How Bad TCP Can Perform In Mobile Ad Hoc Networks

Zhenghua Fu, Xiaoqiao Meng, Songwu Lu
UCLA Computer Science Department, Los Angeles, CA 90095
E-mails: {zfu,xqmeng, slu}@cs.ucla.edu

Abstract

Several recent studies have indicated that TCP performance degrades significantly in mobile ad hoc networks. This paper examines how bad TCP may perform in such networks and provides a quantitative characterization of this performance gap. Previous approach typically makes comparisons by ignoring the inherent dynamics such as mobility, channel error, and shared-channel contention. Our work provides a realistic, achievable TCP throughput upper bound, and may serve as a benchmark for the future TCP modifications in ad hoc networks. Our simulation findings indicate that node mobility, especially mobility-induced network disconnection and reconnection events, has the most significant impact on TCP performance. TCP NewReno merely achieves about 10% of a reference TCP's throughput in such cases. As mobility increases, the relative throughput drop ranges from almost 0% in static case to 1000% in highly mobile scenario (mobility speed is 20m/sec). In contrast, congestion and mild channel error (say, 1%) have less visible effect on TCP (with less than 10% performance drop compared with the reference TCP).

1 Introduction

The proliferation of mobile computing devices such as notebook computers and PDAs has spurred growing interests in the use of mobile ad hoc networks. In an ad hoc wireless network, users can communicate with each other or even access the Internet over the multihop, shared wireless channel using a transport protocol. Since TCP has been the *de facto* standard protocol used in the Internet, it is very likely that TCP will also find its application in mobile ad hoc networks to ensure reliable packet delivery and provide efficient bandwidth utilization.

Several recent studies have examined TCP performance in mobile ad hoc networks. Gerla et al. [6] studied the impact of wireless MAC protocols such as CSMA, FAMA and MACAW on TCP. Their study showed that TCP throughput will decrease significantly over multihop wireless links that implemented these MAC protocols. Holland et al. [1] studied the effects of mobility-induced link breakage upon TCP throughput. Their results showed that TCP throughput drops significantly when nodes move, due to TCP's inability to recognize the difference between link failure and congestion. They further suggest a technique of explicit link failure notification (ELFN) to improve TCP performance. In ELFN, the TCP sender is notified upon mobility-induced link failure events and subsequently does not invoke its congestion control actions. A more recent study [5] examines the limitations of ELFN proposal. They showed that the introduction of ELFN may decrease TCP throughput by as much as 5% in static topologies, since link failure may happen due to MAC-layer contentions. Other studies ([7],[8]) seek to differentiate packet losses between congestion and route failure to further improve TCP performance.

All these recent studies showed that TCP performance degraded significantly in mobile ad hoc networks. What is missing in TCP research is a better quantitative characterization of this performance degradation. If TCP has to operate within the rich dynamics and harsh constraints imposed by ad hoc networks, which include mobility, channel errors, and link-layer contentions in the multihop, shared wireless channel, it is not correct to expect that TCP performs as well as in static, wired or wireless networks. The authors of [1] defined a metric *expected throughput* to gauge the impact of route changes on TCP. The expected throughput is defined as TCP throughput simulated over a *static* (fixed) network of a linear chain topology having the same number of hops as in the mobile scenario. This approach, however, fall short of realistically considering the dynamics of mobility, channel error and congestions, the key impacts of the multihop ad hoc network that makes TCP perform different from the traditional wire-line networks.

This paper seeks to quantitatively characterize the performance degradation of TCP under these dynamics of the mobile ad hoc networks. To this end, we design and implement a reference TCP (R-TCP) protocol, in the event-driven network simulator *ns-2*. R-TCP uses perfect information on packet loss, collected from the simulator traces, to help the TCP sender respond appropriately. Therefore, R-TCP does not ignore the dynamics induced by mobility, shared channel contention and channel errors. Instead, it seeks to re-

Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)
1530-1346/02 $17.00 © 2002 IEEE

act adaptively upon each event. This way, our performance comparison of TCP and R-TCP gives a more realistic approximation. Furthermore, it serves as a benchmark to evaluate the effectiveness of future TCP modifications over ad hoc networks.

