TFRC Veno: An Enhancement of TCP Friendly Rate Control over Wired/Wireless Networks

Bin Zhou
School of Computer Eng.
Nanyang Technological University
Singapore
Email: zhoubin@pmail.ntu.edu.sg

Cheng Peng Fu
School of Computer Eng.
Nanyang Technological University
Singapore
Email: ascpfu@ntu.edu.sg

Victor O. K. Li
Dept. of Electrical and Electronic Eng.
The University of Hong Kong
Hong Kong, China
Email: vli@eee.hku.hk

***Abstract*— TFRC is a TCP-Friendly Rate Control protocol based on TCP Reno's throughput equation. It is designed to provide optimal service for unicast multimedia flow operating in the wired Internet environment. However, in wireless networks, TFRC, same as TCP Reno, suffers significant performance degradation. In this paper, we propose to make use of a more advanced equation to enhance TFRC over wireless networks. This new equation is directly derived from the modeling of the wireless TCP rather than the wired TCP. After incorporating this equation into TFRC, two achievements are obtained: 1) this enhanced TFRC has a significant throughput improvement; it is shown that in wireless networks with 10% loss rate, it can obtain 300% improvement over the original TFRC; 2) this enhanced TFRC inherits the desirable features of TFRC, namely good fairness, nice TCP-friendliness and smoothness of sending rate. The extensive experiments, including simulation and live Internet measurements, validate our proposed scheme. Moreover, our scheme only needs to modify the sender-side protocol of TFRC while the receiver-side or intermediate node protocol stack remains intact.**

I. Introduction

Wireless communication technology will be playing an increasingly important role in access networks, as evidenced by the widespread adoption of wireless local area network (WLAN), wireless home networks, and cellular networks. These wireless access networks are usually interconnected using wired backbone networks, and many applications on the networks run on the TCP/IP protocol.

TCP is a reliable connection-oriented protocol that implements flow control by means of a sliding window algorithm. TCP Reno [1], which makes use of the SS (slow start) and CA (congestion avoidance) algorithms to adjust the window size, has enjoyed much success to date. While TCP congestion control is appropriate for applications such as bulk data transfer, real-time applications, i.e., online TV and VoIP, cannot be supported because of TCP's abrupt window reduction. In order to provide optimal service for unicast multimedia flow operating in the best-effort Internet environment, TCP-Friendly Rate Control (TFRC) [2], employing a throughput equation, was standardized in RFC 3448. This throughput equation, derived from TCP Reno congestion control algorithm [3] (hereinafter, called the "Reno equation"), dictates the performance of TFRC.

It is known that wireless networks may suffer high packet loss rate. TCP Reno always treats the occurrence of packet loss as a manifestation of network congestion. This assumption does not apply to networks with wireless channels, in which packet loss is often induced by noise, link error, or reasons other than network congestion. As in [4], we refer to such packet loss as random packet loss. Misinterpretation of random loss as an indication of network congestion causes Reno to reduce the sending data rate unnecessarily, resulting in significant performance degradation [4] [5] [6]. Consequently, TFRC, equipped with the Reno equation, will also unavoidably suffer severe performance degradation in wireless networks.

In this paper, we explore the enhancement of TFRC over wireless networks, using a more advanced equation to replace the Reno equation. Firstly, a more advanced throughput equation is derived through the modeling of TCP Veno, a wireless TCP congestion control algorithm [7]. The simulation and live test results verify the effectiveness of this new equation (hereinafter, called the "Veno equation") under different lossy situations.

Secondly, we apply this new equation to enhance TFRC performance. The new protocol is thus referred to as TFRC Veno. Our extensive experiments, including simulations and live Internet measurements, demonstrated that 1) TFRC Veno can achieve significant throughput improvement up to 300% over TFRC in wireless networks with 10% loss rate, and 2) the desirable features of fairness, TCP-friendliness and smoothness in this enhanced TFRC are same as in the original TFRC; that is, our improved throughput does not bring about any negative effect. It should also be noted that TFRC Veno only involves modification of TFRC at the sender side without requiring any changes of the receiver protocol stack or intervention of the intermediate network nodes. It can therefore be deployed immediately in server applications over the current Internet.

The remainder of the paper is organized as follows. Section II describes the related work, including several previous enhancements of TFRC. Then in Section III we talk about TFRC Veno with the Veno equation introduced. We comprehensively evaluate the overall performance of TFRC Veno in Section IV, with different metrics such as throughput, fairness, TCP-friendliness, and sending rate variation. We also compare our proposal with another sender-modified enhancement MULT-

1-4244-1588-8/07/$25.00 ©2007 IEEE

FRC [12]. Finally, Section V concludes this paper.

II. Related Work

Over the last years, some researchers have studied the throughput degradation problem of TFRC over wireless networks, and tried to eliminate such degradation with various approaches. Roughly speaking, these efforts can be divided into two categories: network-supported enhancements and end-to-end enhancements.

Network-supported enhancements of TFRC need the support of the intermediate nodes in the network, such as routers, proxies, access points, or other kinds of devices. The typical examples are ECN-based TFRC [8], Proxy-based TFRC [9], WM-TFRC [10], and AED-based TFRC [11].

ECN-based TFRC [8] uses intermediate routers to detect incipient network congestion and informs the receiver using Explicit Congestion Notification (ECN) marking. The receiver then measures the marking event rate p and feeds this information back to the sender. In this way, ECN-based TFRC ignores the losses occurring over the wireless hop or the effect of packet reordering encountered in the Internet, and only accounts for congestion losses in sending rate adjustments.

Proxy-based TFRC [9] employs a proxy to split the client-server connection, and uses TFRC only over the shared part of the client-server connection, i.e., the part between server and proxy. In the other part between client and proxy, it tunnels RTP via a TCP which is TCP-friendly by definition. This method improves the average throughput and wireless link utilization.

WM-TFRC [10] uses the access point (AP) in wireless LAN to measure the rate of wireless loss events (i.e., loss events caused by bit errors) p_w and feeds this value back to the sender periodically. Meanwhile, the receiver also provides feedback about the rate of total loss events (including wireless loss and congestion loss) p to the sender. Therefore, the sender can deduce the rate of congestion loss events (i.e., loss events caused by congestion) p_{con} by subtracting p_w from p. In this way, the sending rate of WM-TFRC, which is deduced from p_{con} instead of p, can be improved.

