulation time. Voices are carried by UDP at 96 kb/s rate. New VoIP call arrivals to the pool of 5 VoIP sources form a Poisson process with the average inter-arrival time of 1 second. We run the simulation for TCP-Tahoe, -Reno, -Westwood, and -Jersey, respectively, and plotted the goodputs in Fig. 12.

Each TCP scheme suffers from performance degradation when the network is under congestion, compared with the goodput in Fig. 11, because the VoIP flows carried by UDP do not have any congestion control mechanism and never backoff. In other words, they are not friendly to TCP traffics. It can be seen that TCP-Jersey outperforms other TCP schemes at all error rate levels. Particularly, at error rate of 1%, TCP-Jersey improves the goodput over TCP-Westwood and TCP-Reno by 9% and 76%, respectively. Fig. 13 shows the overall VoIP flows' load and their throughput when the link error is set to be 1% of packet loss under TCP-Jersey. It is observed that VoIP traffics pass through at a rate close to their actual loads, implying that at the presence of network congestion, TCP-Jersey does not over aggressively consume the available network bandwidth.

B. Fairness of TCP-Jersey

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 function (7), proposed in [27], to justify the fairness of TCP schemes. The fairness index function is expressed as

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)} \tag{7}$$

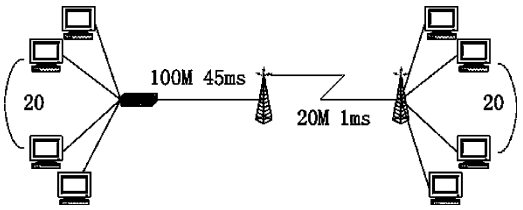


Fig. 14. Simulation network for obtaining fairness index.

TABLE I
FAIRNESS COMPARISON

Error Rate	Reno	Westwood	Jersey
0.0	0.98	0.99	1.0
0.1	0.99	0.97	1.0
0.5	1.0	0.99	0.99
1.0	0.99	1.0	0.99
5.0	0.99	0.99	0.97
10.0	0.97	0.97	0.97

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 0.0% to 10%.

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, (7) would yeild $1/n$.

We setup the simulation as shown in Fig. 14, where a total of 20 same TCP flows share a 20 Mb bottleneck link. We run the simulation for different TCP schemes and compare their fairness index; the results are summarized in Table I. All TCP variants including Jersey achieve fairly satisfactory fairness index.

C. Friendliness of TCP-Jersey

A friendly TCP scheme should be able to coexist with other TCP variants and not cause them starvation. To verify the friendliness of TCP-Jersey, we construct a mixed wired and wireless network, where TCP-Jersey coexists with Reno. The simulation network is shown in Fig. 15. The wired link has 100 Mb/s bandwidth and 45 ms delay, and the wireless link has

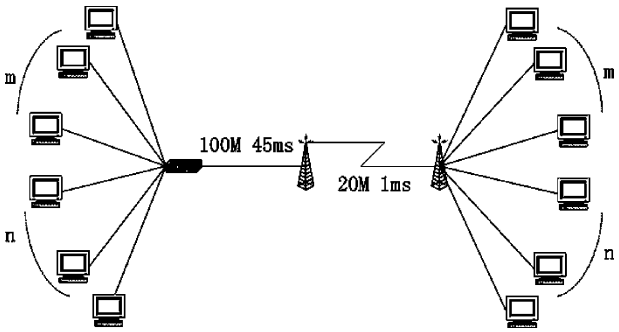


Fig. 15. Simulation network for verifying friendliness.

TABLE II
THROUGHPUT COMPARISON OVER GOOD LINK

Reno Source	Jersey Source	Reno Mean Throughput	Jersey Mean Throughput
3	17	961.50	958.79
5	15	959.94	958.95
10	10	959.43	958.97
15	5	959.21	959.16
17	3	959.16	959.40

The number of Reno and Jersey sources varies. The total number of all sources is 20. Mean throughput is in units of Kbps. Bottleneck link has no random error.

TABLE III
THROUGHPUT COMPARISON OVER LOSSY LINK

Reno Source	Jersey Source	Reno Mean Throughput	Jersey Mean Throughput
3	17	895.23	965.76
5	15	903.53	972.33
10	10	916.20	993.86
15	5	937.57	1007.40
17	3	944.35	1015.16

The number of Reno and Jersey sources varies. The total number of all sources is 20. Mean throughput is in units of Kbps. Bottleneck link has 0.1% random error rate.

20 Mb/s and 1 ms delay. There are 20 pairs of connections, of which m are Jersey connections and n are 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 Mb/s per connection.

We first set the link error rate to 0% at the bottleneck link and record the throughput of each connection at the bottleneck link. The mean throughput of TCP-Reno and -Jersey is calculated by summing up the throughput of the same TCP scheme and divided by the number of connections, respectively. The results are listed in Table II. It is observed that the bandwidth allocation of each TCP connection is close to its fair share at the bottleneck link.

