ROUTING HTTP TRAFFIC IN A MOBILE AD HOC NETWORK

Thomas D. Dyer
Rajendra V. Boppana
Computer Science Department, The Univ. of Texas at San Antonio, San Antonio, TX 78249
tdyer@cs.utsa.edu boppana@cs.utsa.edu

Abstract— **TCP performance is a critical factor in the successful deployment of HTTP-based applications in mobile ad hoc networks (MANETs). In this study, we evaluate a sender-based heuristic, which has been shown to improve performance for large FTP file transfers, for HTTP traffic loads. Using the ns-2 simulator, we examine the impact of the proposed technique on the HTTP performance of three on-demand (AODV and DSR) and adaptive proactive (ADV) routing protocols. Our results show that the sender-side heuristic yields higher HTTP performance for the two on-demand algorithms; for the proactive ADV, the improvements are not as significant. Furthermore, ADV performs well compared to AODV and DSR in terms of service time and throughput.**

I. INTRODUCTION

Mobile ad hoc networks (MANETs)provide communication without relying on any existing infrastructure. So MANETs can be particularly useful in military situations where a mobile group of soldiers backed by equipment such as tanks and helicopters, need to communicate and retrieve data during a military operation. Given the success and widespread use of Web-based interface for diverse applications, it is imperative that any communication network used for military purposes shall support Web-based applications. The successful deployment of Web-based applications in mobile ad hoc networks (MANETs) requires good HTTP performance, which in turn depends to a large extent on good TCP performance. In this paper, we evaluate the suitability of MANETs for supporting HTTP traffic with and without interfering background traffic from other sources.

Because TCP's congestion control mechanisms were not designed for the mobile wireless environment, in which the network topology may change rapidly and wireless links are shared, TCP performance is known to suffer in MANETs. A TCP sender assumes that the network is congested if a data packet it sent is not acknowledged by the receiver within a certain duration, called the retransmit timeout interval (RTO). The normal TCP sender response to this situation is to retransmit the oldest unacknowledged packet and then double the RTO. If the current retransmission is not successful, the sender must wait twice as long before trying again so as not to add to the network congestion presumed to have caused the packet loss. In a MANET, when the failed retransmission is due to a temporary route failure rather than congestion, this approach can hurt TCP performance. For a MANET routing protocol which relies on route discovery, the exponential backoff of the RTO results in an ever-increasing delay between attempts to repair the broken route.

This research has been partially supported by DOD/AFOSR grant F49620-96-1-0472, NSF grants EIA-9633299 and EIA-0117255.

To mitigate TCP performance problems, we have recently proposed a heuristic which can be employed by a TCP sender to respond to network changes faster and improve overall performance. This technique, called TCP Reno-F, has been shown in simulations to substantially increase TCP throughput for FTP file transfers [5]. However, these results are not directly applicable for HTTP traffic, which consists of interleaving quiescent and bursty communication periods.

Another factor that impacts TCP performance is the choice of underlying routing protocol used to discover and maintain routes in a routing table or route cache. Based on several studies which considered primarily UDP traffic (simulated using constant bit rate, CBR, sources), on-demand algorithms, which do not attempt to maintain routes by exchanging information among nodes unless a currently used path is affected, perform better than proactive algorithms, which refresh routes by exchanging routing tables or neighbor connectivity information even when active paths are unchanged.

In this paper, we evaluate Reno-F in the context of Web-based client-server transactions in which HTTP traffic is injected at variable rates over relatively short-lived TCP connections. We believe this type of traffic is more representative of the network loads one might anticipate in a real-world MANET. Another contribution of this paper is evaluation of the effectiveness of three routing algorithms in supporting HTTP and CBR traffic. More specifically, we compared two on-demand algorithms called AODV [12] and DSR [9] and one proactive algorithm called ADV [1]. The results of our performance analysis demonstrate that with the use of Reno-F, the on-demand protocols AODV and DSR achieve higher throughput and shorter response times. Among the three routing algorithms, ADV outperforms the other on demand algorithms, especially in the presence of background

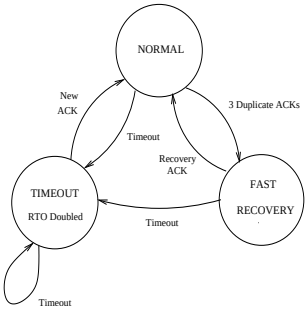

Fig. 1. Simplified TCP state diagram illustrating normal operation of the TCP Reno protocol after the connection has been established. The RTO is doubled on the first timeout and every consecutive timeout thereafter.

