

Enhancement of IFTP for Transmission over Wireless Access Networks

Hala ElAarag Chris Hogg
Department of Mathematics and Computer Science
Stetson University
DeLand, Florida, U.S.A.
Email: {helaarag, chogg}@stetson.edu

Abstract

This paper presents a new end-to-end transport protocol designed to enhance the performance of Internet friendly multimedia applications transmitted over a wireless network. This protocol is an enhancement of Internet Friendly Transport protocol (IFTP) that we designed to alleviate the TCP-friendly problem. Similar to IFTP, the proposed protocol is designed to avoid congestion, as indicated by either retransmission timer expiration or the receipt of multiple duplicate ACKs, by reducing the transmission flow. However, due to the relatively high bit error rate of most wireless mediums, packet drops are often not the result of network congestion. Hence, they result in unnecessary reduction of throughput. The proposed protocol differentiates between packet loss due to congestion and wireless errors and so maintains higher throughput in such environments. In addition, it still maintains its fairness and TCP-friendliness properties.

1. Introduction

Wireless connections are becoming increasingly important as portable devices have growing needs for Internet connections. These connections now have decent bandwidths however they suffer from high bit error rates (BER). This is usually due to two main reasons: either due to object obstruction of signal propagation or the disconnection of the mobile host during handoff. This causes packets to be corrupted. Traditionally, transport protocols like TCP-Tahoe and Reno were not designed with wireless problems in mind and hence their performance is greatly degraded in such environments. TCP is designed to interpret an out of order or duplicated ACK as a sign of congestion and so it drops its transmission rate. Thus despite potentially having a lot of available bandwidth it decreases the number of packets it sends in a given round trip time (RTT) and so do not fully utilize the wireless link and hence suffer serious throughput reduction [1].

The TCP-friendly problem is another problem in the field. It happens when a transport protocol with no flow control, like UDP, share a bottleneck with a well-behaved

protocol like TCP monopolizing the available bandwidth. IFTP [4] is a best-effort flow control protocol we designed to solve this problem by calculating a TCP-friendly sending rate. However, IFTP is designed for wired networks.

The protocol presented in this paper is an end-to-end protocol that tries to solve the two aforementioned problems. It utilizes client side information to determine if packet loss is due to congestion or random loss due to a wireless problem. This is done by using estimations of how long a given window should take to transmit and comparing it to how long it actually took to transmit. The rest of the paper is organized as follows. Section 2 presents some of the previous research in this area. In section 3 we briefly explain IFTP and IFTP-W [5, 8]; our previous proposal to enhance IFTP's performance on wireless networks. Section 4 discusses the proposed protocol IFTPV. Sections 5 and 6 explain the network simulator and the simulation results, respectively. Finally section 7 concludes the paper.

2. Previous Research

The issue of enhancing the performance of transport protocols in wireless environments with high bit error rates motivates many researchers in the field. There are several proposed solutions to this problem. ElAarag [3] classifies the proposed solutions to improve the performance of TCP on wireless networks into three categories: link layer, end-to-end and split. Each category has its advantages and disadvantages but none of the proposals solve all the problems that arise from wireless links [3]. TCP-ELN [2] is based on Explicit Loss Notification to notify the sender about packet losses on the wireless channel.

Recently, there are four proposed TCP protocols optimized for wireless networks and based on the end-to-end paradigm. They are JTCP [9], TCP-Jersey [10], TCP Westwood [11] and TCP-Veno [7]. JTCP uses the inter-arrival jitter and the jitter ratio to infer network congestion. While TCP-Jersey estimates the congestion window after the loss has occurred using the available bandwidth estimator algorithm. TCP Westwood frees TCP from the traditional Additive Increase Multiplicative Decrease

(AIMD) algorithm and rather statistically estimates the bandwidth with each packet sent. TCP-Veno is an improvement on TCP-Reno. It calculates the rate as the ratio of the congestion window to the RTT. It uses two main parameters. One is the expected rate which uses the best RTT, and the other is the actual rate which uses the last measured RTT. Todorovic and Benitez [1] compare these four proposals and conclude that there is no one solution that showed superiority in all scenarios. Their results are not conclusive as they mention that the choice of the different parameters could provide different results.