Our main simulation findings can be summarized as follows. We observe that node mobility, especially mobility-induced network disconnection and reconnection, has the most significant impact on TCP performance. In contrast, congestion and mild channel error has less visible effect on TCP. Other factors such as routing protocols, receiver and router buffer size, average hop number may lead to 10%$\sim$700% throughput decrease for TCP.

The rest of the paper is organized as follows: Section 2 describes the design ad implementation of the R-TCP protocol. Section 3 applies R-TCP in various simulation scenarios and compares it to the performance of traditional TCP. An analysis is also presented. Section 4 concludes this paper.

2 Design and Implementation of R-TCP using *NS-2*

Design of R-TCP We start our evaluation by designing a reference TCP (R-TCP) protocol for ad hoc networks in *ns-2* simulator. Two main components of R-TCP are as follows. (1) A global monitor (GM) is available at both the R-TCP sender and the receiver side. This global monitor has perfect knowledge of each packet's loss behavior by extracting information from the lower-layer traces of *ns-2*. (2) When packet loss occurs, the reaction of R-TCP sender differs from TCP NewReno according to network conditions. During normal transmission state when no packet loss happens, R-TCP behaves identical to TCP NewReno. When timeout is triggered or three duplicate ACKs are received at the sender, the sender will collect packet loss information from the global monitor and reacts accordingly.

Obviously, the assumption of real-time, perfect knowledge of real packet loss causes is not realistic and not implementable in real hardware. However, recent studies have shown that *ns-2* simulation results match fairly well with real hardware experimental measurements in ad hoc networking scenarios [9]. Our R-TCP simply assumes that the sender is intelligent enough to make a correct decision for each packet loss. Other than this, R-TCP does not make any additional modifications within the ad hoc networking infrastructures. Thus, we believe that the performance comparison between R-TCP and standard TCP protocols is meaningful.

R-TCP can also be used to analyze the performance degradation due to each component in ad hoc networks. These components can be radio channel contention, node mobility, channel error and etc. This detailed breakdown of performance penalty may shed new lights on TCP design over ad hoc networks.

```
UPON 3rd DUPLICATE ACK:

reason = get_reason();
  //invoke the global monitor to investigate
if(reason & Congestion)
  /* congestion happens,
     do what TCP NewReno does */
  slowdown();
  reset_rtx_timer(1,0);// don't backoff
  output(last_ack_+1, TCP_REASON_DUPACK);
else  // no congestion
  if(reason & Channel_Error)
    /* channel error happens, transmit the
       lost packet without slow down*/
    reset_rtx_timer(1,0);
    output(last_ack_ + 1, TCP_REASON_DUPACK);
  else if(reason & Route_Change)
     /* new path established,
        initiate a slow-start */
    slowdown();
    reset_rtx_timer(1,0);  //don't backoff
    output(last_ack_ + 1, TCP_REASON_DUPACK);

UPON RETRANSMISSION TIMEOUT:

reason = get_reason();
if(reason & Congestion)
  set CONGESTION flag;
  slowstart();
  reset_rtx_timer(0,1); //exponentially backoff
  send_much(0, TCP_REASON_TIMEOUT, maxburst_);
else if((reason & Disconnection) ||
        (reason & Channel_Error) )
  /* prolonged disconnection or blackout
     period, enter probing state */
  set PROBING_STATE on;
  rtx_timer_.reschedule((t_srtt_ >>
                    T_SRTT_BITS)*tcp_tick_);
  rtt_active_ = 0;
  probing();
else if(reason & Route_Change)
   //route change
    slowstart();
    reset_rtx_timer(1,0);
    output(last_ack_ + 1, TCP_REASON_DUPACK);
```

