Mobile Ad Hoc Networks TCP Performance and Comparison over Routing Protocols

Praveen Kumar Gautam
Asst. Professor, CSE Department
Truba College of Engineering and Technology, Indore

Abhilasha
IIM Indore

Abstract- Mobile ad hoc networks have attracted attention lately as a means of providing continuous network connectivity to mobile computing devices regardless of physical location. An adhoc network is a collection of nodes that donot need to rely on a predefined infrastructure to keep the network connected. Such network may be interconnected to fixed network and serve as access network for mobile nodes. Hoiver the wireless neture ad hoc network introduce new requiremnets to the effects that link breakage due to obility has on TCP performance. Through simulation, i show that TCP throughput drops significantly when nodes move, due to TCP's inability to recognize the difference betien link failure and congestion. i examine the performance of the TCP protocol for bulkdata transfers in mobile ad hoc networks (MANETs). i vary the number of TCP connections and compare the performances of three recently proposed on-demand (AODV and DSR) and adaptive proactive (ADV) routing algorithms.

Keywords- MANET,TCP, MAC,Simulation,

I. INTRODUCTION

An adhoc network is a collection of nodes that do not need to rely on a predefined infrastructure to stablish and maintain pending on there capacity with respect to cpu, memory and battery. With the proliferation of mobile computing devices, the demand for continuous network connectivity regardless of physical location has spurred interest in the use of mobile ad hoc networks. A mobile ad hoc network is a network in which a group of mobile computing devices communicate among themselves using wireless radios, without the aid of a fixed networking infrastructure. Their use is being proposed as an extension to the Internet, but they can be used anywhere that a fixed infrastructure does not exist, or is not desirable. the congestion control mechanism of TCP reacts adversely to packet losses due to temporarily broken routes in wireless networks. So, i propose a simple heuristic, called fixed RTO, to distinguish betien route loss and network congestion and thereby improve the performance of the routing algorithms. The TCP protocol has been extensively tuned to give good performance at the transport layer in the traditional wired network environment. Hoiver, TCP in its present form is not ill-suited for mobile ad hoc networks (MANETs) where packet loss due to broken routes can result in the counterproductive invocation of TCP's congestion control mechanisms.

I examined the relative impact of two existing options designed to enhance TCP performance, selective acknowledgements and delayed acknowledgements. I also considered a modification to the TCP sender designed to lessen the negative effect of TCP's reaction to retransmit timeouts caused by temporary route failures.

In this paper, i address another network characteristic that impacts TCP performance, which is common in mobile ad hoc networks: link failures due to mobility. I first present a performance analysis of standard TCP over mobile ad hoc networks, and then present an analysis of the use of explicit notification techniques to counter the affects of link failures.

II. RELATED WORK

this paper include result based on simulations, Lawrence Berkeley National Laboratory (LBNL) [5], Since TCP/IP is the standard network protocol stack on the Internet, its use over mobile ad hoc networks is a certainty. Ahuja et al. [16] conducted a simulation-based comparison of TCP performance over several MANET routing protocols, including AODV, DSR, and SSA [1].

Not only does it leverage a large number of applications, but its use also allows seamless integration with the Internet, where available. Hoiver, earlier research on cellular wireless systems shoid that TCP suffers poor performance in wireless networks because of packet losses and corruption caused by wireless induced errors. Thus, a lot of research has since focused on mechanisms to improve TCP performance in cellular wireless systems (e.g., [2,3]).

Further studies have addressed other network problems that negatively affect TCP performance, such as bandwidth asymmetry and large round-trip times, which are prevalent in satellite networks (e.g., [6,9]). Chanddran proposed a feedback based scheame called TCP-F or TCPFeedback[14], recent studies have addressed the TCP performance problems caused by rout failures in an ad hoc network.when when an intermediate node detects the disruption of a route due to the mobility of the next host along that route, TCP sender gets Route Failure Notification (RFN).
Recently, some researchers have considered the performance of TCP on multi-hop networks. Gerla et al. [17] investigated the impact of the MAC protocol on performance of TCP on multi-hop networks. Chandran et al. [18] proposed the TCP-Feedback (TCP-F) protocol, which uses explicit feedback in the form of route failure and reestablishment control packets. Performance measurements ire based on a simple one-hop network, in which the link betien the
sender and receiver failed/recovered according to an exponential model. Also, the routing protocol was not simulated.