There are also several proposals to the TCP-friendly problem. A good survey of TCP-friendly protocols can be found in [12].

3. IFTP and IFTP-W

IFTP [4] is an Internet protocol that was designed to provide an Internet friendly transmission protocol for multimedia applications. By Internet friendly we mean that it scales its transmission rate in a fashion similar to TCP as network congestion is introduced [4]. Such a protocol was needed because most multimedia applications are transmitted using UDP, which is considered a non-TCP-friendly protocol. In addition to its TCP-friendliness property, IFTP has a very useful feature which is its ability to keep a low jitter and hence improves the performance of multimedia applications. IFTP emulates TCP in almost all of its operations except in retransmitting lost packets. It does that by increasing its congestion window in a fashion similar to the three phases of TCP namely slow start, congestion avoidance and maximum window. It follows the AIMD algorithm and hence halves the number of packets in the window upon packet loss. In every round trip, it recalculates the inter-packet gap so that the receiver gets the packets of the same window at even intervals.

Assume that IFTP sends n packets in the i^{th} period. It then calculates the inter-packet gap for $i+1^{\text{th}}$ period G_{i+1} , which is the time to wait in between sending the last bit of one packet and the first bit of the next packet, according to equations 1-3.

Slow Start Phase:

$$G_{i+1} = \frac{R}{2n} - \tau \quad \dots\dots\dots (1)$$

Congestion avoidance phase:

$$G_{i+1} = \frac{R}{n+1} - \tau \quad \dots\dots\dots (2)$$

Packet Loss:

$$G_{i+1} = \frac{2R}{n} - \tau \quad \dots\dots\dots (3)$$

Where,

R: is the smoothed round trip time.

τ : is the time it takes to send a single packet in seconds

Note that IFTP assumes a WAN Internet environment and hence $R \gg \tau$ TCP window.

IFTP-W [5] is a proposal to improve the performance of IFTP on wireless networks. It is based on IFTP but designed to allow for a greater percentage of packets to be accepted as viable despite having errors. It does this because on a wireless link with a high bit error rate, packets are often partially corrupted rather than lost. TCP is a reliable protocol and hence retransmits corrupted or lost packets. However IFTP does not retransmit corrupted or lost packets as multimedia applications need in order transmission. Retransmitting a particular packet disrupts this order and also increases the inter-packet delay and delay jitter. IFTP-W allows the sender to specify a section as being error-sensitive. Bit errors not occurring in this section are ignored. IFTP-W does not decrease its sending rate as often as IFTP would on a connection with a high bit-error rate. IFTP-W follows the rationale that dropping packets because of bit-errors in the error insensitive portion would cause worse disruption than accepting these packets and ignoring the bit errors that might have changed the colors of a few pixels or altered a millisecond of sound.

4. IFTPV

IFTP-W was shown to perform better than IFTP on wireless networks [5, 8]. However, it was still, like IFTP, faithful to TCP-Reno to maintain its TCP-friendliness property. TCP-Reno is inherently inefficient on wireless networks as it interprets every packet loss as congestion. The recent proposals of TCP protocols optimized for wireless networks, as explained in section 2, seemed encouraging. Although the techniques are different, they all attempt to differentiate between errors due to congestions and those due to the wireless link. It appears to be logical to have IFTP faithful to one of these proposals instead of TCP-Reno to improve its performance on wireless networks. However, we do not want to sacrifice the IFTP's fairness and TCP friendliness properties. Kaneko and Kato [13] proved that TCP Westwood's friendliness with TCP Reno is deteriorated according to network situations such as the buffer size of a bottleneck link router. TCP Jersey requires ECN capable routers in the network [10]. The authors of JTCP [9] mention that its fairness is not the best. However, TCP Veno [7] is the most compatible to TCP Reno which the original IFTP protocol is based on. Also Zhou et al. [6] demonstrated that TCP Veno [6] is fair to TCP Reno.