AED-based TFRC [11] uses a new differentiation scheme called Accurate and Explicit Differentiation (AED) to improve the performance of TFRC. AED assumes that agents are deployed before and after each wireless link. The agents inspect each packet and detect a loss by finding a packet with an out-of-order sequence number. Agents at the edge of wired networks treat a packet loss as a congestion loss and agents after a wireless link treat a packet loss as a wireless loss. The agents then mark the packets that are not lost with information about the lost packets (i.e., whether those packets were lost due to congestion or random errors). Thus, when receiving these marked packets, the receiver can calculate the correct loss event rate and provide feedback to the sender.

As described above, network-supported enhancements need to add new functions to the intermediate nodes in the network. This will cause the deployability problem in the real networks because it is difficult to change or replace the existing devices in the Internet. In contrast, end-to-end enhancements can avoid these deployment difficulties since their modifications on TFRC only involve the end nodes (sender or receiver or both). MULTFRC [12] belongs to this category, and it only modifies TFRC protocol on the sender side.

MULTFRC [12] borrows its idea from MULTCP [13], which was originally used to improve TCP performance in high bandwidth-round-trip-time product networks. The authors of MULTFRC find that using one TFRC connection in streaming applications results in under-utilization of the wireless bandwidth. They then propose the use of multiple simultaneous TFRC connections for a given wireless streaming application. The basic idea behind MULTFRC is to measure the round-trip time and adjust the number of connections accordingly. Specifically, MULTFRC increases the number of connections n by α/n or decrease it by β depending on the rtt measurements, that is,

$$n = \begin{cases} n - \beta, & ave\_rtt - rtt\_min > \gamma rtt\_min \\ n + \alpha/n, & otherwise \end{cases} \quad (1)$$

where α, β and γ are preset constant parameters, and the default values are 1, 1, and 0.2 respectively; ave_rtt is the average round trip time; for a given route, rtt_min is the minimum round trip time for that route, i.e. with no queuing delay. The authors also provide an enhancement of MULTFRC, called AIO-TFRC [14] to overcome the complexity issue of the MULTFRC implementation and undesirable results of quantization effect in MULTFRC. However, in spite of the high throughput achieved, both MULTFRC and its variant suffer from poor fairness and poor TCP-friendliness, as with MULTCP [13].

III. TFRC Veno

TFRC Veno is also a kind of end-to-end enhancement which only modifies TFRC on the sender side. Different from all the previous efforts that keep the Reno equation in TFRC intact, TFRC Veno in this paper tries to use a more advanced equation to replace the Reno equation. In fact, this equation is derived from wireless TCP, namely, TCP Veno.

A. Veno Equation

TCP Veno was proposed in 2003 [7] and has been incorporated in the Linux Kernel since version 2.6.18 [15]. The key idea of TCP Veno is to use state differentiation – congestion state or non-congestion state – to circumvent the difficult judgment of the packet loss type – congestion loss or random loss. Specifically, the number of packets on the connection N is estimated as follows:

$$N = \frac{cwnd}{RTT} \times (RTT - BaseRTT) \quad (2)$$

where $cwnd$ is the current congestion window size; $BaseRTT$ is the minimum of the measured round-trip time collected so far, and is reset whenever packet loss is detected either due to time-out or duplicated ACKs; RTT is the smoothed measured round-trip time.

If N is smaller than a certain threshold β when a packet loss occurs, the packet loss is deduced as random loss, and Veno decreases its congestion window ($cwnd$) with factor θ_1; otherwise, the packet loss is regarded as congestion loss and Veno thus decreases its congestion window with factor θ_2. Equation (3) describes this operation. Normally β is set to be 3, θ_1 is set to be $\frac{4}{5}$, and θ_2 is set to be $\frac{1}{2}$. A detailed Veno algorithm is included in [7].

$$cwnd = \begin{cases} cwnd \times \theta_1, & N < \beta \\ cwnd \times \theta_2, & N \geq \beta \end{cases} \tag{3}$$

As a matter of fact, TCP Veno acts quite similarly as Reno, and the only difference between TCP Veno and TCP Reno is how much the $cwnd$ value should be reduced when losses occur: Reno always decreases its $cwnd$ with factor $\frac{1}{2}$ while Veno sometimes decreases it with factor θ_1 (random loss), and sometimes with factor θ_2 (congestion loss). In this case, we introduce a random variable λ to represent a dynamic window drop factor, rather than a fixed factor of $\frac{1}{2}$ as in [3]. That is, λ can be θ_1 or θ_2 with certain probability. In fact, λ reflects how Veno performs window multiplicative reduction in response to packet loss.

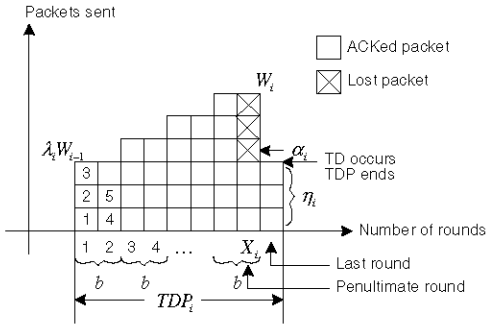


Fig. 1. Packets sent during a TD period.

Following [3], we consider a TD period (TDP) in Veno, which is a period between two TD losses (losses indicated by three-duplicated ACKs). As shown in Figure 1, the y-axis represents the number of packets sent during the current TDP, and the x-axis represents the number of rounds in the current TDP. Here a "round" starts with the back-to-back transmission of W packets, where W is the current size of the TCP congestion window. Once all packets falling within the congestion window have been sent in this back-to-back manner, no other packets are sent until the first ACK is received for one of these W packets. This ACK reception marks the end of the current round and the beginning of the next round. In Figure 1 the number of rounds in a TDP is represented by X_i. b is the number of packets that are acknowledged by a received ACK. Many TCP receiver implementations send one cumulative ACK for two consecutive packets received (i.e., delayed ACK), so b is typically 2. If W packets are sent in the current round and are all received and acknowledged correctly, then W/b ACKs will be received. Since each ACK increases the congestion window size by $1/W$, in the current round the windows size is incremented by $W/b \times 1/W = 1/b$. W_{i-1} and W_i are the congestion window sizes at the end of the previous TDP and the current TDP, respectively. According to the Veno algorithm, the current TDP starts with the congestion window size of $\lambda_i W_{i-1}$, where λ_i is either θ_1 or θ_2. After that, the window size is incremented by $1/b$ at each round until the end of the TDP.