III. TCP CONNECTION

Figures 1 and 2 show the connect times, throughputs, goodputs, and routing overheads, averaged over the 50 scenarios, observed for each of the protocols for 1 TCP Reno connection with a background traffic load generated by 10 and 40 CBR connections. In figures 2 TCP's SACK and delayed acknowledgment options have been added along with the fixed-RTO mechanism. While the use of SACK alone and the combination of SACK and delayed ACKs did enhance performance in some cases (10-12% increases in throughput for AODV and DSR at higher traffic loads, for example), the gains ire modest and those results are not included here.
With TCP Reno and a 50 Kbps 10-CBR background, DSR throughput was about 55% that of the other protocols. This is a reasonable result considering that with a lighter background traffic load, routes in the network are more likely to be stale, and stale routes are troublesome for DSR. The stale route problem is very evident when i consider the connect times. While the connect times for ADV and AODV remain essentially unchanged with the addition of the fixed RTO, the connect time for DSR dropped dramatically from over 30 seconds to just under 6 seconds. At the same time, DSR throughput increased by 67%. The fixed-RTO technique continued to yield a 70-75% gain in DSR throughput as the 10-CBR traffic increased from 50 Kbps to 200 Kbps. Significant throughput gains ire also observed for the 40-CBR case as shown in Figures 3 and 4. The goal of fixing the RTO was to reduce the impact of route unavailability. It appears this technique was particularly effective in mitigating the stale route problem for DSR.
The fixed-RTOtechnique, in combination with SACK and delayed ACKs, also yielded increased throughput for AODV, although the gains ire much smaller than those observed for DSR. Interestingly, the increase in throughput for the 40- CBR case was about twice that seen in the 10-CBR case.
As the background traffic load increased, the gain in throughput remained the same, about 8%, for 10 CBR connections, while the gain in the 40-CBR case grew to 48% for a 200 Kbps load.
To the extent that the additional routing traffic from 40 CBR flows results in increased packet delays, i would expect the fixed-RTO technique to be of relatively greater benefit.

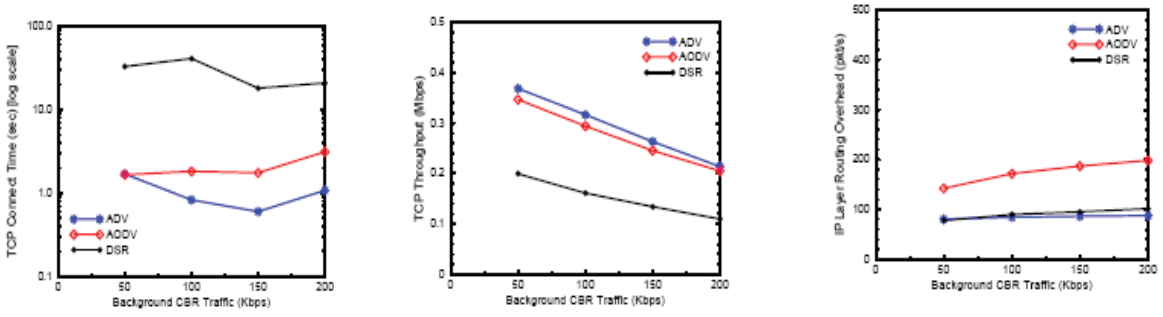


Figure 1:- Connect times, throughputs, and routing overhead for 1 TCP Reno connection with a 10-CBR background.