In this paper, we propose IFTPV which enhances the performance of IFTP protocol by being faithful to TCP Enhanced Veno [6] which is a modification from TCP-Veno [7]. IFTPV maintains two variables; the backlog, B , and the congestion rate, con_r . The first variable, B , estimates the numbers of packets in a connection until a packet loss occurs i.e. tracks the number of packets suspected to wait in a queue. IFTPV determines this by keeping track of the round trip times and how long a packet would take to be acknowledged. When it seems that a given window took longer to transmit than its size indicated then a packet must have spent time in a router's queue. Thus B is calculated according to equation 4, where RTT_{base} is the minimum of measured round trip times, RTT_{actual} is the measured smoothed RTT and n is the number of packets transmitted in a round trip.

$$B = \left(\frac{n}{RTT_{base}} - \frac{n}{RTT_{actual}} \right) \times RTT_{base} \dots\dots (4)$$

If the estimated backlog is greater than or equal to a certain threshold, β , then a packet loss indicated by a duplicate ACK is due to congestion and hence the number of packets transmitted in the next period, n , is halved. Otherwise the network is assumed to be in a random loss state rather than a congestion state. Fu's et al. [7] experiments indicate that a good value for β is 3. The second maintained variable by IFTPV is the congestion loss rate, con_lr , which determines if during the random loss state the network is congestive. The congestion loss rate is calculated by keeping track of the number of packets lost to congestion (C) in between three consecutive packet losses due to link error, and dividing it by the time between their associated link error losses. For example, consider that the three consecutive random losses due to link error occurred at times T_1 , T_2 and T_3 , respectively. Now assume that between T_1 and T_2 , the number of packet lost due to congestion is C_{prev} and that between T_2 and T_3 , this number is equal to $C_{current}$. con_lr_{prev} and $con_lr_{current}$ are defined as shown in equations 5 and 6.

$$con_lr_{prev} = \frac{C_{prev}}{T_2 - T_1} \dots\dots (5)$$

$$con_lr_{current} = \frac{C_{current}}{T_3 - T_2} \dots\dots (6)$$

When a random loss occurs these values are updated.

IFTPV is interested in determining whether during the random loss state ($B < \beta$) the congestion loss rate is increasing or decreasing. If it is increasing IFTPV protocol decreases its number of transmitted packets by one fifth in an attempt to avoid congestion, however if the rate is not increasing this number would stay unchanged. IFTPV still

treated a timeout as a sign of extreme network condition and so acted the same way as IFTP, which is the same as TCP, by halving the slow start threshold and entering the slow start phase with a window equal to 1.

The pseudo code of how IFTPV alters the number of packets transmitted in a given period when a duplicate ACK is received is shown below:

```

if (B < β && con_lr_current ≤ con_lr_prev) {
    n = n;
} else if (B < β && con_lr_current > con_lr_prev) {
    n = n * (4/5);
} else if (B ≥ β) {
    n = n * (1/2)
}

```

Where,

- B is the estimated number of backlogged packets
- β is a threshold set to be equal to 3
- $con_lr_{current}$ is the number of packets lost due to congestion per unit time between the last two packets lost due to link error
- con_lr_{prev} is the number of packets lost due to congestion per unit time between the two previous losses due to link error.
- n is the number of packets transmitted in the current window.

5. The Network Simulator

We studied the performance of IFTPV on wireless networks through simulation. The discrete event network simulator that we developed in [5, 8] was used. This simulator is object oriented and was written in C#.

