TCP-NCL: A Unified Solution for TCP Packet Reordering and Random Loss

Chengdi Lai, Ka-Cheong Leung, and Victor O.K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China
E-mail: {laichengdi, kcleung, vli}@eee.hku.hk

***Abstract*—The problems of TCP packet reordering and random loss over wireless networks have motivated the development of a number of TCP variants. However, most of these variants focus on resolving only one of the aforementioned two problems. A few unified solutions for both problems generally extend beyond the scope of the transport layer. In this paper, we propose a new TCP variant, known as TCP for non-congestion loss (TCP-NCL), to tackle both problems under one compact framework. Different from previous unified solutions, the modifications are limited to sender-side TCP only, thereby facilitating possible future wide deployment. A retransmission decision timer and a congestion response decision timer have been installed to trigger packet retransmission and congestion response, respectively. Our simulation studies reveal that TCP-NCL is robust against packet reordering as well as random packet loss while maintaining responsiveness against situations with purely congestive loss.**

I. Introduction

Transmission Control Protocol (TCP) is the most important transport layer protocol over current networks, providing connection-oriented end-to-end in-order data delivery services to various applications. Traditionally, the design of TCP relies on the assumptions of nearly in-order packet delivery and error-free transmission. A TCP receiver would expect the sequence numbers of the received packets belonging to the same flow to be consecutively ordered. Otherwise, it will send back a duplicate acknowledgment (*dup-ack*) to its corresponding TCP sender for each received packet failing the expectation. At the sender side, when the number of dup-acks exceeds a certain threshold value, various congestion control measures will be activated and the size of the congestion window (*cwnd*) will be reduced. Therefore, out-of-order packet events are effectively treated as an indication of network overload.

However, while network congestion does constitute the dominant cause of out-of-order packet events over conventional wired networks, wireless networks have often seen random packet loss and packet reordering as other predominant sources of such events.

As compared with wired media, the wireless medium provides much more lossy physical links for data transmission. Signals propagating over wireless channels suffer severely from degradation, interference, and noise. Packets received may be damaged to an extent beyond the recovery capability of error correcting codes, if any. These packets are thus discarded, leading to random packet losses. In wireless networks, random packet losses are not rare events.

Packet reordering refers to the network behavior that the packet order of a TCP flow is disrupted. Despite conventional beliefs that packet reordering is a transient or pathological network behavior, it is in fact persistently observed over wireless networks and can be caused by link layer retransmissions (LLRTX) [1], [7] and path changes. LLRTX are mechanisms proposed to locally retransmit damaged packets at the link layer to combat high transmission error rates of wireless channels. As a side effect of local retransmission, the packet order of a flow is altered. A path change often occurs over mobile ad-hoc networks, where a TCP flow traverses through a number of wireless nodes. The transmission path of the flow may be altered when some nodes move. The round trip time (RTT) of the connection may change too. Consequently, it is possible that some packets transmitted after the path change arrive at the receiving end before those packets transmitted prior to the path change.

Therefore, in wireless networks, packet reordering and random packet loss, in addition to network congestion, form important causes for the occurrence of the out-of-order packet events. Due to the incapability of the traditional TCP designs to differentiate among these three causes, congestion control measures are often spuriously activated, keeping *cwnd* unnecessarily small, and resulting in under-utilization of available network resources.

A. Related Work

The problems of packet reordering and random packet loss necessitate the modification of TCP to adapt it to wireless networks. Currently, these TCP variants fall into three different categories according to the problems they aim to tackle, namely, the problem of packet reordering, random packet loss, and both.

The solutions for packet reordering include RR-TCP [11], TCP-DCR [2], TCP-DOOR [10], TCP-Eifel [9], and TCP-PR [3]. TCP-DOOR and TCP-Eifel focus on detecting spurious retransmission after activating packet retransmission and congestion response. They infer a path change upon each successful detection and recover *cwnd* accordingly. Yet, they are generally unable to recover *cwnd* unnecessarily reduced in the event of random packet loss. Furthermore, as shown by our simulation results, the performance of TCP-DOOR can be significantly undermined under persistent packet reordering.