If Y_i is the number of packets sent in the period, and A_i is the duration of the period, then the long-term steady-state Veno throughput can be shown to be:

$$B = \frac{E[Y]}{E[A]} \tag{4}$$

It is observed in [3] that:

$$E[Y] = \frac{1-p}{p} + E[W] \tag{5}$$

and,

$$E[A] = RTT(E[X]+1) \tag{6}$$

where p is the probability that a packet is lost, given that either it is the first packet in its round or the preceding packet in its round is not lost. Note that p includes both congestion and random losses. RTT is the average value of round trip time. Detailed derivations can be found in [3]. According to Figure 1 we have:

$$W_i = \lambda_i W_{i-1} + \frac{X_i}{b} \tag{7}$$

Then the packets transmitted between these two TD losses are:

$$\begin{aligned} Y_i &= \sum_{k=0}^{X_i/b-1} (\lambda_i W_{i-1} + k)b + \eta_i \\ &= \frac{X_i}{2}(\lambda_i W_{i-1} + W_{i-1} - 1) + \eta_i \end{aligned} \tag{8}$$

where η_i is the number of packets sent in the last round, as seen in Figure 1. η_i is considered a uniformly distributed random variable between 1 and W_i in [3]. We inherit this distribution in the Veno derivation.

From (7), (8) and (5), we have:

$$(1 - E[\lambda])E[W] = \frac{E[X]}{b} \tag{9}$$

and,

$$\frac{1-p}{p} + E[W] = \frac{E[X]}{2}((1 + E[\lambda])E[W] - 1) + E[\eta] \tag{10}$$

where $E[\eta] = \frac{E[W]}{2}$.

Let $\gamma = E[\lambda]$, and from (9) and (10) we can get:

$$E[W] = \frac{b(1-\gamma)+1}{2b(1-\gamma^2)} + \sqrt{\frac{2(1-p)}{b(1-\gamma^2)p} + \frac{\left(\frac{b(1-\gamma)+1}{2}\right)^2}{b^2(1-\gamma^2)^2}} \tag{11}$$

Then from (9) and (6), we have:

$$E[X] = \frac{b(1-\gamma)+1}{2(1+\gamma)} + \sqrt{\frac{2b(1-\gamma)(1-p)}{(1+\gamma)p} + \frac{\left(\frac{b(1-\gamma)+1}{2}\right)^2}{(1+\gamma)^2}} \tag{12}$$

and,

$$E[A]=RTT\left(\frac{b(1-\gamma)+1}{2(1+\gamma)}+\sqrt{\frac{2b(1-\gamma)(1-p)}{(1+\gamma)p}+\frac{\left(\frac{b(1-\gamma)+1}{2}\right)^2}{(1+\gamma)^2}}+1\right) \quad (13)$$

Finally, substituting $E[A]$ and $E[W]$ into (4) and (5) we have:

$$B(p)=\frac{\frac{1-p}{p}+\frac{b(1-\gamma)+1}{2b(1-\gamma^2)}+\sqrt{\frac{2(1-p)}{b(1-\gamma^2)p}+\frac{(\frac{b(1-\gamma)+1}{2})^2}{b^2(1-\gamma^2)^2}}}{RTT(\frac{b(1-\gamma)+1}{2(1+\gamma)}+\sqrt{\frac{2b(1-\gamma)(1-p)}{(1+\gamma)p}+\frac{(\frac{b(1-\gamma)+1}{2})^2}{(1+\gamma)^2}}+1)} \quad (14)$$

For small values of p, the above can be approximated by:

$$B(p)\approx\frac{1}{RTT}\sqrt{\frac{(1+\gamma)}{2b(1-\gamma)p}} \quad (15)$$

This is the throughput formula without accounting for TO (time-out) and advertisement window limitation. Following the steps in [3] we can also derive the complete formula. Due to the space constraints, we only give the final approximated result. Detailed derivations can be found in [16].

$$B(p)\approx\min\left(\frac{W_{\max}}{RTT},\frac{1}{RTT\sqrt{\frac{2b(1-\gamma)p}{1+\gamma}}+T_0\min\left(1,3\sqrt{\frac{b(1-\gamma^2)p}{2}}\right)p(1+32p^2)}\right) \quad (16)$$

Now we need to determine the value of γ. From our definition:

$$\begin{aligned}\gamma &= E[\lambda]\\ &= \theta_1\times P(N<\beta)+\theta_2\times P(N\geq\beta)\\ &= \theta_1\times P(N<\beta)+\theta_2\times(1-P(N<\beta))\\ &= \theta_2+(\theta_1-\theta_2)\times P(N<\beta)\end{aligned} \quad (17)$$

According to (2):

$$N=\frac{W}{RTT}\times(RTT-BaseRTT) \quad (18)$$

Here we use W to indicate the current congestion window size. For simplicity, we assume the value of W at losses (including congestion and random losses) is uniformly distributed between 0 and $W_{\max}$, where $W_{\max}$ is the maximum congestion window size during the transmission. Then we have:

$$\begin{aligned}P(N<\beta) &= P(W<\beta\times\frac{RTT}{RTT-BaseRTT})\\ &= min\left(1,\ \frac{\beta\times RTT}{(RTT-BaseRTT)\times W_{\max}}\right)\end{aligned} \quad (19)$$

From (17) and (19) we have:

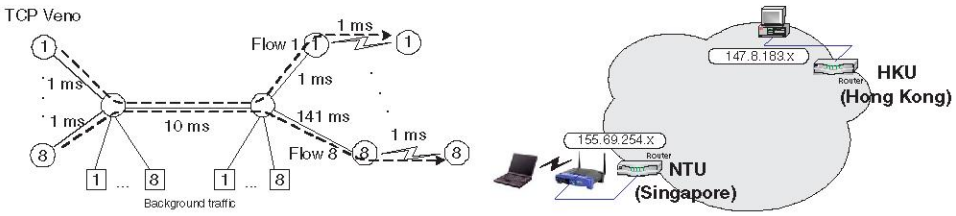
$$\gamma=min\left(\theta_1,\ \theta_2+(\theta_1-\theta_2)\times\frac{\beta\times RTT}{(RTT-BaseRTT)\times W_{\max}}\right) \quad (20)$$

Equation (16) with (20) is derived from TCP Veno, which is also referred to as the Veno equation. Different from the equation of TCP Reno [3], the Veno equation uses a variable $\gamma\in[\theta_2,\theta_1]$ to replace the original constant $\frac{1}{2}$. Here we call it the "Veno parameter". The following validations on NS-2 [17] simulation and live Internet experiments show that this Veno equation indeed approximates the experimental results quite well.