The Network Topology is illustrated in Figure 1. It is the typical one router bottleneck dumbbell. In this topology, there are multiple servers with their corresponding clients sharing the bottleneck router. The router uses FCFS scheduling and drop-tail queuing. Each wired link has a delay of 10 milliseconds and each wireless link has a delay of 100 milliseconds. This represents a wireless WAN with long wireless delay as it is explained in the assumption noted in section 3. For IFTP, IFTP-W, IFTPV and TCP, data packets are 1024 bytes, acknowledgement packets are 40 bytes, unless otherwise stated. The slow start threshold is 32, maximum window size is 50, and three duplicate ACKs initiate fast recovery. UDP packets are also 1024 bytes in length and are sent at

an average rate of 1 megabit per second with an exponentially distributed inter-packet gap. The links between the sources and the router are wired and have a bandwidth of ten megabits per second, while the links between the router and the corresponding receivers are wireless with a bandwidth of one megabit per second. All presented results are averaged from 60 runs of simulated connections running for approximately 10 seconds for each value of BER.

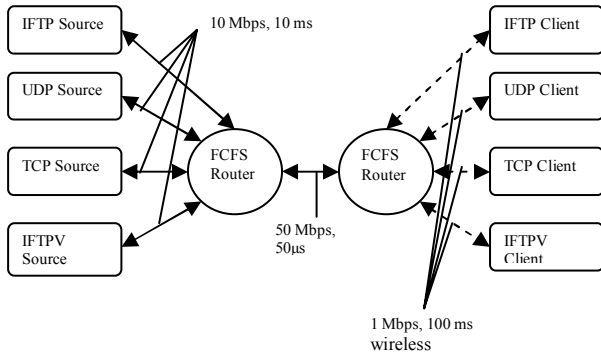


Figure 1: Network Topology

Several performance metrics were used, namely, throughput, goodput and delay jitter. For the sake of space limitation, only the results for the throughput are reported in this paper. Throughput is defined as number of bits per second received at each receiver.

6. Simulation Results

We studied the performance of IFTPV extensively in various network conditions. The results of six different scenarios are reported below.

6.1 Scenario 1:

In this scenario we study the performance of each protocol flow, UDP, TCP, IFTP, and IFTPV, as it exists alone in the network. We focus on the wireless problems and we study the effect of the BER on the performance of IFTPV and compare it to the other protocols. The network topology in this case has a single source and its corresponding client sharing the bottleneck router. In this scenario, the router has an infinite buffer to ensure that all packet drops are caused by the wireless link and no packet is lost due to the network congestion.

We varied the BER values from 10^{-3} to 10^{-7} . The BER value represents the ratio of bits experiencing corruption. Thus, a BER of 10^{-3} is equivalent to one bit corruption out of 1000 bits. Our simulator models the occurrence of bit

errors with an exponential distribution. The average throughput of UDP was much higher than the other three protocols. They were not plotted on the graph in order to zoom in and distinguish between the other three protocols. The corresponding throughput for UDP for BER 10^{-3} to 10^{-6} was found to be 0.92, 0.891, 0.899, and 1.0 Mbps, respectively.

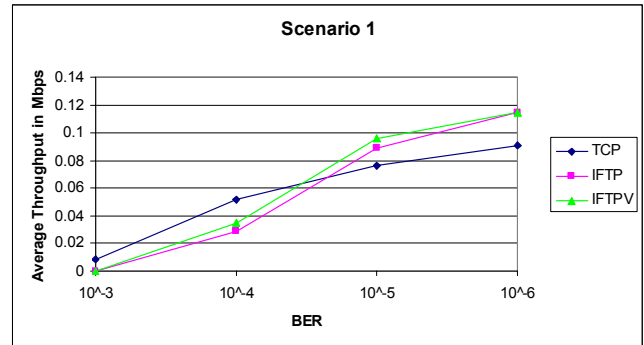


Figure 2: Average throughput of protocols as they stand alone in a wireless network with various BER

It can be seen from Figure 2 that IFTPV manages to maintain a better throughput than IFTP. It can also be seen that IFTP and IFTPV can outperform TCP, in networks with BER lower than 10^{-5} .