978-1-4244-5213-4/09/ $26.00 ©2009 IEEE

By contrast, RR-TCP, TCP-DCR, and TCP-PR aim to prevent persistent packet reordering from spuriously activating congestion response by deferring packet retransmission and congestion response until the occurrence of packet loss can be accurately confirmed. However, these variants rely on LLRTX to provide an almost error-free transmission channel and assume the absence of random loss. Upon the detection of packet loss, congestion response, in addition to packet retransmission, will also be deemed necessary and activated. Our simulation results reveal that they will suffer severe performance deterioration when LLRTX is unavailable.

The solutions for random packet loss include TCP-Veno [6] and TCP-Westwood (TCP-W) [4], and they focus on differentiating between congestion loss and non-congestion loss while ignoring packet reordering. This limitation, however, restricts the applicability of TCP-Veno and TCP-W to networks with nearly in-order packet delivery. Particularly, their interoperability with LLRTX are undermined.

The unified solutions for both problems generally extend beyond the scope of the transport layer. ATCP [8] introduces an ATCP layer between TCP and IP. The new layer switches TCP among various pre-defined states in accordance with the network condition, trying to avoid spurious packet retransmission and congestion response.

B. Organization of the Paper

The focus of this work is to propose a unified solution for the problems of packet reordering and random packet loss, which can effectively differentiate among the occurrences of network congestion, packet reordering, and random packet loss so as to take appropriate actions accordingly. Different from the previous unified solutions, the modifications involved will be limited to the sender-side TCP only, thereby facilitating possible wide deployment in the future. Section II presents our proposed TCP variant, known as TCP for non-congestion loss, for dealing with situations when packet reordering and random loss coexist with congestive packet loss. Section III compares the performance of TCP-NCL with some popular TCP variants, namely, RR-TCP, TCP-DCR, TCP-DOOR, TCP-PR, TCP-Veno, and TCP-W. Section IV concludes and discusses some possible extensions of our work.

II. TCP-NCL

As discussed in Section I-A, packet retransmission and congestion response mechanisms are generally bundled together in the existing TCP variants for dealing with persistent packet reordering (RR-TCP, TCP-DCR, and TCP-PR). A major limitation in doing so is that any information inferred by the occurrences of the events which happen after packet retransmission is completely ignored, thereby increasing the possibility of spurious congestion control activation. Specifically, if an acknowledgment (ACK) for a packet lags behind the retransmission of that packet by a short duration, the congestion control measures generally bundled with the packet retransmission are rather unnecessary. The acknowledgment packet can either be for the originally transmitted packet or the retransmitted packet. The former case obviously rejects the possibility of a congestive loss. In the latter case, a fairly short round-trip time and thus a lightly loaded network can be inferred.

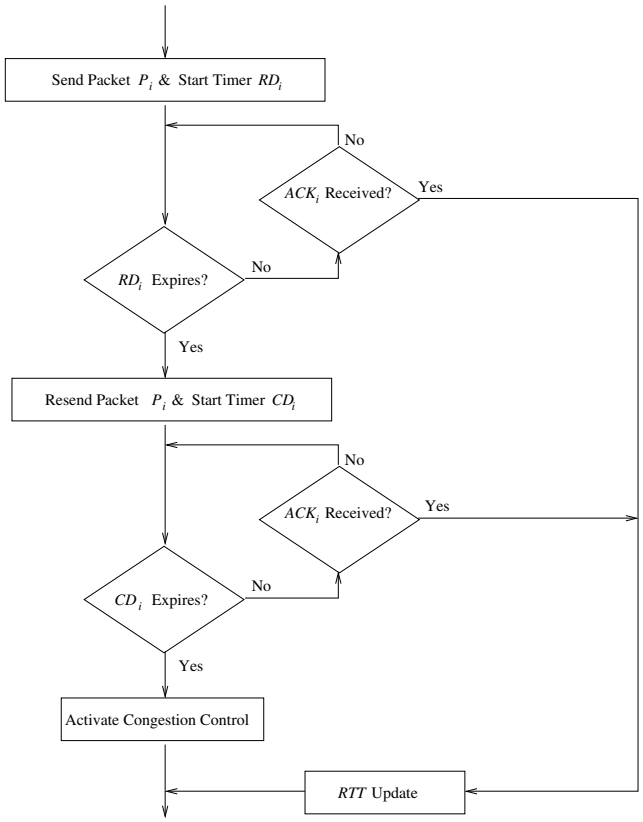

Fig. 1. Flowchart for TCP-NCL.