Figure 2(a) illustrates a simulation topology, where eight TCP Veno flows (Flow $1\sim8$) run from the left side network to the right side network through the bottleneck link. The left side network has wired links with bandwidth of 10 Mbps, and delay of 1 ms. The right side network is a heterogeneous network with wired and wireless links, whose bandwidths are all 10 Mbps. The delays of the wired links are $(20\times i-19)$ ms, $i=1,2,...,8$, and the delays of the wireless links are all 1 ms. Random losses in wireless links follow the exponential distribution and the packet loss rate ranges from 0.0001 to 0.1. The bottleneck link between these two networks is a wired link with bandwidth of 12 Mbps and delay of 10 ms. Its packet buffer size is set to be 40. Packet sizes of TCP Veno flows are 1000 bytes. We also set up 8 UDP connections as background traffic in our tests. These UDP flows follow Pareto distributions [18]. The burst time and idle time are both 100 ms. The rate of each connection is 300 Kbps and the packet size is 512 bytes.

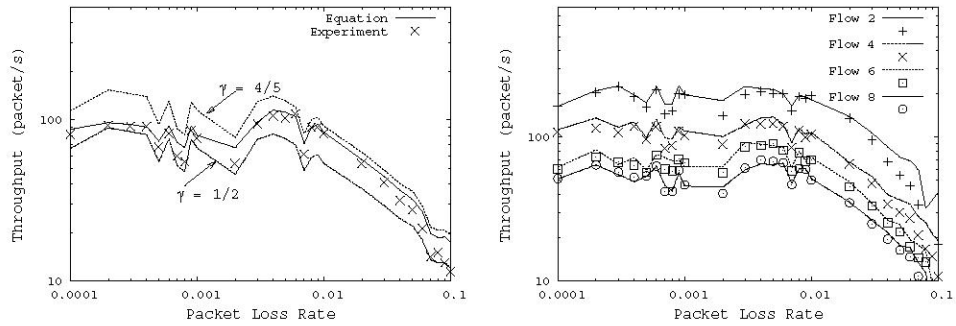
Figure 3(a) gives the validation result on one of the TCP Veno flows – Flow 5. The theoretical results are quite consistent with the experimental results. We also plot the curves when $\gamma=\theta_2=\frac{1}{2}$ and $\gamma=\theta_1=\frac{4}{5}$, which are the minimum and maximum boundaries of the Veno equation. Observing Figure 3(a), we find that almost all experimental results indeed fall into a region between these two curves. Figure 3(b) plots the validation results of Flow 2, 4, 6, 8. Obviously, all these experimental results can be accurately predicted by the Veno equation.

In order to further validate this model, the live Internet experiments are conducted over cross-country WANs. We set up a wireless client at Nanyang Technological University (NTU) in Singapore, and a TCP Veno server at The University of Hong Kong (HKU) in Hong Kong, as seen in Figure 2(b). One TCP Veno flow runs over the link from the server to the client, and the packet size is 512 bytes. In the wireless side, the client is a Compaq laptop with W200D wireless card. The wireless Access Point is a Cisco Linksys WRT54G. They are placed in the different corners of a room and the distance between them is about 8 meters. The bandwidth of the wireless network is 11 Mbps and the frequency is 2.4 GHz. The measurement method in the Internet experiments is the same as that described in [3]. The 1 hour flow trace is divided into 36 consecutive 100-second intervals, and we count the number of packets transmitted, RTT, $BaseRTT$, and the packet loss rate p in every interval. Note that p is given as the total number of loss indications (TD and TO) divided by the number of packets transmitted. Then, we put the average values of RTT and $BaseRTT$ into the Veno equation to calculate the theoretical result. Due to space constraints, we only give two samples collected recently, as shown in Figure 4(a) and 4(b). The results demonstrate that the Veno equation can model TCP Veno's throughputs fairly well.



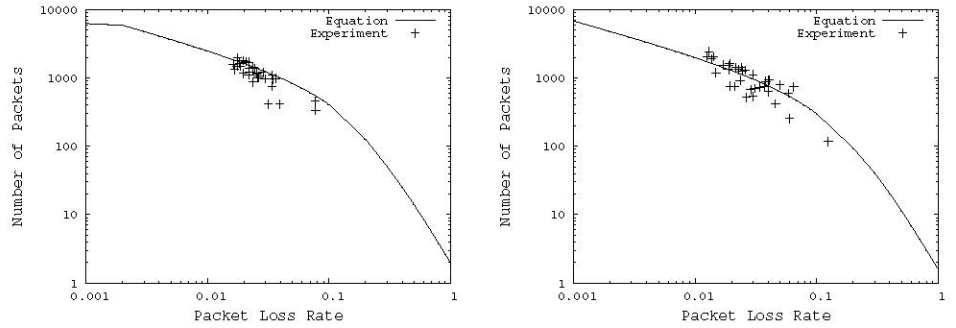
(a) (b)

Fig. 2. Topologies of (a) simulation experiments, and (b) Internet experiments.



(a) (b)

Fig. 3. Simulation results on (a) a single flow, and (b) multiple flows.



(a) (b)

Fig. 4. Internet experiment results.

B. TFRC Veno Mechanism

Figure 5 gives a brief introduction of the original TFRC mechanism. Specifically, TFRC lets the receiver estimate the packet loss rate p on the link and return such information to the sender through ACKs. After receiving the value of p, the sender adjusts its sending rate based on the Reno equation:

$$T_{calc} = \frac{S}{RTT\sqrt{\frac{2bp}{3}} + 3T_0\sqrt{\frac{3bp}{8}}p(1+32p^2)} \quad (21)$$

where T_{calc} is the expected sending rate in bytes/sec; S is the packet size in bytes; RTT is the round-trip time; p is the steady-state rate of loss event; T_0 is the retransmit timeout value.

TFRC Veno modifies TFRC by replacing the above Reno equation with the Veno equation, that is, the sender now

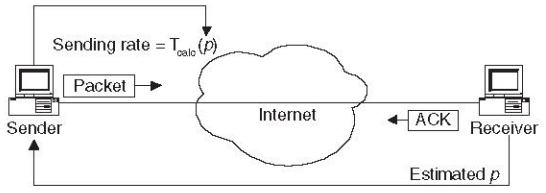


Fig. 5. Basic mechanism of TFRC.

adjusts its sending rate based on the following equation:

$$T_{calc} = \frac{S}{RTT\sqrt{\frac{2b(1-\gamma)p}{1+\gamma}} + 3T_0\sqrt{\frac{b(1-\gamma^2)p}{2}}p(1+32p^2)} \quad (22)$$

Note that here we ignore the effect of the maximum congestion window size in the Veno equation (16). Furthermore, the original Veno equation calculates throughput in packets/sec,

while here we multiply it by the packet size (S) to calculate throughput in bytes/sec. These two changes are the same as those of the Reno equation when used in TFRC. The other mechanisms in TFRC, such as slow start, packet loss estimation, and time-out mechanism, are not changed in TFRC Veno. Here γ is in the range $[\theta_1, \theta_2]$, with the settings of $\theta_1 = 0.95$ and $\theta_2 = 0.5$.

