Title	Enhancing wireless TCP a serialized-timer approach
Author(s)	Lai, C; Leung, KC; Li, VOK
Citation	Proceedings - Ieee Infocom, 2010
Issued Date	2010
URL	http://hdl.handle.net/10722/99332
Rights	IEEE INFOCOM Proceedings. Copyright © IEEE, Computer Society.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

Enhancing Wireless TCP: A Serialized-Timer Approach

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*— In wireless networks, TCP performs unsatisfactorily since packet reordering and random losses may be falsely interpreted as congestive losses. This causes TCP to trigger fast retransmission and fast recovery spuriously, leading to under-utilization of available network resources. In this paper, we propose a novel TCP variant, known as TCP for non-congestive loss (TCP-NCL), to adapt TCP to wireless networks by using more reliable signals of packet loss and network overload for activating packet retransmission and congestion response, separately. TCP-NCL can thus serve as a unified solution for effective congestion control, sequencing control, and loss recovery. Different from the existing unified solutions, the modifications involved in the proposed variant are limited to sender-side TCP only, thereby facilitating possible future wide deployment. The two signals employed are the expirations of two serialized timers. A smart TCP sender model has been developed for optimizing the timer expiration periods. 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 in current networks, providing in-order data delivery services to various applications and featuring reliable transmission and *end-to-end* congestion control. In fulfilling the former feature, signals of packet loss have been used for triggering retransmission of lost data packets or segments to ensure the eventual delivery of every data byte. In fulfilling the latter, signals of network overload have been used for triggering congestion response, which reduces the size of the congestion window (*cwnd*) and thus avoids further overloading a congested network[1]. However, how to identify signals and utilize the identified signals to perform effective congestion control and loss recovery has long been a very challenging research problem.

The popular TCP variants, such as TCP Reno [1], [18], use the same set of signals for indicating packet loss and network overload. Two types of signals are used, namely retransmission timeout (RTO) and triple duplicate acknowledgments (ACKs). A retransmission timer is started when a data packet is first injected into the network, and will timeout if an ACK for the packet is still missing when the timer expires. Upon the occurrence of an RTO, all the outstanding packets will be retransmitted. At the same time, the network is deemed severely congested and *cwnd* will be forced to reopen from one packet size by employing the slow start algorithm.

A TCP receiver would expect all the data packets received to be consecutively ordered. Otherwise, it will send back a duplicate ACK to its corresponding TCP sender for each received packet failing the expectation. At the sender side, when the number of duplicate ACKs reaches a certain threshold value, say, three, fast retransmit and fast recovery will be activated, retransmitting the packet expected by the receiver and halving *cwnd*. Therefore, the arrival of triple duplicate ACKs, a direct signal of out-of-order packet events, is further used as an indication of congestive packet loss.

By using triple duplicate ACKs for activating packet retransmission and congestion response, the conventional TCP designs rely on the assumptions of a nearly in-order channel of negligible or recoverable transmission error. While the assumptions might hold over traditional wired networks, they are generally violated over wireless networks due to the significant level of occurrences of random packet losses and packet reordering [14].

As compared with the wired media, the wireless medium provides much more lossy physical links for data transmissions. Signals propagating over wireless channels suffer from degradation, interference, and noise. Packets received may be damaged beyond the recovery capability of error control codes, if any. These packets are thus discarded, leading to the occurrence of random packet losses.

Packet reordering refers to the disruption of the packet order of a TCP flow. Despite conventional beliefs that packet reordering is a transient or pathological network behavior, it is in fact persistently observed over modern networks and can be caused by a myriad of reasons [14], [15]. Due to the high transmission error rates and, in some cases, mobility in wireless networks, packet reordering is further increased substantially when the transmission medium evolves from physical cables to wireless. Specifically, four causes of packet reordering are commonly observed over wireless networks, including link layer retransmission (LLRTX) [2], [10], path change, hand-off in cellular wireless networks, and packet-level multi-path routing.

Hence, in wireless networks, triple duplicate ACKs amount

[1]Many popular TCP variants, including our proposed TCP-NCL in this paper, are designed based on the additive-increase-multiplicative-decrease (AIMD) algorithms.

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

to fairly poor signals of packet loss. The conventional TCP designs use the same signals for inferring packet loss and network overload. Thus, they tend to falsely trigger packet retransmission and reduce *cwnd* from time to time, injecting duplicated bytes into the network and keeping *cwnd* unnecessarily small. Consequently, the available network resources are wasted and under-utilized.