Based on the observation stated above, we decide to split packet retransmission and congestion control in our model, as illustrated in Fig. 1. A new retransmission decision timer RD_i is started whenever a new packet P_i is injected into the network. If ACK_i is received at the TCP sender before RD_i expires, RD_i is cancelled. Otherwise, P_i will be retransmitted and a congestion response decision timer CD_i will be started.

CD_i will be cancelled if ACK_i arrives before it expires. Otherwise, the congestion control mechanisms will be activated upon the expiration of CD_i. Therefore, the installation of CD_i allows the TCP sender extra time after the packet retransmission to decide whether congestion control shall be activated. ACK_i arriving before the expiration of CD_i shall be treated as an indication of the absence of network congestion. Thus, this eliminates the need for activating the congestion control measures.

The determination of the expiration periods of CD_i and RD_i (denoted as τ_{RD_i} and τ_{CD_i}, respectively) is a non-trivial task, requiring careful balancing of various factors involved. We propose to determine CD_i and RD_i based on the statistical distribution of round trip time (RTT). Sections II-B and II-C evaluate τ_{RD_i} and τ_{CD_i}, respectively. The complete TCP-NCL

framework is shown in Fig. 2.

Process	Description
RD_i	*Transmit* (P_i) *Start*RD_i (*maxRTT*)

CD_i	**if** *Status* (RD_i) = *EXPIRED*
	then {
	Retransmit (P_i)
	StartCD_i (*minRTT*)
	}

RTT–Update	**Procedure** *NewAck* (ACK_i) {
	// *called when a new ACK arrives*
	if *Status* (RD_i) = *PENDING*
	then {
	UpdateRTT (*RTTSample*)
	Cancel (RD_i)
	} **else if** *Status*(CD_i) = *PENDING*
	then {
	if *delay–after–rtx* < $\beta\,\tau_{CD_i}$
	then *UpdateRTT* (*RTTSample*)
	Cancel(CD_i)
	}}
	Procedure *UpdateRTT* (*RTTSample*) {
	RTTRecord.Add(*RTTSample*)
	While *Size*(*RTTRecord*) > *MRRL*
	Do *Discard* (*RTTSample*)
	}

Fig. 2. The complete TCP-NCL framework.

A. TCP-NCL-RTT-Update

The distribution of RTT is time-variant over wireless networks, where the network topology and/or loading may change over a TCP session. The time-varying property requires periodic updating of RTT samples, so that recent RTT samples are recorded and some outdated samples are discarded. For this purpose, we would first define the *maximum RTT record length* (MRRL). Whenever the total number of the stored RTT samples exceeds MRRL, the oldest samples will be discarded. Thus, in the long run, only the most recent MRRL RTT samples will be stored.

When sampling RTT, an ACK_i which arrives no later than time $\beta\,\tau_{CD_i}$ (where β is a system-designed parameter falling between zero and one) after the retransmission of P_i will be counted as an acknowledgment for the originally transmitted P_i, since it is impossible for the retransmitted P_i to be acknowledged within such a short interval. To this effect, we measure the time elapsed between the retransmission of P_i and the arrival of ACK_i, *delay-after-rtx*. If the measured value is less than $\beta\,\tau_{CD_i} (0 < \beta < 1)$, we will record an RTT sample by subtracting the time P_i is first transmitted from the time ACK_i arrives; otherwise, the corresponding RTT sample will be ignored.

B. Assignment of τ_{RD_i}

τ_{RD_i} determines how long we have to wait before activating a packet retransmission. An excessively small τ_{RD_i} will result in spurious packet retransmissions. Even though we have decoupled the decision of packet retransmission with that of congestion response, spurious retransmission at a significant level is nevertheless highly undesirable, as it injects duplicated bytes into the network and reduces the efficiency in the use of network resources. Nevertheless, τ_{RD_i} will also need to be upper bounded by the expiration period of the retransmission timer to avoid unnecessary retransmission timeout (RTO).

Thus, we set τ_{RD_i} as:

$$\tau_{RD_i} = max(RTT) \tag{1}$$

where $max(RTT)$ denotes the maximum RTT among all the stored samples. As will be shown by our simulation results, TCP-NCL demonstrates a very efficient usage of network bandwidth in that it almost always attains the highest connection goodput among all our compared TCP variants. This performance can hardly be achieved if spurious retransmission constitutes a significant wastage of total throughput. Thus, the choice of setting τ_{RD_i} as $max(RTT)$ seems to be appropriate.