Figure 1. Pseudo Code of R-TCP

Implementation of R-TCP in *ns-2 simulator* In order to log each packet loss event within the simulation, we implemented a global monitor in *ns-2*. Every time when the sender triggers a timeout event or receives three duplicate ACKs, it queries the global monitor and finds out the real cause for the packet loss. Typically, packet loss can be incurred by one of the following four synthesized reasons:

- Congestion. Congestion is triggered by queue overflow at an intermediate router or an interface queue. In

ns-2, this can be done whenever an *"IFQ"* (interface queue packet discard) event is detected in the event logs of the global monitor.

- Disconnection. Disconnection is detected when two communicating nodes are out of their communication range. In *ns-2*, this is done by computing the physical distance of two nodes via tracking the physical location of each wireless node in the network.
- Channel error. The occurrence of channel error is detected from the discarded packets marked as *"ERR"*.
- Route change. Route change may trigger out-of-order packet delivery and invoke duplicate ACKs. It can be detected through one of two sequence events: (a) *"RET"* and *"NRTE"*, indicating MAC-layer maximum retransmission time is exceeded, or (b) *"ARP"* and *"NRTE"*, indicating no route to destination available.

After extracting real packet loss cause from the global monitor, R-TCP designs corresponding actions for each loss behavior. In this part, we claim no special wisdom to design intelligent response actions. We take the same action as TCP when a congestion event is triggered. If channel error occurs, R-TCP retransmits the lost packet but does not reduce its current congestion window size. When a route change event is detected, indicating the TCP flow has taken a new path, R-TCP restarts from slow start phase in congestion control. During disconnection period, R-TCP freezes TCP congestion control and starts to actively probe the network to learn when reconnection happens. Figure 1 is the pseudo code of R-TCP.

In some sense, R-TCP takes a conservative approach in its response design. For example, when a route change occurs, R-TCP does not continue with its current window size; instead, it goes back to slow-start along the new path. This is clearly quite conservative.

3 Performance Evaluation

3.1 Basic Simulation Settings

Our basic simulation settings consist of 30 nodes, located in a $400m \times 800m$ area. The effective communication range for each node is 250 meters. The average hop number from the sender to the receiver is about $6 \sim 7$. These parameters are intended to reflect reality and to be consistent with the IETF drafts. For both R-TCP and TCP NewReno, the maximum receiver window size is 8 packets, and each packet size is 1460 bytes. By default, we use IEEE 802.11 DCF mode and DSR as the MAC layer protocol and routing layer protocol respectively. Each wireless node has 50 packet buffer space and its raw radio link capacity is 2M bps. Each simulation lasts for 300 seconds. The average throughput is computed based on the maximum packet sequence number received during the whole TCP session. In all the figures, we use solid line to represent TCP NewReno and dotted line to represent R-TCP.

3.2 Impact of Congestion

We first characterize the performance in a static, clean-channel ad hoc network with some nodes being the bottlenecks. Specifically, we use 2 UDP flows with rate 200Kbps. Both these UDP flows only run within the time intervals $[50, 250]$ and $[100, 200]$, and they go across the same congested node as the TCP flow. Figure 2 (Left) shows that R-TCP performs very closely to TCP NewReno in the presence of congestion only. This means that if nodes are static, without channel error, TCP NewReno does not have significant performance degradation compared to R-TCP. From the traffic trace, we observe that up to 5% performance degradation is due to unnecessary retransmissions at the sender side. When intense MAC layer contention at certain nodes makes RTT extremely long, the sender will experience retransmission timeout.