Our proposed novel TCP variant, known as TCP for non-congestive loss (TCP-NCL), employs more reliable signals of packet loss and network congestion over a general error-prone channel for activating packet retransmission and congestion response, respectively. Thus, TCP-NCL serves as a unified solution for effective congestion control, sequencing control, and loss recovery. Moreover, the proposed modifications involved are limited to sender-side TCP only, thereby facilitating possible future wide deployment. As summarized in [13], most existing TCP variants in the literature [3], [4], [5], [6], [8], [9], [11], [17], [19], [20], [21] only focus on congestion control and loss recovery with the presence of either random packet loss or packet reordering. A few unified solutions, such as [16], generally require information and modifications beyond the scope of the transport layer, hindering possible future wide deployment.

This paper is organized as follows. Section II develops the smart TCP sender (STS) model, explaining the motivation behind our serialized-timer structure and modelling the optimal timer expiration periods. Section III presents TCP-NCL based on the analytical results of the STS model. Section IV compares the performance of TCP-NCL with some popular TCP variants. Section V concludes and discusses some possible extensions of our work.

II. Smart TCP Sender Model

The structure of our smart TCP sender (STS) model, first motivated and described in [13], is 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 will be 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 no network congestion. Thus, this eliminates the need for activating any congestion control measures.

The expirations of two serialized timers (the second timer, CD_i, is started upon the expiration of the first timer, RD_i) are employed for triggering a packet retransmission and a congestion response, respectively. The latter action is thus delayed behind the former by the expiration period of CD_i. This brings us a two-fold advantage as compared with most existing TCP variants, which trigger these two actions concurrently. First, information conveyed by the events after a packet retransmission can be used for deciding whether or not to activate a congestion response. Specifically, if a packet is acknowledged shortly after its retransmission, this ACK is either due to the first transmitted packet or the retransmitted packet. The latter case implies a short round-trip time and thus a lightly loaded network. In either case, there is no need to trigger a congestion response.

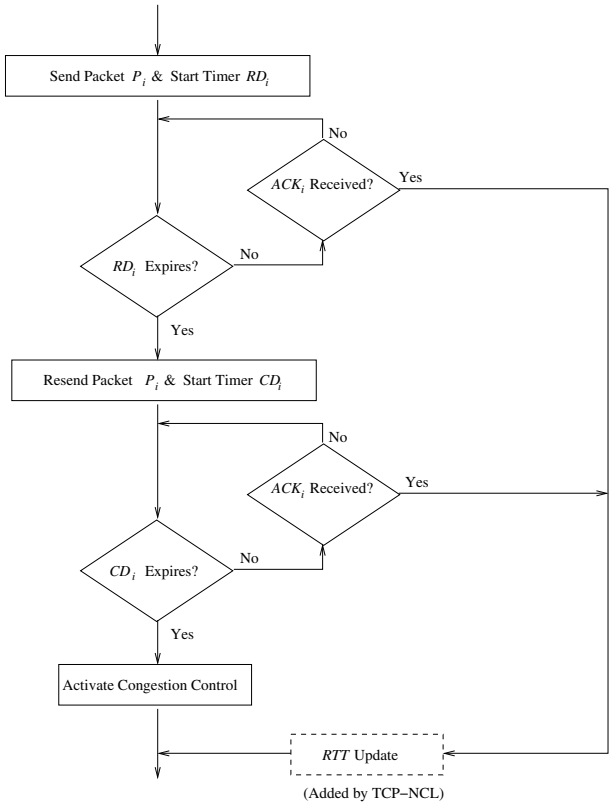


Fig. 1. A flowchart for the STS model.

Second, it is more "affordable" for TCP to trigger a false fast retransmit than a false congestion response. By advancing packet retransmission before the activation of congestion response, fast retransmit can be more promptly activated while avoiding the heavy penalty due to a false congestion response.

The expiration periods of CD_i and RD_i (denoted as τ_{RD_i} and τ_{CD_i}, respectively) are evaluated in Sections II-A and II-B, respectively. To facilitate the subsequent discussion, we summarize the notations in Table I.

A. Retransmission Decision Timer

τ_{RD_i} determines how long we have to wait before activating packet retransmission. The assignment of τ_{RD_i} requires carefully balancing between the objectives of prompt retransmission for preventing RTO and avoiding spurious packet retransmissions.