C. Assignment of τ_{CD_i}

In Fig. 1, when ACK_i arrives at the sender during the period between the expiration of RD_i and CD_i, congestion control will not apply. This ACK_i can be due to the originally transmitted or the retransmitted packet P_i. Our intention here is for the timer CD_i to assume a sufficiently small expiration period so that either one of the following two cases will hold:

1) If the originally transmitted P_i triggers this ACK_i, there is no need to reduce *cwnd* or the slow start threshold (*ssthresh*) since no packet loss occurs.
2) If the retransmitted packet triggers the received ACK_i, we can infer that RTT is too short. This in turn indicates a very small chance of network overload. As for the originally transmitted packet, it can either be reordered or lost due to random channel error. In either event, it would be unnecessary to activate congestion control mechanisms.

Therefore, a TCP sender will decide that congestion loss has occurred and congestion control mechanisms should come into play only when CD_i has expired, and by then both packet reordering and random packet loss would be very unlikely.

Intuitively, we set τ_{CD_i} as:

$$\tau_{CD_i} = min(RTT) + \Delta T \tag{2}$$

where $min(RTT)$ denotes the minimum RTT among all the stored samples. Apparently, $min(RTT)$ qualifies as our prescribed "sufficiently small value", and the term ΔT is introduced for minor adjustment. By increasing ΔT, a TCP sender can detect spurious retransmission more effectively. In our actual implementation, because timestamps are utilized for realizing CD timers and a minor delay in activating congestion control will be inherently introduced, we set ΔT to zero to avoid an excessively aggressive TCP sender.

III. Performance Evaluation

In this section, we present our simulation results and compare TCP-NCL with some popular TCP variants, namely, RR-TCP, TCP-DCR, TCP-DOOR, TCP-PR, TCP-Veno, and TCP-W. Section III-A describes the simulation setup. Section III-B compares the performance of these TCP variants under various network configurations.

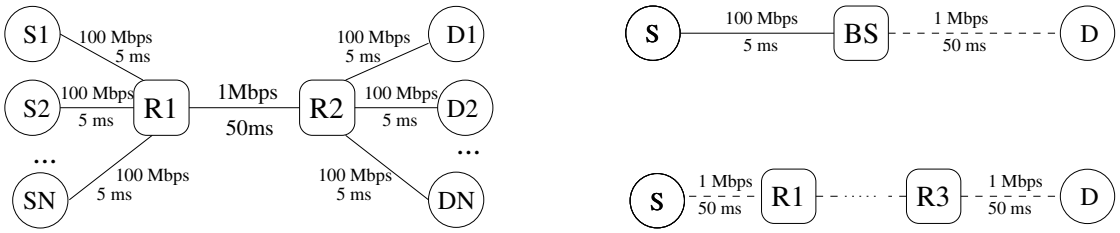

Fig. 3. The network topologies used for performance comparison: (a) Wired network with a dumbbell topology. (b) Top: infrastructure-based wireless network; bottom: multi-hop wireless network.

A. Simulation Setup

Three different network topologies, as illustrated in Fig. 3, have been used in the simulation experiment: a wired network with a dumbbell topology, an infrastructure-based wireless network, and a multi-hop wireless network. We aim to examine the performance of the TCP variants under congestive loss, random packet loss, and persistent reordering.

In the wired network with a dumbbell topology, multiple pairs of TCP-NCL senders and receivers share an error-free ordered bottleneck link. Thus, all occurrences of packet loss are due to network congestion. The number of sender-receiver pairs ranges from one to ten.

In the infrastructure-based wireless network, a TCP sender and a TCP receiver are connected through some wired and wireless links. Random channel error from zero to 15% is deliberately introduced into each wireless link. The link-layer retransmission mechanism is disabled to simulate an in-order channel for a TCP flow.

In the multi-hop wireless network, a TCP sender is connected to a TCP receiver via four wireless links. Random channel error is introduced into the wireless links with an error rate ranging from zero to 15%. The link-layer retransmission is enabled to introduce persistent packet reordering. Under high channel error rate, however, local link-layer retransmission cannot guarantee successful packet delivery due to the retransmission limit (set to three). Consequently, TCP will be confronted with both packet reordering and random packet loss.