3.3 Impact of Mobility

We now let all nodes in the topology move continuously at a speed of $5m/s$ except for the TCP sender and receiver. In order to keep the average hop number from the sender to the receiver unchanged, we fix the sender and receiver at the left-upper and right-lower corner of the topology respectively, then we let all the other nodes roam arbitrarily. The results are depicted in Figure 2 (Middle). It clearly shows that TCP NewReno suffers tremendously in this case. Its throughput is only about 1/7 of that achieved by R-TCP. This phenomenon is due to the fact that TCP NewReno cannot differentiate packet loss incurred by mobility from packet loss incurred by loss congestion. Whenever packet loss happens, it just double the retransmission timeout(RTO), even upon temporary disruption of connectivity along the delivery path. From the trace file, we found a sequence of route change events happening at the 50 second (see Figure 2 middle). This has caused TCP NewReno to experience timeouts repeatedly. As TCP NewReno exponentially enlarges its RTO after each timeout, the wasted time which could have been used for packet transmission also grows significantly. In an unfortunate situation depicted by Figure 2, the TCP NewReno could hardly recover from transient connectivity interruptions; compared to R-TCP, TCP NewReno only utilizes one over seventh of the total available bandwidth in this case.

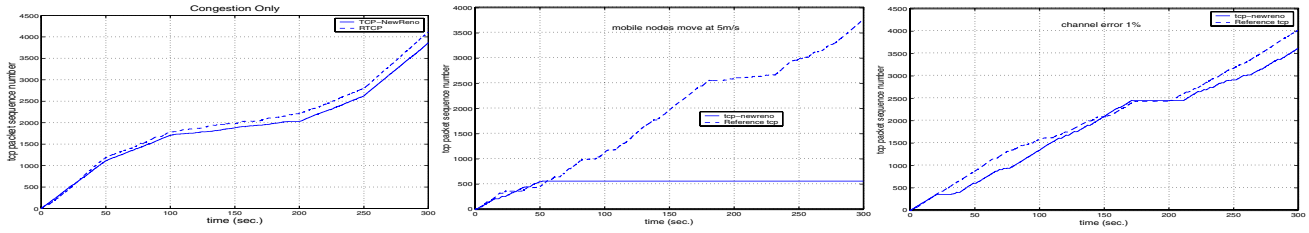

Figure 2. *Comparason of R-TCP and NewReno under different conditions* Left: Congestion, Middle: Nodes move with 5m/s, Right: Channel Error with 1%.

3.4 Impact of Channel Error

Continuing from the scenario of Figure 2 left, where we have two UDP flows running during the periods [50, 250] and [100,200] respectively and we keep all the nodes stationary, we further introduce channel error to the simulation. The channel error process is modeled by a two-state Markov chain with an average error rate of 1%. Surprisingly, here the result in Figure 2 (Right) shows that TCP NewReno performs quite comparably with R-TCP. TCP NewReno's throughput is only 10% less than R-TCP at the end of the simulation. Our explanation is that since we are using IEEE 802.11 MAC protocol, most of the random channel errors have been masked by the backoff and retransmission mechanism of this underlying protocol. However, in the presence of a long bursty channel error, packet loss can still occur. This is exemplified by the blackout period during $[170, 200]$ in Figure 2 (Right), where R-TCP is able to recover faster than TCP NewReno for its probing mechanism.

3.5 Composite Simulation Set

In previous section, we describe the performance degradation of TCP NewReno in ad hoc networks with specific settings of congestion, mobility and channel errors. Now we present a more complete picture on this performance degradation as depicted by Figures 3 to 4. In each of these figures, the x axis represents the mobility speed; the y axis represents the throughput measured in bit per second (bps). To smooth random effects introduced by specific topology and mobility pattern, for each set of simulation parameters, we repeat the experiment for 20 times and take the average. The plots presented in the following is based on such average values.