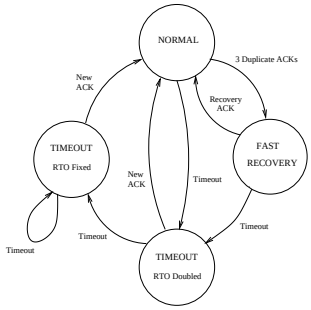

Fig. 2. Simplified TCP state diagram illustrating the fixed-RTO protocol modification. The RTO is doubled on the first timeout, but remains fixed on succeeding consecutive timeouts.

traffic.

II. TCP RENO-F

In MANETs, the loss of a route causes multiple packets to be lost or delayed. (The loss of a route can be temporary, in which case one of the wireless links is down due to interference from other nodes or external sources, or permanent, in which case one of the intermediate nodes has moved away from the radio range of its neighbors for the route.) In the TCP Reno protocol, the TCP sender detects the loss of a packet when its retransmit timer expires before an ACK has been received for that packet. The sender responds by retransmitting the lost packet until it has been acknowledged by the receiver. This is illustrated using a simplified state diagram in Figure 1. When a route failure occurs, the sender is likely to experience multiple timeouts until the route has been repaired and data and ACK packets start moving again. On the other hand, if a TCP sender experiences a single timeout followed by regular flow of ACKs from receiver, then the packet loss is likely due to network congestion or random transmission error [11]. Hence, a TCP sender that is using a wireless interface for its flow can distinguish between the two types of packet loss by interpreting two or more consecutive retransmit timeouts, i.e. timeouts which occur with no intervening acknowledgment of the retransmitted data packet, as a sign of route loss rather than network congestion.

In the proposed modification to the TCP sender, the RTO is fixed rather than doubled when consecutive timeouts occur. The sender doubles the retransmit timeout value just as in the regular TCP protocol for the first timeout; but when another timeout occurs while the sender is in the backoff mode, the sender does not double the retransmit timeout interval. Figure 2 gives the modified state diagram. All other actions performed by a TCP sender are unchanged. Thus the TCP sender retransmits the lost packet at a constant rate, in effect probing the network at regular intervals. The probe interval is based on the current RTO value, and thus is adaptive to network conditions. We denote this modified protocol as TCP Reno-F.

To show the possible impact of fixed RTO, we contrast AODV performance with standard TCP Reno and Reno-F for an FTP transfer. We choose one of the worst-case scenarios for easier illustration. There is a 50 Kbps background network load from 40 different CBR flows. Statistics are collected after warming up the network for 100 seconds. In Figures 3 and 4, the upper graph shows the congestion window size as a function of time along with two sets of hash marks; the upper (darker) hash marks denote retransmissions and the lower (lighter) hash marks denote transmission of a new packet by the sender. The size of the congestion window was limited to a maximum of eight. The lower graph shows the route repair times observed during the simulation; the horizontal dotted line indicates the average route repair time.

In Figure 3, at approximately time 375 a route failure occurs causing the congestion window size to drop to its minimum value of one. Retransmit timeouts follow at progressively longer intervals until the maximum of eight backoffs is reached. The upper hash marks indicate that the sender is probing the network at increasingly longer intervals; the lower hash marks indicate the lack of progress by the TCP sender. The route is not successfully re-established until about time 600, and during this period the observed route repair times shown in the lower graph are generally much longer than those before the route failure at time 375. Longer route repair times, along with fewer attempts to utilize a repaired route before node mobility breaks it again, cause the TCP sender to be stuck in retransmission mode as indicated by a congestion window of size 1 and the big gap in the lower hash marks. With the use of Reno-F in Figure 4, packet retransmissions due to timeouts are more frequent, which in turn stimulates AODV to discover new routes to