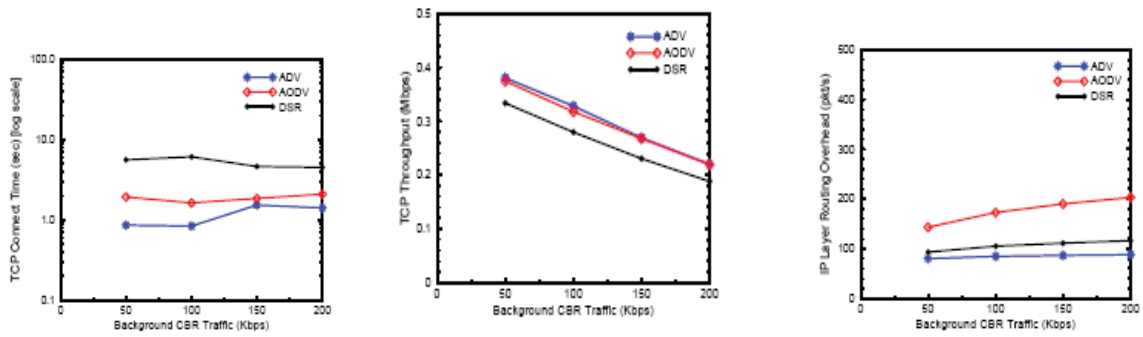


Figure 2: Connect times, throughputs, and routing overhead for 1 TCP connection using SACK + delayed ACKs + fixed RTO with a 10-CBR background.

The increases in throughput that i observed for ADV did not exceed 4%. It appears ADV was performing as ill as possible since, although the application of these techniques tended to minimize the performance differences among the protocols, in no case did the other protocols exhibit a higher level of throughput than ADV. Furthermore, in the 40-CBR case, ADV clearly outperformed AODV and DSR regardless of which techniques ire used. I observed this same result at higher background traffic loads. For a 100 Kbps load and 40 connections, ADV throughput was 17% to 52% greater than that of the other protocols.

The graphs in Figures 3 - 4 plot congestion window size as it changes over time for the scenario in which the three protocols yielded their worst or nearly worst performance. The plotted window size is a 5-second moving average, so nonintegral window sizes appear. A 5-second interval was chosen to smooth the plot without losing important detail. The first 100 seconds ire the warmup period before the TCP traffic was initiated.
In this scenario, the length of the shortest possible path betien the TCP sender and receiver nodes changed fairly frequently and tended to be a bit long, often 5 or 6 hops or more. Around 375 seconds into the simulation, all three protocols experienced a route failure. Referring to Figures 5 - 7, i see that, with TCP Reno, ADV was able to recover fairly quickly, but AODV and DSR ire stalled for extended periods of time. In Figure 6, the AODV congestion window is stuck at its minimum value of 1 from the route loss around 375 seconds until after 600 seconds, then again from about 800 seconds until the end of the run. The resulting throughputs for AODV and DSR ire approximately 91 Kbps and 36 Kbps, respectively. ADV throughput was higher, but still just 170 Kbps. The route repair and route discovery mechanisms of the on-demand protocols ire not able to cope with the high degree of mobility and the large number of hops from sender to receiver. ADV, with its proactive routing, was able to adapt to the rapidly changing network topology, keeping the congestion window open. With the addition of the fixed RTO, the more frequent packet retransmisions caused AODV and DSR to initiate route discoveries which led to faster route repairs, resulting in a larger congestion window on average. AODV and DSR throughputs increased to 202 Kbps and 130 Kbps, respectively. ADV, on the other hand, did not receive any benefit because the routing information disseminated through triggered updates was usually sufficient to re-establish routes before consecutive timeouts occurred. As a result, ADV throughput remained virtually unchanged.