Ideally, it should be guaranteed that, when P_i^o is lost due to congestion, P_i^r should be promptly transmitted so that the remaining time before RTO is sufficient for it to be acknowledged. This ensures that *cwnd* will be halved instead

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

TABLE I
NOTATIONS.

$F(t)$	Cumulative distribution function of RTT for Packet P_i
G	Goodput of TCP sender
$mrtt_i$	Minimum attainable RTT for Packet P_i
P_i^o	First transmitted Packet P_i
P_i^r	Retransmitted Packet P_i
p_c	Congestive loss rate
p_l	Overall loss rate
p_w	Non-congestive loss rate
T	Throughput of TCP sender
W	Average *cwnd* over a TCP session
CL_i^o	P_i^o lost due to congestion
$PA_i(t)$	P_i acknowledged by time period t after RTX_i
PL_i^r	P_i^r lost
$PU_i(t)$	P_i unacknowledged by time period t after retransmission
RTX_i	Retransmission of P_i
TO	Retransmission timeout

of being reset to one at the onset of network overload. In other words, RTO will be incurred if and only if P_i^r is lost:

$$P(TO|CL_i^o) = P(PL_i^r) = p_l \tag{1}$$

On the other hand, an excessively small τ_{RD_i} will result in spurious packet retransmissions. Let $F(\tau_{RD_i}) = 1-\zeta$, where $0 \le \zeta \le 1$. It can be shown that:

$$\frac{G}{T} = [(1+p_l)+\zeta(1-p_l)]^{-1} \tag{2}$$

To improve efficiency, we thus have:

$$\zeta \ll 1 \tag{3}$$

Considering that retransmission timer only serves as a coarse upper bound for performing loss recovery and congestion control, it is possible for (1) and (3) to be simultaneously satisfied with τ_{RD_i} appropriately set.

B. Congestion Response Decision Timer

As discussed earlier, an ACK_i arriving before the expiration of CD_i shall reject the occurrence of a congestive loss for P_i^o. In other words, it should be guaranteed that:

$$P(CL_i^o|PA_i(t)) = 0 \tag{4}$$

where $0 \le t \le \tau_{CD_i}$. It can be shown to be equivalent to[2]:

$$\tau_{CD_i} \le \tau_u \tag{5}$$

where τ_u denotes a certain upper bound value and $\tau_u \ge mrtt_i$.

Therefore, we shall seek an optimal solution for τ_{CD_i} within $[0,\ \tau_u]$. Consider a time period t $(0 \le t \le \tau_u)$ after the retransmission of packet P_i, when a TCP sender is facing the decision of whether or not to activate a congestion response. The risk associated with a positive decision is that the network may not be congested (i.e. P_i^o is not dropped due to congestion). Consequently, the spuriously activated congestion response will reduce *cwnd* unnecessarily. On the other hand, if the network is indeed overloaded (i.e. P_i^o is lost due to congestion), congestion response should be promptly activated by halving *cwnd*. Otherwise, the network congestion may be exacerbated and an expensive RTO may be triggered instead, forcing *cwnd* to reopen from one packet size and introducing extra reduction in the TCP sender throughput.

To quantify the risk associated with activating a congestion response, the following metric is introduced. The expected cost of activating a congestion response, $EC_A(t)$, is defined as the product of the conditional probability of P_i^o not being lost due to congestion, given that P_i is unacknowledged, and the throughput reduction due to the activation of a congestion response, C_S. In other words,

$$EC_A(t) = P(\overline{CL_i^o}|PU_i(t)) \cdot C_S \tag{6}$$

Using the same token, the expected cost of delaying a congestion response, $EC_D(t)$, is introduced to quantify the risk associated with delaying congestion response. It is defined as the product of the conditional probability that a congestive loss and an RTO have occurred, given that P_i is unacknowledged, and the extra TCP sender throughput reduction due to RTO, C_T.

$$EC_D(t) = P(TO \cap CL_i^o|PU_i(t)) \cdot C_T \tag{7}$$

It can be shown that:

$$\begin{aligned} EC_A(t) &= \frac{p_w[1-(1-p_l)F(t)]}{2\{p_c+p_w[1-(1-p_l)F(t)]\}} \\ &\quad \times \lfloor 0.5\,W \rfloor (\lfloor 0.5W \rfloor + 1) \end{aligned} \tag{8}$$

$$\begin{aligned} EC_D(t) &= \frac{p_c\,p_l}{p_c+p_w[1-(1-p_l)F(t)]} \\ &\quad \times (\lceil log_2\,0.5W \rceil\, W - 2^{\lceil log_2\,0.5W \rceil} + 1) \end{aligned} \tag{9}$$

where $0 \le t \le \tau_u$.