6.2 Scenario 2:

Scenario 1 gives us insight on the performance of IFTPV when run alone in comparison to UDP, TCP, and IFTP in reaction to various BER. However, the main objective of IFTPV is to improve the performance of IFTP as it distinguishes between losses due to congestion and those due to the wireless link errors. In this scenario we ran experiments to study the effect of both congestion and wireless problems. In this scenario IFTPV had to fight for bandwidth with competing UDP, TCP and IFTP flows. To illustrate the effect of congestion the buffer size was limited to 50K. The average throughput of UDP was not plotted in the graph for the same reason mentioned earlier. The corresponding throughput for UDP for BER 10^{-3} to 10^{-6} was found to be 0.928, 1.005, 1.015, and 1.01 Mbps, respectively.

It is clear from Figure 3 that TCP is better at handling congestion however it can also be seen that IFTPV manages to outperform IFTP.

6.3 Scenario 3:

In this scenario IFTPV suffers from both congestion and wireless problems as well. The buffer size was limited to 50K as in scenario 2. However, this scenario represents

UDP-based congestion. The BER was set to 10^{-5} . We studied the performance of IFTPV as the sending rate of UDP is varied. The UDP sending rate was varied from 10 Kbps to 1000 kbps. We compared IFTPV's performance to that of TCP's. Figure 4 shows the effect it had on the throughput of TCP and IFTPV. The Figure demonstrates the TCP-friendliness property of IFTPV.

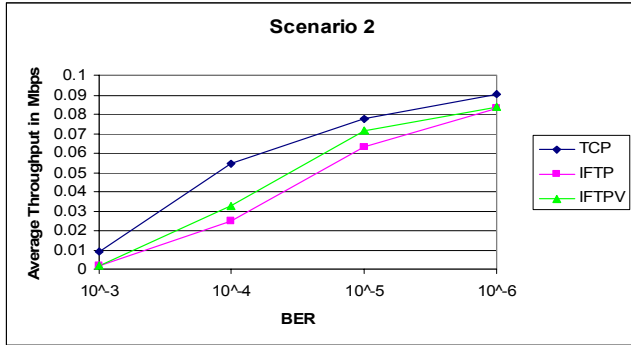


Figure 3: Average throughput of protocols in a congested wireless network with various BER

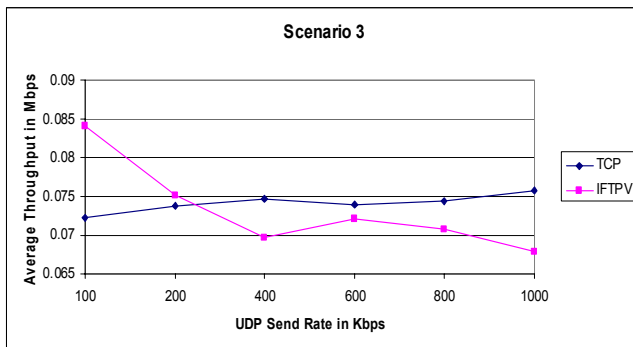


Figure 4: Average throughput in a wireless network with BER 10^{-5} and UDP based congestion

From Figure 4 it can be seen that IFTPV manages to have a transmission rate similar to TCP in highly congested networks. TCP and IFTPV have a similar share of the bandwidth and hence IFTPV is considered TCP-Friendly. However, IFTPV's throughput is mostly less than TCP, but it is possible for IFTPV to achieve greater rates than TCP. As mentioned in [4], IFTP, and hence IFTPV, can operate in an extended mode which gives selected flows a QoS-differentiated service.