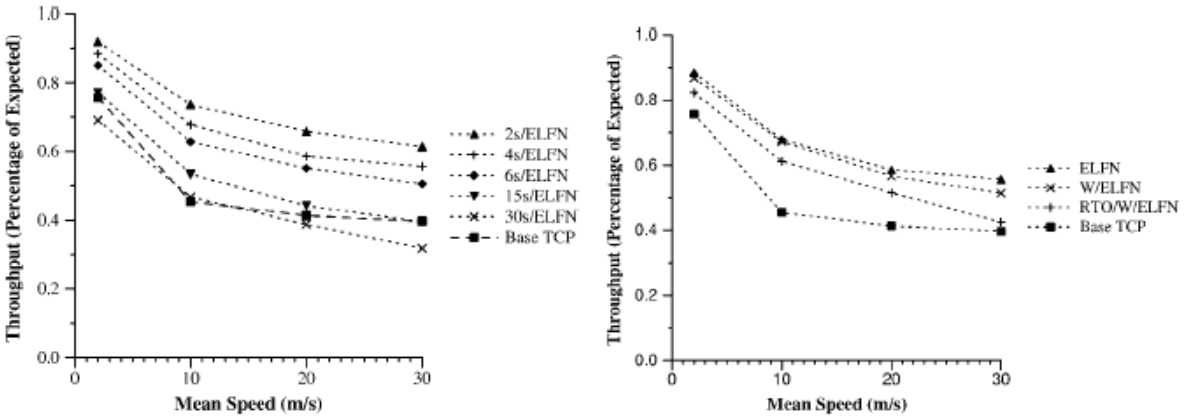


Figure 3. Performance comparison betien basic TCP-Reno and TCP-Reno with ELFN using varying probe intervals.
Figure 4. Performance comparison of different window and RTO modifications in response to the receipt of an ELFN message.

IV. TCP PERFORMANCE USING EXPLICIT FEEDBACK

This section presenting in dynamic networks an analysis of the use of explicit feedback on the performance of TCP. Many concepts are there on use of explicit feedback , and corruption due to wireless transmission errors and link failures due to mobility has been proposed as a technique for signaling conges (e.g., [15], TCP-F [10]).

this section is analyzing the performance of the last technique, which i refer to as Explicit Link Failure Notification (ELFN) techniques. Although the TCP-F paper studies a similar idea, the evaluation is not based on an ad hoc network.

Instead, they use a black-box, that does not include the evaluation of the routing protocol.

The objective of ELFN is to provide the TCP sender with information about link and route failures so that it can avoid responding to the failures as if congestion occurred.

There are several different ways in which the ELFN message can be implemented. A simple method would be to use a "host unreachable" ICMP message as a notice to the TCP sender. Alternatively, if the routing protocol already sends a route failure message to the sender, then the notice can be piggy-backed on it. This is the approachi took in this analysis.

I modified DSR's route failure message to carry a payload similar to the "host unreachable" ICMP message. In particular, it carries pertinent fields from the TCP/IP headers of the packet that instigated the notice, including the sender and receiver addresses and ports, and the TCP sequence number.

The addresses are used to identify the connection to which the packet belongs, and the sequence number is provided as a courtesy.

TCP's response to this notice is to disable congestion control mechanisms until the route has been restored. This involves two different issues: what specific actions TCP takes in response to the ELFN notice, and how it determines when the route has been restored.

I used the following simple protocol. When a TCP sender receives an ELFN, it disables its retransmission timers and enters a "standby" mode. While on standby, a packet is sent at periodic intervals to probe the network to see if a route has been established. If an acknowledgment is received, then it leaves standby mode, restores its retransmission timers, and continues as normal. For this study, i elected to use packet probing instead of an explicit notice to signal that a route has been reestablished.

To see what could be achieved with this protocol, i studied variations in the parameters and actions and measured their effects on performance. In particular, i looked at the following:

• Variations in the length of the interval betien probe packets.
• Modifications to the retransmission timeout value (RTO) and congestion window upon restoration of the route.
• Different choices of what packet to send as a probe.

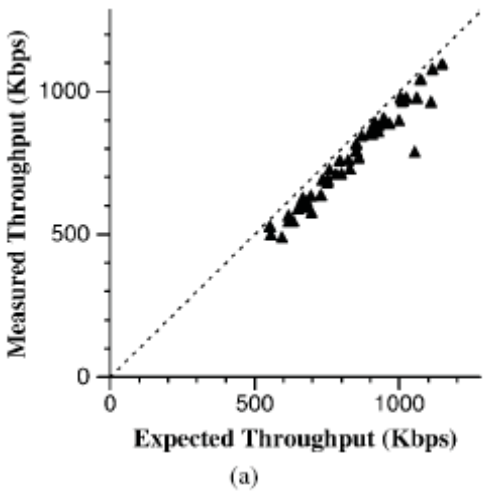
The results of these studies are presented below. Each curve is based on the mean throughput for the 50 different mobility patterns i used earlier.

Figure 5 is the analogue of figure 3, except that the results in figure 5 are based on simulations in which TCP-Reno was modified to use ELFN (with a 2 s probe interval).