Observe that:

$$\frac{\partial}{\partial t} EC_A(t) \le 0, \quad \frac{\partial}{\partial t} EC_D(t) \ge 0 \tag{10}$$

Thus, when the activation of congestion response is postponed further, $EC_D(t)$, which quantifies the risk associated with delaying a congestion response, increases, while $EC_A(t)$, which quantifies the risk associated with activating a congestion response, drops. When $EC_A(t)$ is greater than $EC_D(t)$, it is advantageous to set τ_{CD_i} no less than t since the operation cost of activating congestion response is $EC_A(t)$, which is larger than that of deferring it, $EC_D(t)$. The cost may drop when t increases. Similarly, when $EC_D(t)$ is greater than $EC_A(t)$, it is advantageous to set τ_{CD_i} no greater than t since the operation cost of deferring congestion response is $EC_D(t)$, which is larger than that of activating it, $EC_A(t)$. The cost may drop when t decreases.

Therefore, the evaluation of the optimal solution of τ_{CD_i}, $\tau^*_{CD_i}$, subject to the constraint $0 \le \tau_{CD_i} \le \tau_u$, is given by:

$$\tau^*_{CD_i} = \begin{cases} 0, & p_c > (p_l \cdot \frac{C_T}{C_S} + 1)^{-1}\, p_l \\ \tau_u, & p_c < \{\frac{p_l\, C_T}{[1-(1-p_l)\cdot F(\tau_u)]\cdot C_S} + 1\}^{-1} \cdot p_l \\ \tau_{TH}, & \text{otherwise.} \end{cases} \tag{11}$$

where $\tau_{TH} = F^{-1}\left(\frac{1}{1-p_l} - \frac{C_T\, p_c}{C_S \cdot (p_l - p_c)} \cdot \frac{p_l}{1-p_l}\right)$.

[2]The derivation of the model has been abridged due to the constraints in space. Interested readers can refer to [12] for details.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

III. TCP-NCL

The STS model is a limited, idealized TCP sender. First, it does not provide a closed-form expression for τ_{RD_i}. Second, it assumes, among other things, prior knowledge of the RTT distribution $F(t)$, which is impractical. Therefore, we propose TCP-NCL to closely approximate SLS.

The serialized-timer structure of the SLS model is supplemented by TCP-NCL with an NCL-RTT-Update (NRU) process for maintaining a set of RTT samples so as to estimate $F(t)$. 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 of $F(t)$ requires periodic updating of RTT samples. We would thus define the *maximum RTT record length* as *MRRL*. Only the most recent *MRRL* RTT samples will be stored in *rttRcd* while all older samples will be discarded.

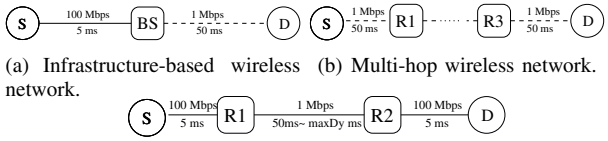
The NRU process is invoked if ACK_i arrives no later than time $\beta\tau_{CD_i}$ (where β is a system-designed parameter such that $0 < \beta < 1$) after the transmission of P_i will be considered as an ACK for P_i^o, since P_i^r cannot be acknowledged within such a short interval.

τ_{RD_i} is set to *max(rttRcd)*, which denotes the maximum sampled RTT stored in *rttRcd*. Intuitively, this assignment is an appropriate choice, keeping spurious retransmission at a minimal level without excessively delaying fast retransmit. By (11), τ_{CD_i} is set to *min(rttRcd)*, which corresponds to the minimum RTT sample stored in *rttRcd*. It can be shown that $p_c < (p_l \frac{C_T}{C_S} + 1)^{-1} p_l$ when $p_c < p_l \ll 1$. Thus, it is reasonable to assume $\tau^*_{CD_i}$ to be τ_u or τ_{TH}, depending on the value of p_c. A conservative estimation for both τ_u and τ_{TH} can be set as $mrtt_i$, which in turn leads to our choice of setting τ_{CD_i} as *min(rttRcd)*. A more adaptive approach capable of varying τ_{CD_i} based on the value of p_c is left as future work. The major foreseeable merit of such an approach is the increased responsiveness to network congestion. Nevertheless, our simulation results reveal that competing TCP-NCL flows are capable of maintaining inter-flow fairness, thereby demonstrating responsiveness against congestive loss.