We next set the link-error rate to 0.1%. The throughput results are listed in Table III. TCP-Jersey achieves a slightly higher throughput than 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.

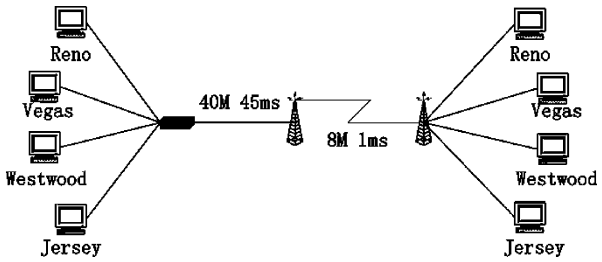


Fig. 16. Simulation network for coexistence of TCP schemes.

TABLE IV
FRIENDLINESS COMPARISON WITH FOUR CO-EXISTING TCP SCHEMES

Error Rate	Reno Throughput	Vegas Throughput	Westwood Throughput	Jersey Throughput
0.0	1664.22	1659.36	1663.98	1663.59
0.1	1158.94	1541.92	1354.50	1623.64
0.5	761.28	936.60	1338.91	1455.36
1.0	552.01	554.78	1077.56	1298.09
5.0	213.76	177.20	419.49	446.40

Four TCP connections are of TCP Reno, Vegas, Westwood and Jersey. Error rate is in units of packet drop percentage. Throughput is in units of Kbps.

We also run the simulation with four different TCP schemes coexisting and competing for the bottleneck link, i.e., coexistence of TCP-Reno, -Vegas, -Westwood, and -Jersey, as shown in Fig. 16. The random error rate at the wireless bottleneck link ranges from 1% to 5%. The bottleneck link has 8-Mb/s bandwidth.

The throughput results are listed in Table IV. For the error rate as low as 0.1%, four TCP schemes perform closely. As the error rate increases, TCP-Jersey achieves significantly higher throughput than TCP-Reno, -Vegas, and -Westwood, conforming to the results obtained from the goodput simulation in Section IV-A. It is also observed that the friendliness of TCP-Jersey is within the same range of Westwood. Both Westwood and Jersey behave more aggressively than other nonwireless oriented TCP schemes when the random wireless link error rate is beyond 0.1%. However, this behavior is anticipated since TCP-Jersey is designed to perform better in lossy wireless environment.

V. CONCLUSION AND FUTURE RESEARCH

In this paper, we have proposed a new TCP scheme, called TCP-Jersey, to improve the TCP performance in the heterogeneous network consisting of wired and wireless links. TCP-Jersey adopts the rate-based congestion control methodology, and employs the ABE using a modified TSW algorithm at the sender side to optimize the congestion window size when network congestion is detected. TCP-Jersey detects the congestion by means of the CW mechanism implemented at the router that marks all packets when the router's buffer occupancy exceeds a threshold. The congestion warning mechanism also aids the TCP sender to effectively differentiate packet losses due to random wireless link errors from those

caused by link congestion, so that the sender could react to DUPACK intelligently. Our approach is easy to implement. It only requires simple configurations at the ECN-enabled router and minimum changes to the TCP sender side code without altering the protocol itself.

Our simulations show that TCP-Jersey is a viable solution to the TCP performance degradation in wireless IP communications. Results from the simulations show that under 1% packet loss rate, a typical characteristic of wireless links, TCP-Jersey outperforms other TCP variants by 17% to 85% improvement in goodput when the network is not congested, and 9% to 76% goodput improvements when the network is congested by VoIP flows that share the same link. Our experiments also show that TCP-Jersey maintains the fair and friendly behavior to other TCP flows.

TCP-Jersey is able to distinguish congestion losses from wireless losses. Our future research in this direction would be to improve TCP-Jersey 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.

REFERENCES

[1] J. Postel, "Internet Protocol," IETF, RFC 791, 1981.

[2] ——, "Transmission Control Protocol," IETF, RFC 793, 1981.

[3] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Aug. 1988, pp. 314–329.

[4] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *J. Comput. Networks*, vol. 17, no. 1, pp. 1–14, June 1989.

[5] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms," IETF, RFC 2001, 1997.

[6] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, pp. 336–350, June 1997.

[7] F. Lefevre and G. Vivier, "Understanding TCP's behavior over wireless links," in *Proc. Communications Vehicular Technology*, 2000, SCVT-2000, pp. 123–130.

[8] V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *J. Wireless Commun. Mobile Computi.*, vol. 2, no. 1, pp. 3–20, Feb. 2002.

[9] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. ICDCS 95*, May 1995, pp. 136–143.

[10] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Comput. Commun. Rev.*, vol. 27, no. 5, pp. 19–43, 1997.

[11] K. Wang and S. K. Tripathi, "Mobile-end transport protocol: An alternative to TCP/IP over wireless links," in *IEEE INFOCOM*, vol. 3, Mar. 1998, pp. 1046–1053.