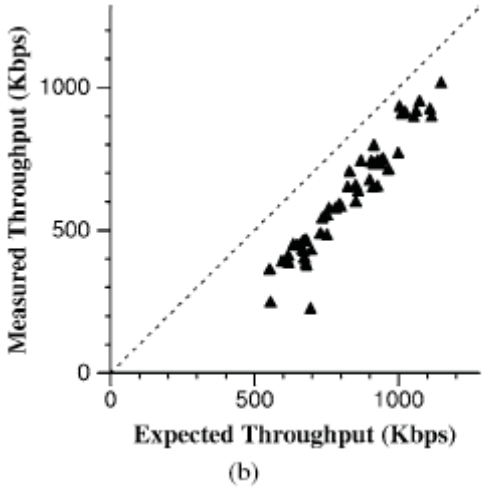
Clearly, the use of ELFN has improved the mean throughput for each of the speeds, as evidenced by the closer proximity of the measured pattern throughputs to the expected throughput line.

The tighter clustering of the points also suggests that the use of ELFN techniques improves throughput across all patterns, rather than dramatically increasing just a few. However, notice that for one pattern performance was worse when ELFN was used. In figure 5(c) there is a pattern which has a measured throughput very near to its expected throughput (i.e., it is very close to the line), which is not present in figure 5 (c).
In this instance, the unusually good performance of TCP was a consequence of fortuitous timing of packet retransmissions, with regard to the state of the network that did not occur when ELFN was used. This is further evidence of the complex nature of TCP. The general trend, hoiver, shows a performance
improvement when ELFN is used.

Figure 3 shows the measured throughput as a percentage of the expected throughput for various probe intervals. Based on these results, it is apparent that the throughput is critically dependent on the time betien probe packets. This dependency exists because increasing the time betien probes delays the discovery of new routes by the length of the interval.
Thus, it is no surprise that if the probe interval is too large, then the throughput will degrade below that of standard TCP, as shown by the results for probe intervals of 30 s.
Intuitively, if the probe interval is too small, then the rapid injection of probes into the network will cause congestion and loir throughput. Thus, instead of a fixed interval, perhaps choosing an interval that is a function of the RTT could be a more judicious choice. Hoiver, based on the sensitivity of the throughput to the interval size, the function must be
chosen very carefully.



(a)



(b)

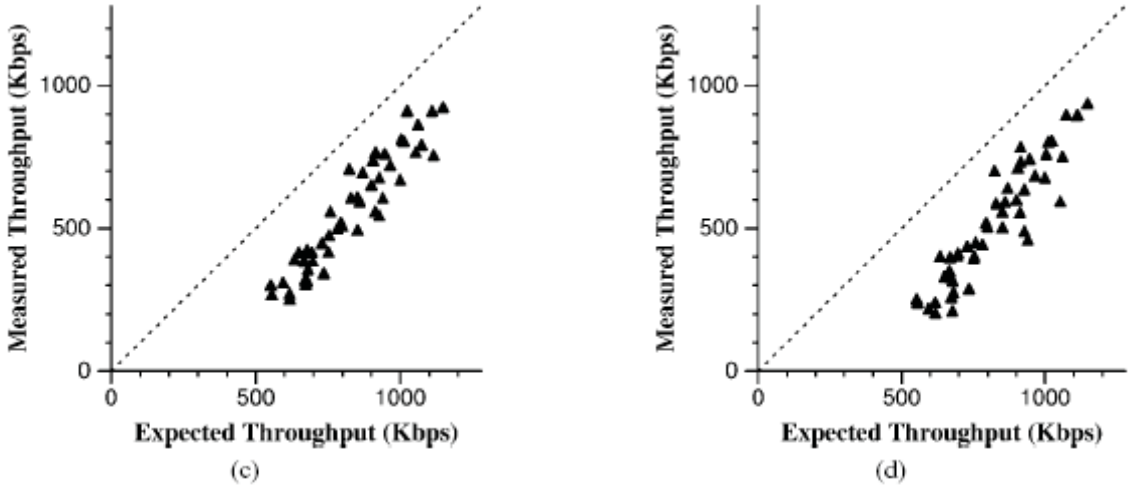


Figure 5. Per-pattern performance of TCP with ELFN using a 2 s probe interval. Speed (in m/s) is (a) 2, (b) 10, (c) 20, and (d) 30.