IV. Performance Evaluation

In this section, we present our simulation results[3] and compare TCP-NCL with some popular TCP variants, namely, RR-TCP [21], TCP-DCR [3], TCP-DOOR [19], TCP-PR [5], TCP-Veno [9], and TCP-W [6]. All the simulation experiments have been performed using Network Simulator (ns) 2.29 [7]. Unless otherwise specified, MRRL and β are set to 1000 and 0.8, respectively, in the experiments.

Three different network topologies, as illustrated in Fig. 2, have been used for performance comparison: an infrastructure-based wireless network, a multi-hop wireless network, and a wired network with a bottleneck link. We examine the performance of all TCP variants under study with random packet losses, persistent packet reordering, and abrupt variations in RTT, respectively.



(a) Infrastructure-based wireless network. (b) Multi-hop wireless network.

(c) Wired network with a bottleneck link.

Fig. 2. The network topologies used for performance comparison.

In the infrastructure-based wireless network, a TCP sender and a TCP receiver are connected through some wired and wireless links. Random packet errors from 0 to 15% are deliberately introduced into each wireless link. The link layer retransmission mechanism is disabled to simulate an orderly 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 a random packet error rate ranging from 0 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 predetermined retransmission limit (set to three in this case). Consequently, TCP will be confronted with both packet reordering and random packet loss.

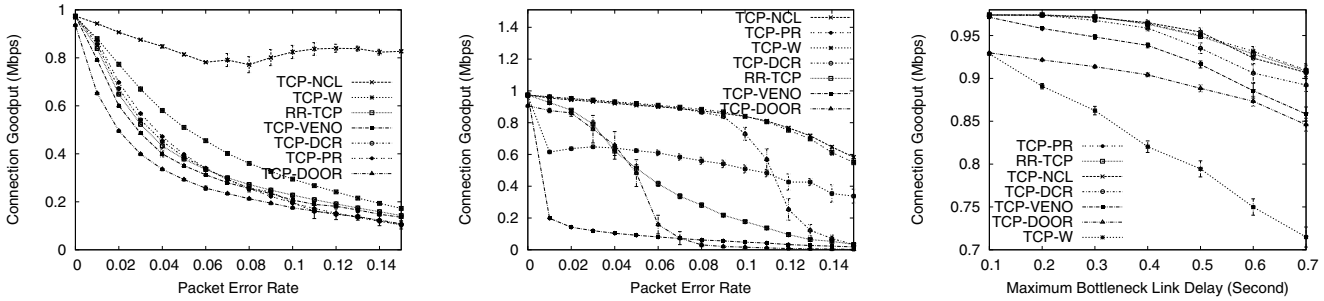
In the wired network, the TCP connection traverses a bottleneck link, and the delay takes on a random value in the interval $[50, maxDy]$ ms and is changed every 20 seconds. $maxDy$ ranges from 100 ms to 700 ms. Thus, RTT will vary abruptly yet infrequently. We aim to test the robustness of the NRU process with this configuration.

In each test over these three topologies, a total of 20 runs, each lasting 2000 seconds and using different seeds for generating the packet error or link delay, have been performed to compute an average value and a 95% confidence interval of TCP goodput in Mbps. In order to remove the effect of the transient states, only the statistics in the last 1000 seconds in each run are collected for computing the goodput.

The simulation results are shown in Fig. 3. In the infrastructure-based wireless network, TCP-NCL essentially maintains a stable goodput level against packet error rates from 0 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. 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.

In the multi-hop wireless network, TCP-DCR, TCP-NCL, and TCP-PR outperform other variants under packet error rate less than 9%, thereby demonstrating robustness to persistent reordering. When the error rate further increases and random losses are no longer transparent to the transport layer, TCP-NCL performs slightly better than TCP-PR while the perfor-

[3]The simulation results have been abridged due to the constraints in space. Interested readers can refer to [12] for details.



(a) Infrastructure-based wireless network. (b) Multi-hop wireless network. (c) Wired network with a bottleneck link.

Fig. 3. Goodput performance under various topologies.

mance of TCP-DCR is seriously deteriorated. In the latter scenario, the installation of the CD timers again helps TCP-NCL increase the reliability for signals of congestive loss.