IV. Performance Evaluation

We conduct various experiments including both NS-2 simulation and live Internet experiments, to comprehensively evaluate the performance of TFRC Veno in terms of throughput, fairness, TCP-friendliness, and sending rate variation. We also compare our proposal with the original TFRC and another sender-modified enhancement MULTFRC [12]. The results in the experiments show that, TFRC Veno combines both advantages of TFRC and MULTFRC while avoiding their drawbacks, that is to say, TFRC Veno has much better throughput than TFRC while keeping the nice features of fairness, TCP-friendliness, and sending rate variation; TFRC Veno has much better fairness and TCP-friendliness than MULTFRC while having the same throughput as MULTFRC.

A. Simulation Experiments

The topology of the simulation experiments is depicted in Figure 6. The left side of the network has wired links with bandwidth of 10 Mbps, delay of 1 ms and buffer size of 50 packets. The right side of the network has wireless links with bandwidth of 10 Mbps, delay of 1 ms and buffer size of 50 packets. The bottleneck link between these two networks is a wired link with a bandwidth of 12 Mbps, and delay of 80 ms. Its buffer size is set to be 20 packets. Three types of flows – TFRC, MULTFRC, and TFRC Veno – are transferred from the wired network to the wireless network. Packet sizes are all 1000 bytes. Furthermore, we set up 8 UDP connections with pareto distribution [18] over the bottleneck link as the background traffic. The burst time and idle time are both 100 ms. The packet size of UDP is 512 bytes and the rate of each connection is 300 Kbps. Each experiment lasts 300 s, and is repeated 20 times. We plot the average of these results in the following figures and use the error bars to represent 95% confidence intervals.

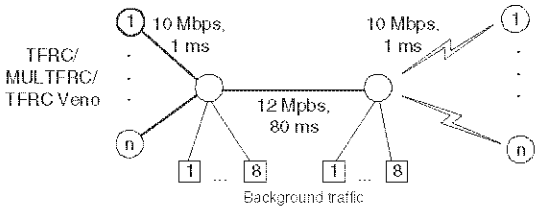


Fig. 6. Topology of simulation experiments.

In order to fully evaluate the performance of TFRC Veno, our simulation uses two different random loss models – exponential distribution and two-state distribution – in the wireless links. In the exponential distribution, packet loss rate ranges from 0.0001 to 0.1. In the two-state distribution, the good state has packet loss rate of 0.00001 and period of 8 s, and the bad state has packet loss rate ranging from 0.0001 to 0.1 and period of 4 s. Furthermore, the good state has 95% probability to transit to the bad state after each period, and the bad state transits to the good state also with 95% probability.

1) Throughput: Firstly we let 8 target (TFRC or MULTFRC or TFRC Veno) flows run in the experiment and calculate their average throughputs. Here the throughput is counted as the number of packets sent divided by the transmission time. Figure 7(a) and 7(b) plot the throughput comparisons of these three protocols – TFRC, MULTFRC, and TFRC Veno – under different types of random losses. Note that the packet loss rate in Figure 7(b) corresponds to the bad state packet loss rate in the two-state distribution. As shown in these two figures, TFRC Veno has a similar throughput as MULTFRC: its throughput is a little lower than that of MULTFRC when random loss follows the exponential distribution while a little higher when random loss follows the two-state distribution. Furthermore, both TFRC Veno and MULTFRC has much better throughput than TFRC. In particular, the throughput improvement can be up to 300% over TFRC when the packet loss rate is 0.1.

Secondly, we vary the number of target flows from 4 to 32 and study their performance. The random packet loss rate is set to 0.01. Observing Figure 8, our proposal also has a similar performance as MULTFRC under different numbers of connections. As the number of flows increases, which means the congestion is gradually dominating the network, both TFRC Veno and MULTFRC become having the similar throughput performance to TFRC.

2) Fairness: Fairness means the same kind of flows should share the total bandwidth fairly. Figure 9(a) and Figure 9(b) plot the sequence numbers of 16 MULTFRC flows and 16 TFRC Veno flows respectively, when they run over a network where random losses follow the exponential distribution and the loss rate is 0.01. Comparing these two figures, TFRC Veno flows share the bandwidth equally while the evolution of the flows among MULTFRC connections may have significant difference.

In order to have a better fairness comparison among TFRC, MULTFRC and TFRC Veno, Jain's Fairness Index f [19] is used here:

$$f = \left(\sum_{i=1}^{n} x_i\right)^2 / (n \sum_{i=1}^{n} x_i^2) \tag{23}$$

where n is the number of connections, x_i is the throughput of the ith connection. The closer f is to 1, the more fairness the target flows enjoy. It should be pointed out that f is not a sensitive function. That is to say, a quite different fairness situation can only result in a little variation of f. For example, f of MULTFRC in Figure 9(a) is only 0.03 less than that of TFRC Veno in Figure 9(b), but the gap between the fastest flow and the slowest flow of MULTFRC is 6 times larger than that of TFRC Veno.

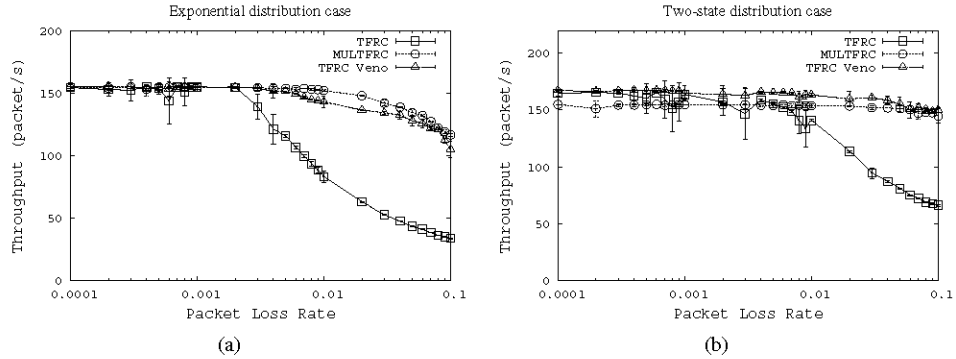


Fig. 7. Throughput comparisons under different random loss rates: (a) exponential distribution case, and (b) two-state distribution case.