[12] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM/Baltzer Wireless Networks J.*, vol. 1, no. 4, pp. 469–481, Dec. 1995.

[13] S. Keshav and S. Morgan, "SMART retransmission: Performance with overload and random losses," in *Proc. IEEE INFOCOM'97*, vol. 3, 1997, pp. 1131–1138.

[14] K. Ratnam and I. Matta, "WTCP: An efficient mechanism for improving TCP performance over wireless links," in *Proc. Int Symp. Computers Communications*, 1998, pp. 74–78.

[15] H. Balakrishnan and R. H. Katz, "Explicit loss notification and wireless web performance," in *Proc. IEEE GLOBECOM Internet Mini-Conf.*, Sydney, Australia, Nov. 1998.

[16] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A new congestion control scheme for satellite IP networks," *IEEE/ACM Trans. Networking*, vol. 9, pp. 307–321, June 2001.

[17] I. F. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for satellite IP networks," *IEEE Commun. Lett.*, vol. 6, pp. 303–305, July 2002.

[18] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," *ACM Mobicom*, pp. 287–297, July 2001.

[19] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.

[20] S. Floyd, "TCP and explicit congestion notification," *ACM Comput. Commun. Rev.*, vol. 24, pp. 10–23, Oct. 1994.

[21] D. D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, pp. 362–373, Aug. 1998.

[22] UCB/LBNL/VINT Network Simulator [Online]. Available: http://www.isi.edu/nsnam/ns/

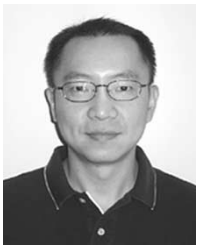
[23] TCP Westwood Modules [Online]. Available: http://www1.tlc.polito.it/casetti/tcp-westwood

[24] T. Bonald, M. May, and J. C. Bolot, "Analytic evaluation of RED performance," in *INFOCOM 2000*, vol. 3, Mar. 2000, pp. 1415–1424.

[25] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proc. Int. Workshop Quality-of-Service (IWQoS)*, 1999, pp. 260–262.

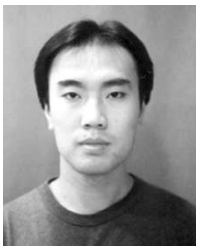
[26] A. Misra and T. Ott, "Jointly coordinating ECN and TCP for rapid adaptation to varying bandwidth," in *IEEE MILCOM 2001*, vol. 1, 2001, pp. 719–725.

[27] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC, Res. Rep.TR-301, 1984.



Kai Xu received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 1989 and the M.S. degree in electrical and computer engineering from Chiba University, Chiba, Japan, in 1993. He is working toward the Ph.D. degree in electrical and computer engineering at New Jersey Institute of Technology (NJIT), Newark.

His current research interests include wireless TCP, congestion control, and sensor networks.



Ye Tian received the B.E. (honors) degree in electrical and electronic engineering from University of Canterbury, Christchurch, New Zealand, in 1999, and the M.S.E.E. degree in electrical and computer engineering from New Jersey Institute of Technology (NJIT), Newark, in 2002, respectively. He is working toward the Ph.D. degree in electrical and computer engineering at NJIT.

His current research interests include TCP in heterogeneous networks, QoS routing, WLAN, and IPSec.

Nirwan Ansari (S'78–M'83–SM'94) received the B.S.E.E. degree (*summa cum laude*) from New Jersey Institute of Technology (NJIT), Newark, in 1982, the M.S.E.E. degree from the University of Michigan, Ann Arbor, in 1983, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

He joined the Department of Electrical and Computer Engineering, NJIT, as an Assistant Professor in 1988, and has been a Full Professor since 1997. He authored *Computational Intelligence for Optimization* with E. S. H. Hou (Norwell, MA: Kluwer, 1997) (and translated into Chinese in 2000), and edited *Neural Networks in Telecommunications* with B. Yuhas (Norwell, MA: Kluwer, 1994). His current research focuses on various aspects of high speed networks and multimedia communications.

Dr. Ansari is a Technical Editor of the *IEEE Communications Magazine*, as well as the *Journal of Computing and Information Technology*. He organized (as the General Chair) the First IEEE International Conference on Information Technology: Research and Education (ITRE2003), was instrumental, while serving as its Chapter Chair, in rejuvenating the North Jersey Chapter of the IEEE Communications Society which received the 1996 Chapter of the Year Award and a 2003 Chapter Achievement Award, served as the Chair of the IEEE North Jersey Section and in the IEEE Region 1 Board of Governors during 2001–2002, and currently serves in various IEEE committees. He was the 1998 recipient of the NJIT Excellence Teaching Award in Graduate Instruction, and a 1999 IEEE Region 1 Award. He is a keynote speaker for the IEEE/ACM co-sponsored ICETE2004 (International Conference on E-Business and Telecommunication Networks).