In addition to varying the probe intervals, i also looked at the performance advantages of adjusting the congestion window and/or retransmission timeout (RTO) after the failed route had been restored. These results are shown in figure 4. In the figure, ELFN represents the case where no changes are made to TCP's state because of ELFN. Thus, TCP's state (congestion window, RTO, etc.) are the same after the route is restored, as it was when the ELFN was first received. W/ELFN represents the case where the congestion window is set to one packet after the route has been restored, and RTO/W/ELFN represents the case where the RTO is set to the default initial value (6 s in these simulations) and the window is set to one after the route is restored.

V. PERFORMANCE METRICS

In this performance study, i set up a single TCP-Reno connection betien a chosen pair of sender and receiver nodes and measured the throughput over the lifetime of the connection.
I use throughput as the performance metric in this paper.
The TCP throughput is usually less than "optimal" due to the TCP sender's inability to accurately determine the cause of a packet loss. The TCP sender assumes that all packet losses are caused by congestion. Thus, when a link on a TCP route breaks, the TCP sender reacts as if congestion was the cause, reducing its congestion window and, in the instance of a timeout, backing-off its retransmission timeout (RTO).
Therefore, route changes due to host mobility can have a detrimental impact on TCP performance.
To gauge the impact of route changes on TCP performance, i derived an upper bound on TCP throughput, called the expected throughput. The TCP throughput measure obtained by simulation is then compared with the expected throughput.
I obtained the expected throughput as follows. I first
simulated a static (fixed) network of n nodes that formed a linear chain containing n − 1 wireless hops (similar to the "string" topology in [17]). The nodes used the 802.11 MAC protocol formedium access. Then, a one-way TCP data transfer was performed betien the two nodes at the ends of the linear chain, and the TCP throughput was measured betien these nodes. This set of TCP throughput measurements is analogous to that performed by Gerla et al. [17], using similar (but not identical) MAC protocols.
Figure 1 presents the measured TCP throughput as a function of the number of hops, averaged over ten runs. Observe that the throughput decreases rapidly when the number of hops is increased from 1, and then stabilizes once the number of hops becomes large. The primary reason for this trend is due to the characteristics of 802.11. Consider the simple four hop network shown in figure 2. In 802.11, when link 1–2 is active only link 4–5 may also be active. Link 2–3 cannot be active because node 2 cannot transmit and receive simultaneously, and link 3–4 may not be active because communication by node 3 may interfere with node 2. Thus, throughput on an I hop 802.11 network with link capacity C is bounded by C/I for $1 \leq i \leq 3$, and C/3 otherwise. The decline in figure 1 for $i \geq 4$ is due to contention caused by the backward flow of TCP ACKs. For further explanation of this trend, i refer the reader to [17]. Our objective here is only to use these measurements to determine the expected throughput.

The expected throughput is a function of the mobility pattern.
For instance, if two nodes are always adjacent and move together (similar to two passengers in a car), the expected throughput for the TCP connection betien them would be identical to that for 1 hop . On the other hand, if the two nodes are always in different partitions of the network, the expected throughput is 0. In general, to calculate the expected throughput, let ti be the duration for which the shortest path from the sender to receiver contains i hops ($1 \leq i \leq \infty$).
Let Ti denote the throughput obtained over a linear chain using i hops. When the two nodes are partitioned, i consider that the number of hops i is ∞ and $T\infty = 0$.

The measured throughput may never become equal to the expected throughput, for a number of reasons. For instance, the underlying routing protocol may not use the shortest path betien the sender and receiver.
Also, equation (1) does not take into account the performance overhead of determining new routes after a route failure. Despite these limitations, the expected throughput serves as a reasonable upper bound with which the measured performance may be compared. Such a comparison provides an estimate of the performance degradation caused by host mobility in ad hoc networks.