Mobility From the 6 plots of Figure 3 and 4, we observe that, as a general trend, the performance of both R-TCP and TCP NewReno decreases when the mobility speed is increased. However, the performance decrease in TCP NewReno is much more dramatic compared with R-TCP. From Figure 3 (Left), at mobility speed of 10 m/s, NewReno's throughput is 10K bps, only 10% of its throughput (95K bps) achieved in static case. Given various mobility speeds in Figure 3 (Left), we show that NewReno consistently performs worse than R-TCP. The improvement potential for NewReno is maximized when mobility speed goes from Low (around 2m/s) to Medium (around 10m/s), there NewReno throughput can be improved as much as 4 times according to our R-TCP measurement. Such performance gap mainly results from NewReno's inability to handle different packet loss situations efficiently, and being unaware of the impact of mobility in the underlying network.

The improvement potential tends to decrease as the speed goes from Medium to High (around 20m/s).

When the mobility speed is high, the performance of both R-TCP and TCP NewReno becomes constrained by the underlying routing protocols. However, since NewReno's slow response to topology change, it always wastes time on backing off when topology changes and effective communication is possible. As shown by Figure 3 (Left), the absolute performance gap at high mobility speeds is about 12K bps and R-TCP performs 1000% better than TCP NewReno.

Congestion In the set of simulations carried in Figure 3 (Middle), we introduce UDP traffic as before to create congestion in certain intermediate nodes. We observe that between mobility speeds 1 m/s and 10 m/s, R-TCP performs up to 300% better than NewReno with absolute throughput gap of 20K bps at low and medium mobility speed. While at higher mobility speed, the absolute performance gap decreases to 8K bps with a 500% performance gap.

Comparing Figures 3 (Left) with Figures 3 (Middle) , we observe that the existence of a congested node reduces the performance gap between TCP NewReno and R-TCP. Similar observation can be obtained by comparing Figures 3 (Right) with Figures 4 (Middle).

Channel Error From Figure 3 (Left) and (Right), we again confirm that channel error at low level(less than 1%) can be masked by lower protocols and does not affect TCP performance significantly. However, for channel error at

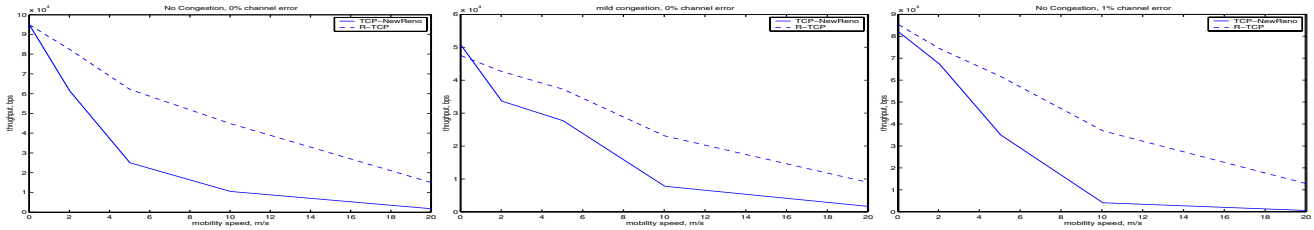

Figure 3. *Comparison of TCP NewReno and R-TCP under more complex conditions.* Left: Mobility Only, Middle: Mobility + Congestion, Right: Mobility + Channel Error 1%

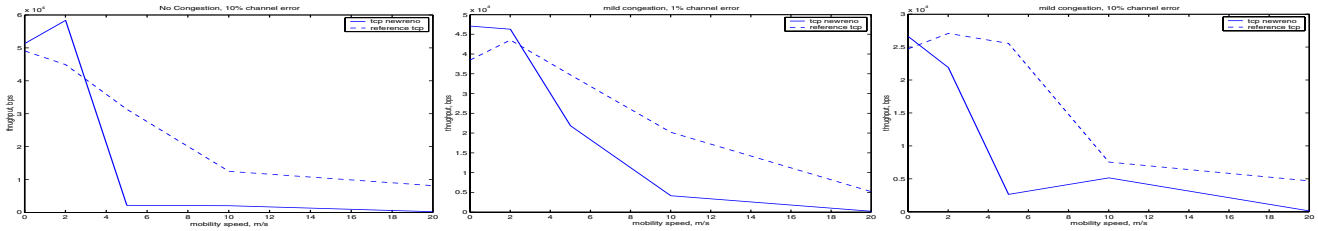

Figure 4. *Comparison of TCP NewReno and R-TCP under more complex conditions - Continued.* Left: Mobility + Channel Error 10%, Middle: Mobility + Congestion + Channel Error 1% Right: Mobility + Congestion + Channel Error 10%