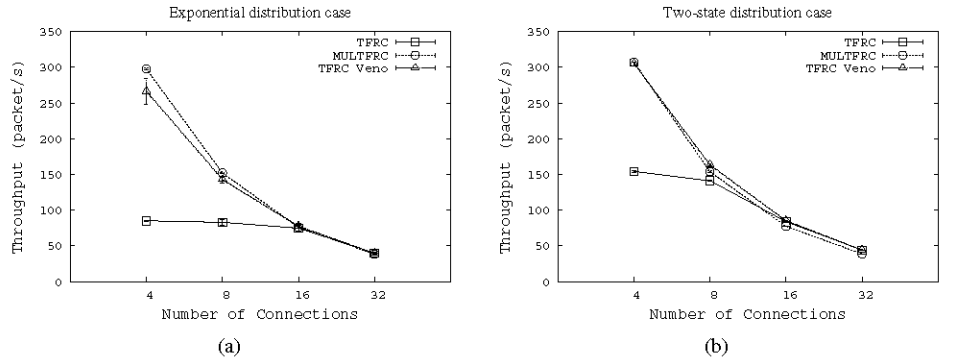


Fig. 8. Throughput comparisons under multiple connections (packet loss rate is 0.01): (a) exponential distribution case, and (b) two-state distribution case.

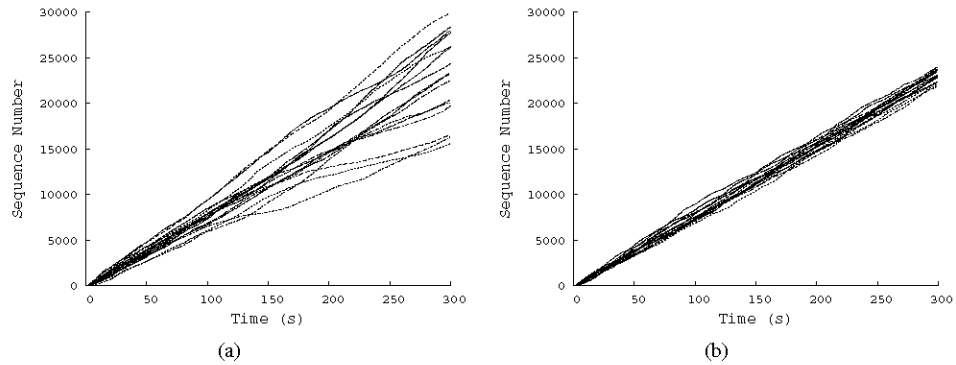


Fig. 9. Sequence number against time, 16 connections, packet loss rate is 0.01. (a) MULTFRC, f=0.967; (b) TFRC Veno, f=0.998.

We set the number of target flows to 8, and study the fairness of the three protocols under different random packet losses. As seen in Figure 10, TFRC Veno has a good fairness as in TFRC (the fairness index is near 1). Both TFRC Veno and TFRC have better fairness than MULTFRC.

Then we set the packet loss rate to 0.01, and vary the number of target flows from 4 to 32. As the number increases, TFRC Veno keeps a good fairness as in TFRC. However, the fairness of MULTFRC becomes increasingly poor. Figure 11 plots these results.

3) TCP-friendliness: TCP-friendliness measures whether the target flows are aggressive or not when competing with TCP flows. It can be studied as follows: we first set up a certain number of TCP Sack flows over the link, and calculate their average throughput T_1. Then we replace half of them with the target flows and recalculate the average throughput of the remaining TCP Sack flows T_2. If $\frac{T_2}{T_1}$ equals to 1, it means Sack flows are not affected by the target flows, and thus the target flows are totally friendly. The closer $\frac{T_2}{T_1}$ is to 1, the more friendly the target flows are.

Figure 12 shows the results when there are totally 8 connections, and random packet loss ranges from 0.0001 to 0.1. Figure 13 plots the results under different numbers (ranging from 4 to 32) of connections, where the random packet loss rate is 0.01. As shown in these figures, TFRC Veno flows are not harmful to the existing TCP Sack flows when random

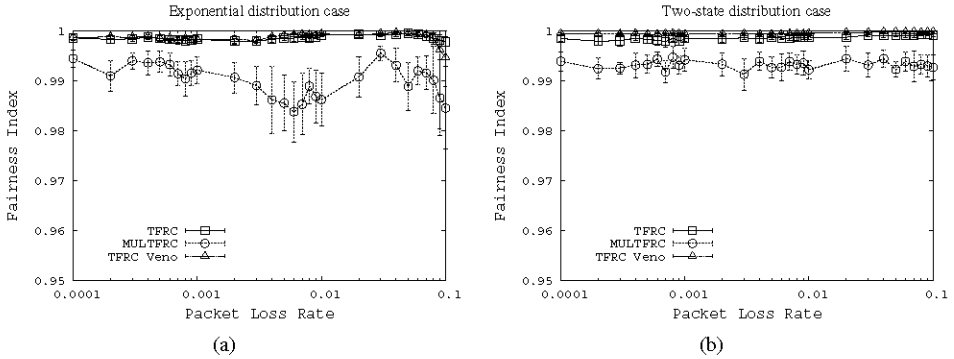


Fig. 10. Fairness comparisons under different random loss rates: (a) exponential distribution, and (b) two-state distribution.

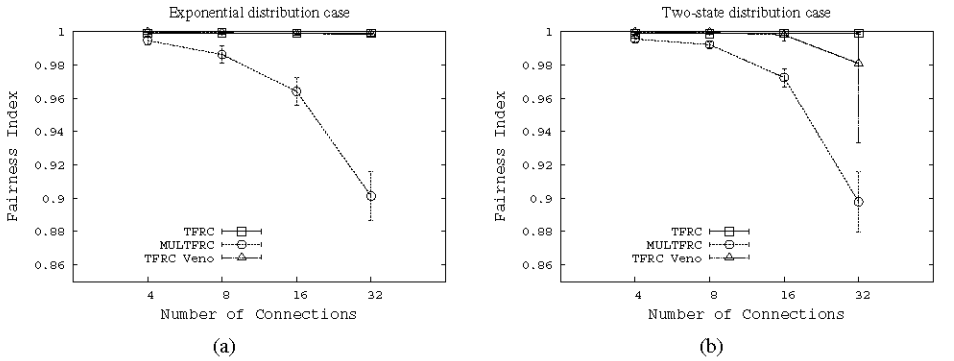


Fig. 11. Fairness comparisons under multiple connections (packet loss rate is 0.01): (a) exponential distribution, and (b) two-state distribution.