In the wired network topology simulating abrupt variations in RTT, RR-TCP, TCP-NCL, and TCP-PR offer the best connection goodputs. Thus, NRU is partially verified as a robust RTT sampling process mechanism against abrupt increases in RTT. Otherwise, TCP-NCL should tend to set its timers too conservatively and probably trigger false fast retransmission and congestion response. As a matter of fact, TCP-NCL has been observed to attain even better performance in this scenario when MRRL is changed from 1000 to 200. This facilitates quicker removal of the outdated RTT samples.

V. Conclusions

In this paper, we have proposed a novel TCP variant, known as TCP-NCL, as a unified solution for performing loss recovery, sequencing control, and congestion control over general error-prone channels. In particular, we propose the use of two serialized timers for obtaining more reliable signals for packet loss and network overload separately. The STS model has been constructed based on the concept of expected cost and analytical expressions are derived as references for setting the timer expiration periods. We note that the timers are mostly determined intuitively in existing work. Our simulation investigations reveal that TCP-NCL offers significant performance improvement over various wireless/wired network scenarios.

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

1) implement a closer approximation of the STS model capable of varying τ_{CD_i} based on the value of p_c, as described in Section III;
2) develop a distribution model for RTT so as 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.

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. *IETF RFC 2581*, Apr. 1999.
[2] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Trans. Networking*, Vol. 5, No. 6, pp. 756-769, Dec. 1997.
[3] 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.
[4] E. Blanton and M. Allman. On Making TCP More Robust to Packet Reordering. *ACM SIGCOMM Computer Comm. Rev.*, Vol. 32, Issue 1, pp. 20-30, Jan. 2002.
[5] S. Bohacek, J.P. Hespanha, J. Lee, C. Lim, and K. Obraczka. A New TCP for Persistent Packet Reordering. *IEEE/ACM Trans. Networking*, Vol. 14, No. 2, pp. 369-382, Apr. 2006.
[6] 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.
[7] K. Fall and K. Varadhan. The *ns* Manual (formerly *ns* Notes and Documentation). *The VINT Project*, 6 January 2009.
[8] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An Extension to the Selective Acknowledgment (SACK) Option for TCP. *IETF RFC 2883*, July 2000.
[9] C.P. Fu and S.C. Liew. TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks. *IEEE J. Selected Areas in Comm.*, Vol. 21, No. 2, pp. 216-228, Feb. 2003.
[10] F. Hu and N. Sharma. Enhancing Wireless Internet Performance. *IEEE Comm. Surveys and Tutorials*, Vol. 4, No. 1, pp. 2-15, Dec. 2002.
[11] A. Lahanas and V. Tsaoussidis. Improving TCP Performance over Networks with Wireless Components using 'Probing Devices'. *Int'l. J. Commun. Systems*, Vol. 15, No. 6, pp. 495-511, July/Aug. 2002.
[12] C. Lai, K.-C. Leung, and V.O.K. Li. Enhancing Wireless TCP: A Serialized-Timer Approach. *Technical Report, TR-2009-006, Dept. of Electrical and Electronic Engineering, Univ. of Hong Kong*, Sep. 2009.
[13] C. Lai, K.-C. Leung, and V.O.K. Li. TCP-NCL: A Unified Solution for TCP Packet Reordering and Random Loss. *Proceedings of the IEEE PIMRC 2009*, Sep. 2009.
[14] K.-C. Leung and V.O.K. Li. Transmission Control Protocol (TCP) in Wireless Networks: Issues, Approaches, and Challenges. *IEEE Comm. Surveys and Tutorials*, Vol. 8, No. 4, pp. 64-79, Fourth Quarter 2006.
[15] K.-C. Leung, V.O.K. Li, and D. Yang. An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges. *IEEE Trans. on Parallel and Distributed Systems*, Vol. 18, No. 4, pp. 522-535, Apr. 2007.
[16] J. Liu and S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. *IEEE J. Selected Areas in Comm.*, Vol. 19, No. 7, pp. 1300-1315, July 2001.
[17] R. Ludwig and R.H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. *ACM SIGCOMM Computer Comm. Rev.*, Vol. 30, Issue 1, pp. 30-36, Jan. 2000.
[18] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. *IETF RFC 2001*, Jan. 1997.
[19] 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, June 2002.
[20] E. H.-K. Wu and M.-Z. Chen. JTCP: Jitter-Based TCP for Heterogeneous Wireless Network. *IEEE J. Selected Areas Comm.*, Vol. 22, No. 4, pp. 757-766, May 2004.
[21] 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, Nov. 2003.