high level (10% or even higher), as exemplified by figure 4 (Left) and (Right), both R-TCP and TCP NewReno suffer from bad channel conditions and experience periodical blackout. One interesting observation from these two figures is that the performance at mobility speed $2m/s$ is higher than the static case. From the trace file, we find that here there exists a prolonged bursty channel error, however, at the same time the node mobility induces a path re-selecting process which greatly lessens the impact imposed by this channel error, thus improves the overall performance.

3.6 Some Special Cases

We also test the impact of other factors including advertised window size, routing protocol (AODV versus DSR) selection, buffer size and average hop number. To emulate real environments, without specification, we set the wireless channel error rate as 1% by default, we also assume that nodes randomly move at a speed $5m/s$ within a $400m \times 800m$ topology. Two competing UDP flows are introduced in the same way as in previous section to create a mild congestion. There are 30 mobile nodes in the topology. The average hop number between the sender and the receiver is 6. The default TCP parameter settings are the same as in previous simulations.

Parameters Related to TCP Figure 5-1 shows the effect of increasing TCP advertised window size from 1 to 32 packets. The throughput of TCP NewReno is only one fourth of that of R-TCP when the advertised window size is 2, and the absolute throughput gap is at least 10K bps throughout all the scenarios we simulated.

For both TCP NewReno and R-TCP, the maximum performance occurs when the window size is 2 packets and the average path length is 6 hops. This confirms the observation in [6].

Routing Protocol Figure 5-2 shows the performance when applying another ad hoc routing protocol AODV. We have similar observation. TCP NewReno performs consistently worse than R-TCP. The performance improvement of R-TCP over NewReno ranges from 30% at mobility speed at 5 m/s to 300% at high mobility speed of 20 m/s. Figure 5-3 is on the effect of the buffer size at each intermediate router node. For TCP NewReno, smaller buffer size helps reduce the RTT, and makes TCP sender to time out and retransmit more promptly. Larger buffer size, however, helps to relieve the congestion to some extend.

Parameters of MAC-Layer In Figure 5-4, while keeping the node density fixed, we vary the total number of nodes and topology size to study the effect of different hop lengths. Since the nodes constantly move in the topology, the hop number is an average over the entire simulation run. We observe that for hop length greater than 9, the potential improvement space for TCP NewReno is rather limited, i.e., TCP is not efficient in long-hop transmission in ad hoc networks.

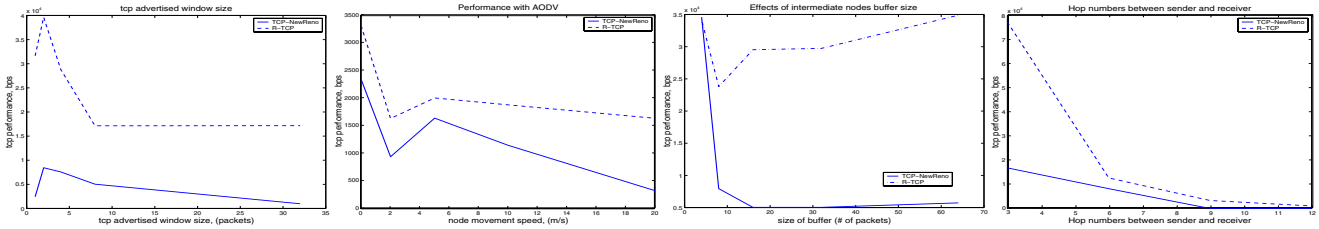

Figure 5. *Comparison of TCP NewReno and R-TCP under some special cases.* Left to Right: 1) Advertised Window Size; 2) Using AODV as Routing Protocol; 3) Buffer Size at each node; 4) Average Hop Number of data path.