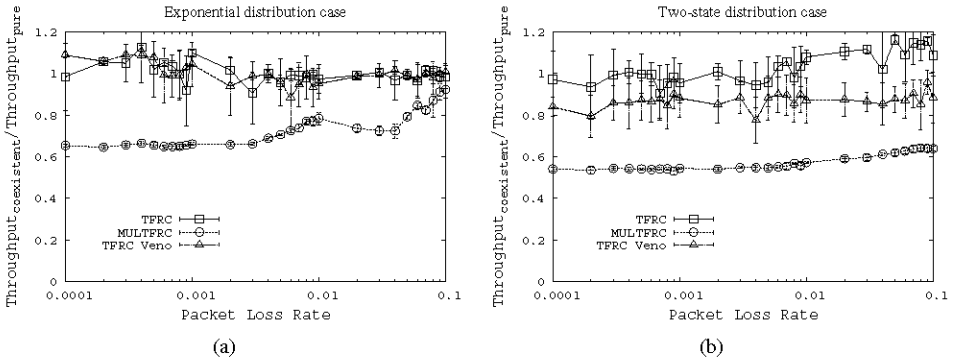


Fig. 12. TCP-friendliness comparisons under different random loss rates: (a) exponential distribution, and (b) two-state distribution.

loss follows the exponential distribution ($\frac{T_2}{T_1}$ is always around 1), and are acceptable when random loss follows the two-state distribution ($\frac{T_2}{T_1}$ is close to 0.9). On the other hand, MULTFRC is always very aggressive in either cases (because $\frac{T_2}{T_1}$ is less than 0.6).

4) Sending Rate Variation: As a transport protocol for streaming multimedia transmission, the sending rate of the target flows should not vary excessively. Here, we use the coefficient of variation (CoV) to measure this variation. CoV is defined as the standard deviation divided by the mean. The smaller CoV is, the smoother the sending rate is.

In the experiment, a certain number of the target flows run with the same number of TCP Sack flows. We count the number of packets sent in the target flows within every 0.1 s as samples, and finally calculate CoV of all these samples. Figure 14 shows the result when 4 target flows run with 4 TCP Sack flows, and Figure 15 plots the result when the total numbers of flows (half are the target flows) increases from 4 to 32 where the packet loss rate is 0.01. The results demonstrate that MULTFRC has a smoother sending rate in most cases. However, TFRC Veno can also keep CoV at small values, which are almost the same as those of TFRC.

In summary, through the study in Section IV-A.1, IV-A.2, IV-A.3, and IV-A.4 we can conclude that, TFRC Veno is an efficient enhancement of TFRC since the significant throughput improvement in TFRC Veno does not sacrifice

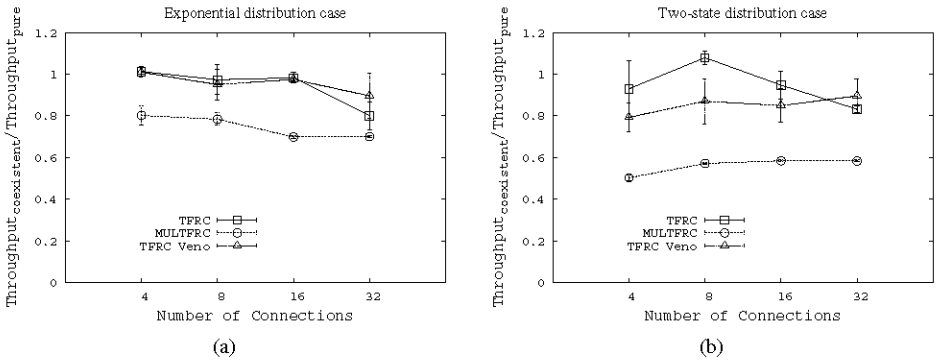


Fig. 13. TCP-friendliness comparisons under multiple connections (packet loss rate is 0.01): (a) exponential distribution, and (b) two-state distribution.

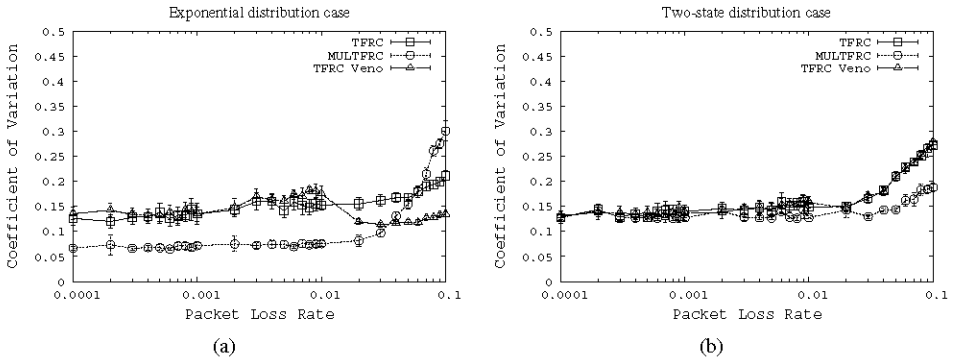


Fig. 14. CoV comparisons under different random loss rates: (a) exponential distribution, and (b) two-state distribution.

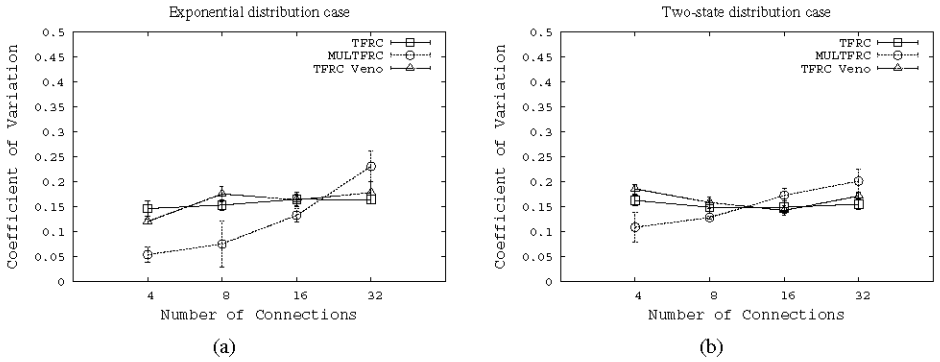


Fig. 15. CoV comparisons under multiple connections (packet loss rate is 0.01): (a) exponential distribution, and (b) two-state distribution.

the other desirable features of TFRC such as fairness, TCP-friendliness, and sending rate smoothness.