6.4 Scenario 4:

This scenario is similar to scenario 3, but represents TCP based congestion. In this case IFTP connections competed with no UDP connections. This scenario demonstrates the fairness property of IFTPV. This is an

important feature of IFTPV, as we demonstrate its compatibility with TCP, the denominating transport protocol on the Internet today. In this scenario the network also suffers from congestion and BER.

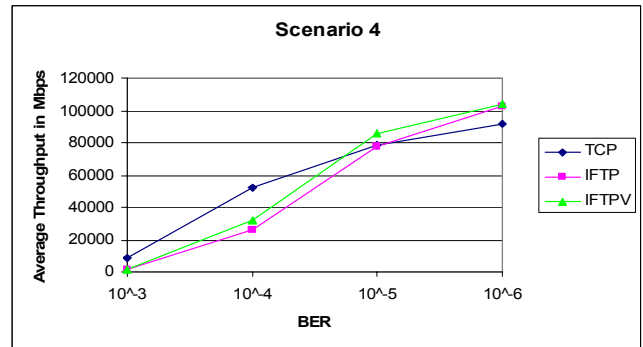


Figure 5: Average throughput in wireless TCP-based congested network with various BER

As can be seen in Figure 5 IFTPV performs better than IFTP in a TCP-based congested wireless network. IFTPV and TCP have a fair share of the bandwidth. IFTPV has on average a throughput lower than TCP in BER 10^{-3} to 10^{-5} , however it does eventually surpass TCP's throughput as BER lowers.

6.5 Scenario 5:

Scenarios 5 and 6 compare the performance of IFTP-W and IFTPV. In Scenario 5 we study their performance under various BER. Both protocols had a packet size of 1464 which is the default packet size for packets containing streaming video data over EDCF on 802.11e networks. The size of IFTP-W error sensitive part is fixed to 512.

In Figure 6 it can be seen that IFTPV outperforms IFTP-W in terms of throughput. Our experiments also show that IFTPV maintains a lower delay jitter and so would be the better choice for multimedia applications.

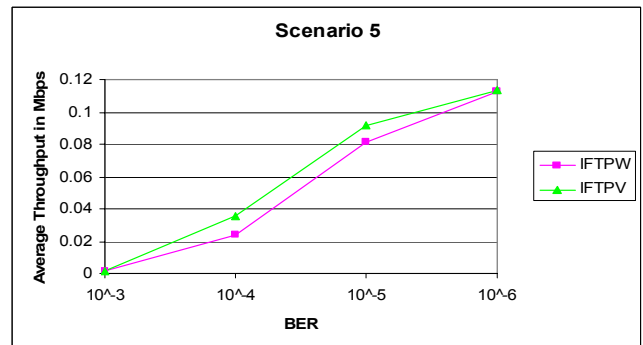


Figure 6: Comparison of average throughput IFTP-W and IFTPV under various BER

6.6 Scenario 6:

In this scenario the BER is fixed to 10^{-4} . Similar packet sizes as scenario 5 were used. We vary the length of the error sensitive part of IFTP-W from 40 which is the merely the header to 1464 which is the whole packet.

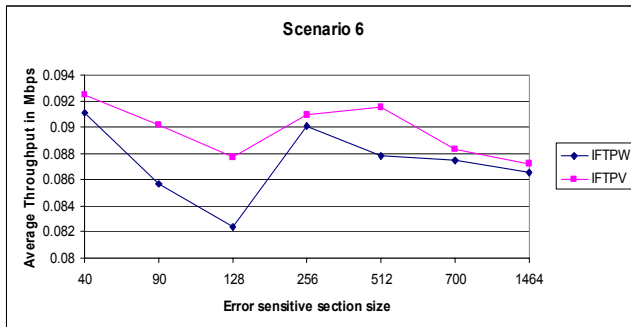


Figure 7: Comparison of IFTPV and IFTP-W with different lengths of error sensitive section

In Figure 7 it can be seen that IFTPV maintains an average throughput greater than IFTP-W no matter what IFTP-W's sensitive section length is. Our experiments also show that IFTPV did maintain a lower jitter in packet delay than IFTP-W. The results of these experiments are not included in this paper for space limitation.