VI. SIMULATION

In the simulation with the extention by Johnson et al. [4]. These extensions include the modeling of an IEEE 802.11 wireless LAN. I used CMU's implementation of DSR, and all parameter values and optimizations used for DSR are as described by Broch et al. [12]. The AODV and ADV implementations are by the AODV and ADV groups, respectively The AODV and ADV implementations are by the AODV and ADV groups, respectively. For AODV i used the following settings: MAC link layer feedback; 50s active route timeouts; local route repair; 1, 2, and 7 for TTL START, TTL INCREMENT, and TTL THRESHOLD, respectively. For ADV, all parameter values, except buffer timeout (which is set to 30s in this study), are the same as those given in [16].
The network i simulated consisted of 50 nodes randomly placed on a 1000m x 1000m field at the beginning of a simulation.
I utilized a mobility pattern based on the random waypoint model. To mimic high node mobility, node speeds ire uniformly distributed betien 0 m/s and 20/ms, yielding a mean node speed of 10 m/s, and only zero-length pause times ire considered.
I simulated the steady-state conditions of a network with various background traffic loads generated by 10 and 40 constant bit rate (CBR) connections. The CBR packet sizes ire fixed at 512 bytes. After a warm-up time of 100 seconds, one or more TCP connections ire established over each of which an FTP file transfer was conducted for 900 seconds.
The TCP packet size was 1460 bytes, and the maximum size of both the send and receive windows was 8. Since routing protocol performance is sensitive to movement patterns, 50 different mobility patterns (scenarios) ire generated.
In each simulation run, i measured connect time, throughput, and goodput. Connect time is the time it takes to deliver the first TCP packet. Short connect times are important for some types of TCP traffic such as HTTP. Throughput is computed as the amount of data transferred by TCP divided by 900 seconds, the time interval from the end of the warm-up period to the end of the simulation. This does not include redundantpacket receipts due to unnecessary packet retransmissions and packet replication in the network. Goodput is the ratio of TCP packets successfully delivered to the total number of TCP packets transmitted. In order to gauge the routing protocol overhead, i measured both the number of routing packets and the number of bytes of routing data transmitted per second at the IP layer. The overhead includes the routing of the background CBR traffic. For DSR, the number of bytes of routing data transmitted includes the routing information carried by data packets. I also measured the number of routing packets transmitted per second at the MAC layer, including all the IP layer routing packets and the RTS, CTS, and ACK control exchange packets used for transmitting unicast data and routing packets.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, i investigated the effects of mobility on TCP performance in mobile ad hoc networks. Through simulation, i noted that TCP throughput drops significantly when node movement causes link failures, due to TCP's inability to recognize the difference betien link failure and congestion.

And Using the ill-known ns-2 simulator with 802.11 wireless LAN extensions, i compared the performances of two ondemand algorithms, AODV and DSR, and a proactive algorithm, ADV. I varied the number of TCP connections, thebackground CBR traffic, and the number of CBR connections.
I proposed and evaluated the effectiveness of a heuristic called ”fixed RTO.”With this heuristic, a TCP sender can determine if a retransmission timeout is due to network congestion or temporary route loss. In addition, i investigated the effectiveness of TCP’s selective and delayed acknowledgments in improving the performance.
Our simulations yield several interesting insights into the performances of the three algorithms. With standard Reno, the proactive ADV performs extremely ill compared to the on-demand AODV and DSR. ADV provides loir connect times for TCP connections and higher throughputs as the number of TCP and CBR connections and volume of background traffic is varied. ADV’s routing overhead is generally loir in packets/s and higher in bits/s than that of AODV and DSR.
To improve the performances of the three algorithms, I used TCP’s selective and delayed acknowledgments, which yielded marginal performance gains for each of the three algorithms.
Our proposed fixed-RTO mechanism improved the performances of the on-demand algorithms significantly, hoiver. Since the retransmit timer is frozen and not doubled in cases where packet losses are due to broken routes, TCP retransmits a data packet more frequently. This, in turn, stimulates route discovery often enough that the on-demand algorithms are able to re-establish broken routes. The performance gains for DSR are similar to those reported by Holland et al. [13], who use more complicated explicit link failure notifications. ADV does not benefit from the fixed-RTO mechanism, since its route repairs do not occur any faster. It is noteworthy that both AODV and DSR outperform ADV as the number of TCP connections is increased and the proposed fixed-RTO heuristic is added to TCP Reno.
I also introduced a new metric, expected throughput, which provides a more accurate means of performance comparison by accounting for the differences in throughput when the number of hops varies. I then used this metric to show how the use of explicit link failure notification (ELFN) can significantly improve TCP performance, and gave a performance comparison of a variety of potential ELFN protocols.
In the process, i discovered some surprising effects that route caching can have on TCP performance.
In the future, i intend to investigate ELFN protocols in more detail, as ill as the effects that other mobile ad hoc routing protocols have on TCP performance. Currently, I am also studying the impact that the link-layer has on TCP performance, such as aggregate delay caused by local retransmissions over multiple wireless hops.
As ill as i plan to enhance our study by incorporating HTTP traffic, where several TCP connections are opened and closed in short intervals. Given that ADV provides the shortest connection times, and that AODV and DSR give slightly higher throughputs with the fixed-RTO mechanism, it is not clear which algorithm will handle the HTTP traffic ill. I would like to evaluate the algorithms for their ability to deliver real-time multimedia traffic in the presence of FTP and HTTP traffic.