B. Internet Experiments

We use the same link between NTU and HKU as shown in Figure 2(b). The TFRC/TFRC Veno client is at NTU and the TFRC/TFRC Veno server is at HKU. At first, we report the single connection results. We select six time slots (1 hour period) over a day. In each time slot, we first let a TFRC flow run from the server to the client for 5 minutes, and then let a TFRC Veno flow run in the same direction for 5 minutes. After a 2-minute break, we repeat such test. Thus totally there are five tests in each time slot, and we calculate the average results of TFRC and TFRC Veno, respectively. Three measurements are calculated here: the number of packets sent, the receiving ratio, and the CoV. Note that the packet size is 500 bytes during transmission. We performed five days of testing from Monday to Friday. Here we only present the results of one day because the general trend of the results in each day is similar. As depicted in Table I, TFRC Veno can obtain throughput improvements of up to 70% over TFRC, with the same receiving ratio.

We also test TFRC Veno's performance when coexisting with one competing TCP Sack flow. We select five time slots (1 hour period) over a day. In each time slot, we sequentially let one TFRC Veno flow, one TFRC flow, and one TCP Sack

TABLE I

INTERNET EXPERIMENT RESULTS ON SINGLE FLOW.

TFRC/TFRC Veno						
Time Slots	9:00∼10:00	11:00∼12:00	13:00∼14:00	15:00∼16:00	17:00∼18:00	22:00∼23:00
Packets Sent	2916.8/4128.6	2458.3/3249.7	2037.3/2981.3	2165.3/2640.0	3471.2/3507.4	1058.3/1812.7
Receiving Ratio	0.83/0.83	0.84/0.83	0.78/0.81	0.80/0.81	0.90/0.88	0.71/0.73
CoV	0.33/0.37	0.27/0.22	0.19/0.24	0.25/0.31	0.13/0.11	0.34/0.29

TABLE II

INTERNET EXPERIMENT RESULTS ON COEXISTING FLOWS.

Time Slots	10:30∼11:30	13:30∼14:30	16:30∼17:30	19:30∼20:30	21:30∼22:30
TFRC Veno/TCP Sack	1469.8/1132.3	1657.5/1253.3	2038.7/1202.5	2212.3/1357.8	1762.5/1224.5
TFRC/TCP Sack	1462.3/1340.0	1159.3/1179.8	1496.8/1343.3	1609.8/1525.3	1435.5/1349.0
TCP Sack/TCP Sack	1390.0/1021.8	1328.3/1347.5	1364.8/1404.8	1299.0/1312.8	1225.3/1458.8

flow run with an existing TCP Sack flow for 5 minutes. This test will be repeated four times in each time slot, and we calculate the average number of packets transmitted in each flow. Note that all flows have same packet sizes of 500 bytes and are transferred from the same server to the same client. Since the general trend of the results in each day of the entire five-day period is similar, we only present the results of one day. As shown in Table II, the TFRC Veno flow does not cause any bias degradation to the competing TCP Sack flow, and obtains higher throughput while compared to the TFRC flow.

V. CONCLUSION AND FUTURE WORK

TFRC is designed to provide optimal service for unicast multimedia flow operating in the best-effort Internet environment. In wireless environments TFRC suffers throughput degradation due to its embedded Reno equation. In this paper, we replace the Reno equation with the Veno equation, which is derived from the throughput model of TCP Veno, to enhance TFRC performance. The comprehensive experiments show that our enhancement, TFRC Veno, can improve TFRC performance significantly over wireless networks for multimedia transmission. As compared to another sender-modified enhancement MULTFRC, our proposal is more friendly to other network traffic when achieving the same high throughput.

The performance of TFRC Veno depends on the Veno parameter γ, which is a function of the parameters θ_1, θ_2, and β in TCP Veno. In the future, we will study how these parameters impact the performance of TFRC Veno, i.e., the throughput, fairness, TCP-friendliness, and sending rate smoothness of TFRC Veno.

VI. ACKNOWLEDGMENTS

We would like to thank Prof. Soung Chang Liew for his valuable comments and Prof. Lawrence K. Yeung for his kind help in setting up the Internet test bed.

REFERENCES

[1] V. Jacobson. "Congestion Avoidance and Control." ACM SIGCOMM'88, August 1988.
[2] S. Floyd, M. Handley, and J. Padhye et al. "Equation-Based Congestion Control for Unicast Applications." ACM SIGCOMM'00, August 2000.
[3] J. Padhye, V. Firoiu, and D. Towsley et al. "Modeling TCP Throughput: A Simple Model and its Empirical Validation." ACM SIGCOMM'98, August 1998.
[4] T. V. Lakshman, and U. Madhow. "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss." IEEE/ACM Trans. on Networking, Vol. 5, No. 3, 1997.
[5] H. Balakrishnan, V. Padmanabhan, and S. Sechan et al. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links." ACM SIGCOMM'96, 1996.
[6] H. Balakrishnan. "Challenges to Reliable Data Transport over Heterogeneous Wireless Networks." Ph.D thesis, University of California, Berkeley, 1998.
[7] C. P. Fu, and S. C. Liew. "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks." IEEE Journal of Selected Areas in Communications, Vol. 21, Issue 2, 2003.
[8] R. Chaudhary, and L. Jacob. "ECN based TCP-friendly rate control for wireless multimedia streaming." IEEE ICCCN'03, October 2003.
[9] L. Huang, U. Horn, and F. Hartung et al. "Proxy-based TCP-friendly streaming over mobile networks." ACM WoWMoM'02, September 2002.
[10] J. Y. Pyun, Y. Kim, and K. H. Jang et al. "Wireless Measurement Based Resource Allocation for QoS Provisioning over IEEE 802.11 Wireless LAN." IEEE Trans. on Consumer Electronics, vol. 49, Issue 3, 2003.
[11] V. Arya, and T. Turletti. "Accurate and Explicit Differentiation of Wireless and Congestion Losses." IEEE ICDCSW'03, May 2003.
[12] M. Chen, and A. Zakhor. "Rate control for streaming video over wireless." IEEE INFOCOM'04, March 2004.
[13] J. Crowcroft, and P. Oechslin. "Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP." ACM Computer Communication Review, Vol. 28, Issue 3, 1998.
[14] M. Chen, and A. Zakhor. "AIO-TFRC: A light-weighted rate control scheme for streaming over wireless." IEEE WirelessCom'05, June 2005.
[15] Linux Kernel. Available at http://www.kernel.org/
[16] B. Zhou, C. P. Fu, and D. M. Chiu et al. "A Simple Throughput Model for TCP Veno." IEEE ICC '06, June 2006.
[17] The Network Simulator. Available at http://www.isi.edu/nsnam/ns/
[18] NS-2 Manual. Available at http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf
[19] R. Jain. "The art of computer systems performance analysis." Wiley, New York, 1991.