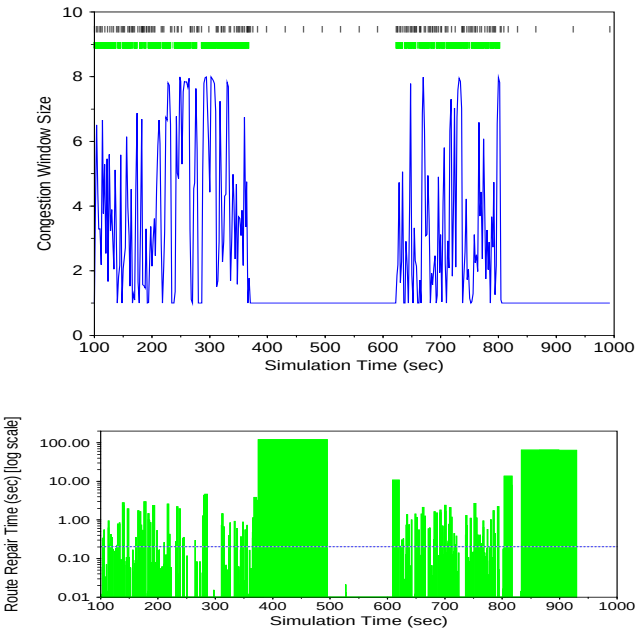

Fig. 3. AODV congestion window sizes and route repair times for 1 TCP Reno connection with a 50 Kbps background load from 40 CBR connections. The upper graph shows the congestion window size as a function of time along with two sets of hash marks; the upper (darker) hash marks denote retransmissions and the lower (lighter) hash marks denote transmission of a new packet by the sender. Average route repair time = 1.627 seconds. Throughput = 0.0914 Mbps.

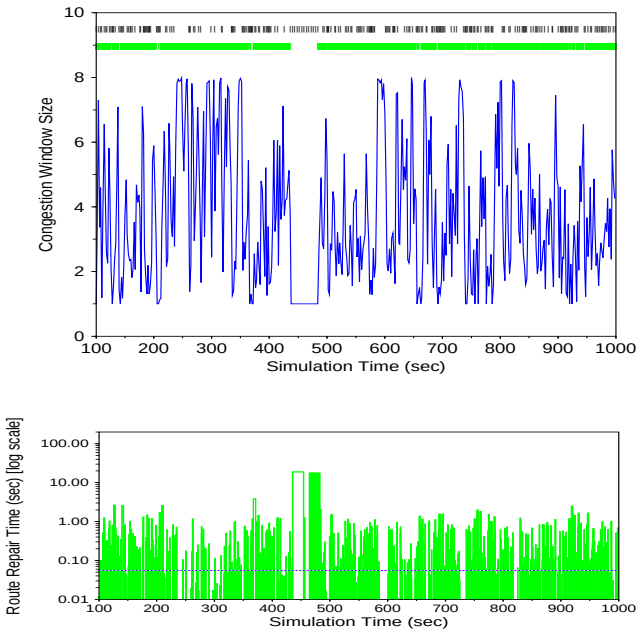

Fig. 4. AODV congestion window sizes and route repair times for 1 TCP Reno-F connection with a 50 Kbps background load from 40 CBR connections. Average route repair time = 0.446 seconds. Throughput = 0.2023 Mbps.

the TCP receiver more frequently, thus reducing route repair time. Looking at the lower graphs, Reno-F reduces the average route repair delay (indicated by horizontal lines) by more than 70% and maximum route repair delay by 80%. So the TCP sender is able to utilize repaired routes quickly and keep the congestion window open.

Since HTTP traffic has interleaving quiescent periods between communication among nodes, which is different from FTP traffic, it will be interesting to see if Reno-F yields any performance improvements over Reno.

III. Simulation Methods

We have used the *ns-2* network simulator [6] with 802.11 MAC extensions by the CMU Monarch group [4] to simulate an ad hoc network comprised of 50 mobile nodes on a 1000m x 1000m field. The simulation model that we used for a mobile node is depicted in Figure 5. Node speeds were uniformly distributed between 0 m/s and 20/ms, yielding a mean node speed of 10 m/s. We used CMU's implementation of DSR, and all parameter values and optimizations used for DSR are as described by Broch et al. [2]. The AODV and ADV implementations are by the AODV and ADV groups, respectively. The local route repair option is turned on for AODV. The maximum size of both the TCP send and receive windows is 8.