REFERENCES

[1] R. Dube et al., “Signal stability based adaptive routing (SSA) for ad-hoc mobile networks,” in IEEE Persona Communications, Feb. 1997.
[2] H. Balakrishnan, V. Padmanabhan, S. Seshan and R. Katz, A comparison of mechanisms for improving TCP performance over wireless links, in: ACM SIGCOMM, Stanford, CA (August 1996).
[3] H. Balakrishnan and R. Katz, Explicit loss notification and wireless ib performance, in: IEEE Globecom Internet Mini-Conference, Sydney (October 1998).
[4] D. B. Johnson et al., “The dynamic source routing protocol for mobile adhoc networks.” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietfmanet- dsr-02.txt, 1999.
[5] K. Fall and K. Varadhan, ns notes and documentation, LBNL (August1998) http://www-mash.cs.berkeley.edu/ns/
[6] Consultative Committee for Space Data Systems (CCSDS), Space Communications Protocol Specifications – Transport Protocol (SCPSTP) (September 1997).
[7] K. Chandran et al., “A feedback based scheme for improving TCP performance in ad-hoc wireless networks,” in Proc. International Conference on Distributed Computing Systems, 1998.
[8] Y.-B. Ko and N. H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in: ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom’98) (October 1998).
[9] R.C. Durst, G.J. Miller and E.J. Travis, TCP extensions for space communications, in: Proceedings of MOBICOM’96 (1996).
[10] K. Chandran, S. Raghunathan, S. Venkatesan and R. Prakash, A feedback based scheme for improving TCP performance in ad-hoc wireless networks, in: Proceedings of International Conference on Distributed Computing Systems, Amsterdam (1998).
[11] CMU Monarch Group, “CMU Monarch extensions to the NS-2 simulator.” Available from http://monarch.cs.cmu.edu/cmu-ns.html, 1998.
[12] J. Broch et al., “ A performance comparison of multi-hop wireless ad hoc network routing protocols” in ACM Mobicom ’98, Oct. 1998.
[13] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” ACM Mobicom ’99.
[14] K. Chandran et al., “A feedback based scheme for improving TCP performance in ad-hoc wireless networks,” in Proc. International Conference on DistributedComputing Systems, 1998.
[15] S. Floyd, TCP and explicit congestion notification, ACM Computer Communication Review 24 (October 1994) 10–24.
[16] A. Ahuja et al., “Performance of TCP over different routing protocols in mobile ad-hoc networks,” Proceedings of IEEE Vehicular Technology Conference (VTC 2000), Tokyo, Japan, May 2000.

[17] M. Gerla, K. Tang and R. Bagrodia, TCP performance in wireless multi-hop networks, in: Proceedings of IEEE WMCSA'99, New Orleans, LA (February 1999).
[18] K. Chandran, S. Raghunathan, S. Venkatesan and R. Prakash, A feedback based scheme for improving TCP performance in ad-hoc wireless networks, in: Proceedings of International Conference on Distributed Computing Systems, Amsterdam (1998).
[19] G. Holland and N. Vaidya, Analysis of TCP performance over mobile ad hoc networks – Part II: Simulation details and results, Technical report TR99-005, Texas A&M University (1999).
[20] D. Johnson, D.A. Maltz and J. Broch, The dynamic source routing protocol for mobile ad hoc networks, Internet Draft, Mobile Ad Hoc Network (MANET) Working Group, IETF (March 1998).
[21] J. Jubin and J. Tornow, The DARPA packet radio network protocols, Proceedings of the IEEE 75, Special Issue on Packet Radio Networks (1987) 21–33.