7. Conclusions

In this paper we proposed a new protocol for the transmission of multimedia applications on wireless networks. We analyzed the performance of IFTPV in extensive scenarios with various network conditions. We demonstrated that IFTPV outperforms IFTP and IFTP-W, while maintaining the fairness and TCP-friendliness properties. It also maintained a low packet delay jitter on all networks and so makes for a good multimedia transmission protocol.

IFTPV's performance could be further enhanced by taking advantage of IFTP-W's ability to specify which section is error-sensitive. Furthermore, similar to IFTP, a scale factor can be easily incorporated into the logic of IFTPV to adapt the scheme to the individual needs of certain best-effort flows. With this simple modification, a best-effort flow is allowed to get a bandwidth exceeding that of its TCP counterpart (e.g., one and half or two times) but will still behave in a friendly manner and reduce its sending rate at times of congestion. This improves the capability of IFTPV to provide better QoS-differentiated services to selected traffic flows.

8. Acknowledgement

The authors would like to thank Andrew Moedinger, the original

author of the network simulator, for his help in the simulation of IFTPV.

The authors are grateful to the four anonymous reviewers for their comments and suggestions that helped improve the quality of this paper.

9. References

- [1] Milan Todorovic, Noé López-Benitez, "Efficiency Study of TCP Protocols in Infrastructured Wireless Networks", IEEE 2006
- [2] Georgó Buchholz, Thomas Ziegler, Tien Van Do, "TCP-ELN: On the Protocol Aspect and Performance of Explicit Loss Notification for TCP over Wireless Networks", WICON'05
- [3] Hala ElAarag, "Improving TCP performance over mobile networks," ACM Computing Surveys, 2002.
- [4] Hala ElAarag, Mostafa Bassiouni, "An Internet Friendly Transport Protocol for Continuous Media over Best Effort Networks", International Journal of Communication Systems, John Wiley and Sons, 2002, 15: 881-898.
- [5] Hala ElAarag, Andrew Moedinger, "IFTP-W: A TCP-Friendly Protocol for Multimedia Applications over Wireless Networks", ACM Southeast Conference, Vol 2, pp.36-40, March 2005.
- [6] Bin Zhou, Cheng Peng Fu, "An Enhancement of TCP Veno over Light-Load Wireless Networks", IEEE Communications Letters, Vol 10 No 6 June 2006
- [7] Cheng Peng Fu, Soung C. Liew, "TCP Veno: TCP Enhancements for Transmission Over Wireless Access Networks", IEEE Journal on Selected Areas in Communication, 21(2): 216-227, February 2003
- [8] Hala ElAarag, Andrew Moedinger, "Performance Evaluation of an Internet Friendly Transport Protocol over Networks with Lossy Links", Applied Telecommunication Symposium, Spring Simulation Multiconference, San Diego, California, pp. 21-25, April 2005
- [9] Wu E. and Chen M.Z., "JTCP: Jitter Based TCP for Heterogeneous Wireless Networks", IEEE Journal on Selected Areas in Communications, 22(4): 757-766, May 2004
- [10] Xu K. et al., "TCP-Jersey for Wireless IP Communications", IEEE Journal on Selected Areas in Communications, 22(4):747-756, May 2004.
- [11] Casetti et al., "TCP Westwood: End-to-end Bandwidth Estimation for Enhanced Transport over Wireless Links", Wireless Networks, 8:467-479, 2002
- [12] J. Widmer, R. Denda, and M. Mauve., "A survey on TCP-Friendly congestion control," IEEE Network, vol. 15, pp. 28-37, May 2001.
- [13] Kazumi Kaneko and Jiro Katto, "Reno Friendly TCP Westwood based on Router Buffer Estimation", Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS), 2005