We considered two variants of the ADV protocol. In the first version, denoted ADV 30s/30s in the graphs, any data packet (TCP or UDP) may be buffered for up to 30 seconds (denoted buffer refresh time) in the source or an intermediate node when there is no route. In the second version, denoted ADV 30s/1s, the buffer refresh time is 1 second for UDP packets and 30 seconds for TCP packets. In AODV and DSR, a packet may be buffered for up to 30 seconds in its source node; packets that do not have a valid route upon reaching an intermediate node are dropped. The main purpose of using two buffer refresh times for UDP packets in ADV is to see if buffering UDP packets has any adverse impact on the HTTP traffic.

Using an HTTP traffic generator [7], we have simulated 10 Web sessions in which browsers on 10 different mobile nodes issue requests and receive replies from Web servers running on 3 other nodes. Each session consists of an alternating sequence of think and HTTP transaction modes. In the think mode, the client thinks for a random period of time and does not generate any network traffic. In an HTTP transaction, the client issues a request, and the server then responds with a random number of replies of variable length. The think times, the number of replies, and the length of the

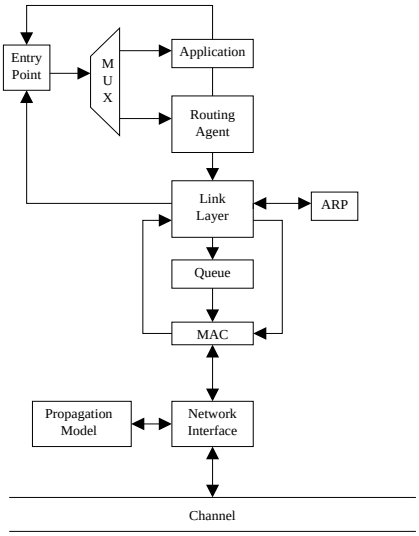

Fig. 5. *ns-2* simulation model of a mobile node.

replies that we used were drawn from the distributions supplied with the traffic generator. However, to keep the Web sessions short enough so that the client-server exchanges could be completed within the duration of the simulation, we truncated the think time distribution at 15 seconds. To make the simulations repeatable, we generated several sessions and stored them in files, which are used as inputs to the simulator.

In addition to Web traffic, we have simulated for some of the experiments a 100 Kbps background UDP traffic load generated by 10 constant bit rate connections. The CBR packet sizes were fixed at 512 bytes. Performance measurements were collected for 100 seconds following an initial warm-up time of 100 seconds. Each data point presented in this paper is an average of values obtained from 50 different mobility scenarios with all other input values identical.

In each simulation run, we have measured service time and throughput for the HTTP connections. Service time is the time taken to complete a client-server session.

IV. Performance Results

To begin with, we have evaluated the performance of the three routing protocols for HTTP traffic with no background CBR traffic. Figure 6 gives the mean service time and throughput achieved by each routing algorithm. Since ADV 30s/30s and 30s/1s perform exactly the same in the absence of background traffic, we give results for only one. The following observations can be made from the figure. First, Reno-F improves the performance of AODV and DSR substantially, but not that of ADV. Because ADV refreshes routes using routing table exchanges as in a typical distance vector routing protocol, (a) routes do not break as often, and (b) broken routes are repaired in about the same time regardless of how frequently the TCP layer is transmitting (or retransmitting) data packets. Among the two table-based algorithms, ADV has marginally higher throughput than AODV. With Reno-F, AODV service times are decreased by 6% and throughputs are up by over 12%. ADV's performance is not impacted significantly by Reno-F. DSR performance with TCP Reno is low because of its well-documented stale route problem; Reno-F mitigates the stale route problem, however, which can be seen by an over 90% improvement in throughput and 16% reduction in service time.

To evaluate Reno-F and the three routing protocols further, we present results from simulations with CBR background traffic. The observed mean service time and HTTP throughput for each routing protocol are shown in Figure 7. Of the three algorithms, AODV is affected the most by the background traffic because of its high overhead in discovering and maintaining routes; DSR is benefited the most because the stale route problem is mitigated greatly by the additional routing activity needed for the background traffic; and ADV is affected the least because the cost of routing update (when measured in packets) remains nearly the same. ADV gives at least 50% more throughput than AODV and DSR in all cases. DSR underperforms both AODV and ADV, but by a smaller margin when Reno-F is used.