All simulation experiments have been run over Network Simulator (ns) Version 2.29 [5]. The minimum RTO is set to one second. Delayed ACK is disabled so that ACK packets are sent by the TCP receiver upon the arrival of data packet. MRRL is set to 1000 and β is set to 0.8. In each test under the infrastructure-based wireless network and the multi-hop wireless network, a total of 20 runs have been performed to compute an average value and a 95% confidence interval of the performance metric, the connection goodput in megabit per second (Mbps).

B. Performance Comparison

The simulation results are shown in Fig. 4. In the dumbbell topology, there are a number of TCP flows sharing the bottleneck link R1-R2. The normalized goodput of each TCP flow, which is defined as the ratio of its goodput to the average goodput among all flows, is computed and plotted in Fig. 4(a). All the plotted normalized goodputs are approximately one unit. The total goodput of all TCP flows is plotted in Fig. 4(b) and it remains at a high level despite the increase in the number of flows. Thus, the TCP-NCL flows are able to share the bandwidth of a bottleneck link fairly and efficiently, demonstrating competent responsiveness with the presence of congestive loss only.

For the infrastructure-based wireless network as shown in Fig. 4(c), TCP-NCL essentially maintains a stable goodput level against the channel error rate from zero to 15%, whereas almost all of the other TCP variants experience drastic goodput decrease as the error rate increases. The only exception is TCP-W, which exhibits a relatively elegant performance deterioration. The performance of TCP-NCL should be attributable to its effectiveness in differentiating between congestion loss and random packet loss, a merit introduced by postponing the activation of congestion response until the congestion response decision timer expires. In contrast, RR-TCP, TCP-DCR, TCP-DOOR, and TCP-PR exclude the possibility of random packet loss, resulting in under-utilization of network resources.

For the multi-hop wireless network as shown in Fig. 4(d), TCP-DCR, TCP-NCL, and TCP-PR outperform other variants under the channel error rate less than 9%, thereby demonstrating robustness to persistent reordering. When the error rate further increases and random packet loss coexists with packet reordering, TCP-NCL performs slightly better than TCP-PR while the performance of TCP-DCR is substantially deteriorated. Again, in the latter scenario, the installation of the CD_i timer plays a crucial role in achieving the performance improvement for TCP-NCL.

IV. Conclusions

In this paper, we have adapted Transmission Control Protocol (TCP) to tackle packet reordering and random packet loss in wireless networks. The retransmission decision (RD) timer is installed to trigger a retransmission of a packet when its acknowledgment has not been received within a specified time period. The decisions on whether to activate congestion control measures are postponed until the expiration of the congestion response decision (CD) timer so that occurrences of false congestion response can be reduced. A TCP-NCL-RTT-Update process is introduced for maintaining the RTT samples in the storage.