3.7 Simulation Summary

In the above simulations, we study TCP performance in wireless ad hoc networks with the assistance of R-TCP. We first study the impact of each individual factor of congestion, mobility and channel error. We observe that TCP does not respond efficiently to network disconnections. Node mobility has the most significant impact on TCP performance. On the other hand, congestion and channel error do not have significant negative effect on TCP (with less than 10% performance drop compared with R-TCP). This confirms that channel error is shielded pretty well by the retransmissions of 802.11 MAC protocol and routing protocol such as DSR.

Then we simulate the composite effects of congestion, mobility and channel error on TCP performance. We observe that as the mobility speed varies, the relative performance degradation ranges from around 0% in zero mobility case to 1000% in extremely high-mobility case (20 m/s). Also the effect of mobility on TCP increases as the mobility speed increases. This is true for both TCP NewReno and R-TCP. The reason is that, at extremely high-mobility cases, the routing protocol becomes performance bottleneck. In addition, we find that congestion tends to reduce the performance gap introduced by mobility.

Finally, in the special cases we study, we observe that TCP NewReno has a large room to be improved compared with R-TCP. The performance gain range from 10% in Figure 5-4, when average hop number is 9 hops between source and receiver; up to 700% in Figure 5-1 where the advertised window size is 32 packets.

4 Conclusions and Future Work

In this paper, we focus on TCP performance in wireless ad hoc networks. We consider how much TCP performance degrades in such networks. We also analyze the impact of different characters of ad hoc networks on TCP throughput. To this end, we design and implement a reference TCP protocol in *ns-2*, which serves as a benchmark in our experiments and analysis.

In the simulation, we show the performance gap between NewReno and R-TCP ranges from 0% to 1000%. Among factors of congestion, mobility and channel errors, we found that mobility-induced events, especially disconnection and reconnection, have the most significant effect on the performance of TCP NewReno. For example, when mobility is 20 m/s, the performance drop is 1000%. As to channel error, low channel error could be masked by retransmissions at link layer, therefore only less than 10% throughput decrease is observed in this case. However, heavy channel error could quickly stall NewReno performance even in medium mobility speed. This again causes 1000% performance gap between NewReno and R-TCP.

We believe that designing this R-TCP quantifies an achievable performance upper bound for any future TCP design in mobile ad hoc networks. In the future, we plan to design practical TCP modifications in ad hoc networks to fill the performance gap between TCP and R-TCP.

References

[1] G. Holland and N. Vaidya, Analysis of TCP performance over mobile ad hoc networks,*MOBICOM'99*.

[2] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, WTCP: A reliable transport protocol for wireless wide-area networks, *MOBICOM'99*.

[3] H. Balakrishnan, and R. Katz, Explicit loss notification and wireless web performances, *Globecom'98*.

[4] S. Floyd, TCP and explicit congestion notification, *ACM CCR*, 1994.

[5] J. Monks, P. Sinha and V. Bharghavan, Limitations of TCP-ELFN for ad hoc networks, *MOMUC'00*.

[6] M. Gerla, K. Tang, and R. Bagrodia, TCP performance in wireless multihop networks, *WMCSA'99*.

[7] K. Chandran, S. Raghunathan, S. Venkatesan and R. Prakash, "A feedback based scheme for improving TCP performance in ad-hoc wireless networks," *ICDCS 1998*.

[8] T. D. Dyer and R. V. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," *MOBIHOC 2001*.

[9] J. Li, C. Blake, D. Couto, H. Lee, and R. Morris, "Capacity of ad hoc wireless networks," *MOBICOM 2001*.