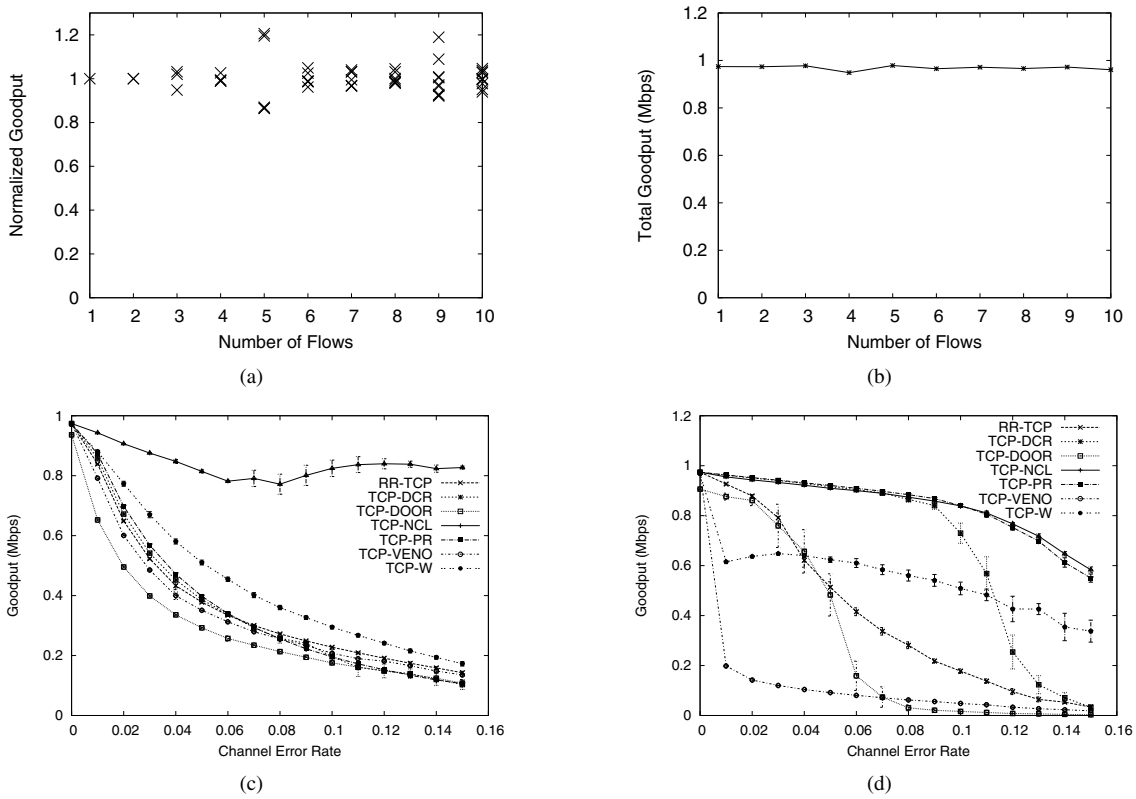

Fig. 4. Goodput performance under various topologies: (a) Wired network with a dumbbell topology: normalized goodput. (b) Wired network with a dumbbell topology: total goodput. (c) Infrastructure-based wireless network. (d) Multi-hop wireless network.

Various test cases have been designed to simulate the problems of congestive loss, packet reordering, and random packet loss. The performance of TCP-NCL under these cases have been examined and compared with other TCP variants. Our simulation results reveal that TCP-NCL offers the best performance in all these test cases. Further simulation experiments are currently in progress to investigate the internal behavior of TCP-NCL as well as other important performance issues, including TCP-friendliness.

There are several possible extensions to our work, some of which are listed below:

1) improve the computational complexity of TCP-NCL algorithms, especially that of TCP-NCL-RTT-Update;
2) develop a distribution model of RTT to make TCP-NCL a memoryless algorithm, and;
3) implement and examine the performance of TCP-NCL on experimental testbeds.

ACKNOWLEDGEMENTS

This research is supported in part by the University of Hong Kong Strategic Research Theme of Information Technology. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which assisted us in improving the quality of the paper.

REFERENCES

[1] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 756-769, December 1997.

[2] S. Bhandarkar and A.L.N. Reddy. TCP-DCR: Making TCP Robust to Non-Congestion Events. *Lecture Notes in Computer Science*, Vol. 3042, pp. 712-724, May 2004.

[3] S. Bohacek, J.P. Hespanha, J. Lee, C. Lim, and K. Obraczka. A New TCP for Persistent Packet Reordering. *IEEE/ACM Transactions on Networking*, Vol. 14, No. 2, pp. 369-382, April 2006.

[4] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang. TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks. *Wireless Networks*, Vol. 8, No. 5, pp. 467-479, 2002.

[5] K. Fall and K. Varadhan. The *ns* Manual (formerly *ns* Notes and Documentation). *The VINT Project*, 6 January 2009.

[6] C.P. Fu and S.C. Liew. TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 21, No. 2, pp. 216-228, February 2003.

[7] F. Hu and N. Sharma. Enhancing Wireless Internet Performance. *IEEE Communications Surveys and Tutorials*, Vol. 4, No. 1, pp. 2-15, December 2002.

[8] J. Liu and S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 7, pp. 1300-1315, July 2001.

[9] R. Ludwig and R.H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. *ACM SIGCOMM Computer Communication Review*, Vol. 30, Issue 1, pp. 30-36, January 2000.

[10] F. Wang and Y. Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-Of-Order Detection and Response. *Proceedings of ACM MOBIHOC 2002*, pp. 217-225, Lausanne, Switzerland, 9-11 June 2002.

[11] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A Reordering-Robust TCP with DSACK. *Proceedings of IEEE ICNP 2003*, pp. 95-106, Atlanta, GA, USA, 4-7 November 2003.