Reno-F improves DSR's performance significantly: throughput is improved by 31% and service time reduced by 22%. The impact of Reno-F on AODV and ADV is not significant. AODV's performance does not improve because the impact of background traffic and because of relatively short duration of HTTP sessions. In a separate set of simulations with longer HTTP sessions, not shown here, AODV's performance is improved by Reno-F. ADV's performance does not improve for the reasons given in the case of no background traffic.

To explore further, we present the service times observed for each of the 10 client-server pairs in Figure 8. With Reno-F, the service times for DSR are more comparable to AODV service times, and for 2 client-server pairs DSR achieved slightly shorter times than AODV. ADV maintained its advantage in service times in every case. Finally, the shorter buffer refresh time for UDP packets marginally improves ADV's HTTP performance.

V. Related Work

Several mechanisms have been proposed for improving TCP performance in MANETs. In two of the proposed methods, TCP-F [3] and TCP-BuS [10], an intermediate node informs the TCP sender of a route failure and the sender then stops its transmissions, freezing its timers and congestion window. The sender resumes sending packets

Fig. 6. Combined service times and throughputs for 10 HTTP connections with no CBR background traffic.

Fig. 7. Combined service times and throughputs for 10 HTTP connections with a 100 Kbps background load from 10 CBR sources.

Fig. 8. Service times for 10 HTTP server-client connections using TCP Reno and TCP Reno-F.

when it receives an explicit notification that the route has been re-established. In addition, packets may be buffered by intermediate nodes in TCP-BuS. Another proposed technique, ELFN [8], also employs route failure notification. The TCP sender freezes its state as in TCP-F and TCP-BuS, but rather than waiting for a route re-establishment message, in ELFN the sender probes the network at regular intervals and resumes normal TCP mode when a new ACK is received. With these three methods, the explicit route loss and route establishment messages (for TCP-F and TCP-BuS) create more overhead and require feedback from routing layer protocols. Also ELFN uses a constant probe interval of 2 seconds, while Reno-F uses an adaptive probe interval.

Our earlier work [5], which evaluates Reno-F in the context of FTP traffic with interfering CBR traffic, indicates the following: (a) Reno-F improves the performances of AODV and DSR substantially, and (b) AODV and DSR slightly outperform ADV with 10 FTP flows. The results presented here complement the results given in [5].

VI. Conclusions

In this paper, we have evaluated the performance of MANETs for HTTP traffic, which mimics bursty communication interleaved by quiescent periods. So HTTP traffic could be typical of the communication that need to be supported in a military operation. We have considered the impact of transport and routing protocols on the overall performance. Of the three routing protocols we have simulated, ADV performs as well or better than AODV and DSR when TCP Reno is used as the transport protocol. With a simple sender-side heuristic called fixed RTO, which prevents doubling of the retransmit time out interval for consecutive timeouts, we have shown that AODV and DSR perform bet-

ter, while ADV does not improve.

The primary benefit of the fixed RTO heuristic is to let TCP probe the network much more frequently than it would otherwise. The frequency at which a TCP sender probes the network while in backoff mode is based on the current RTO, and thus is adaptive to the existing network conditions. Since AODV and DSR can discover routes on demand, more frequent probing results in shorter route repair times and overall higher performance. ADV's performance does not improve significantly with the fixed RTO technique for the following reasons: (a) broken routes are repaired only through routing updates among neighbor nodes and more frequent retransmissions by TCP with the fixed RTO heuristic do not have any significant impact on the route repair time, (b) ADV buffers packets at intermediate nodes and delivers packets to destinations in reasonably short enough time that more frequent retransmissions by TCP sender are ineffective, and (c) ADV exhibits relatively good performance with TCP Reno, which means there is less room for improvement.

A possible alternative to fixing the RTO would be to replace the exponential backoff of the RTO with fractional increases in the RTO. Such an approach could lower the number of packet retransmissions while still reducing the negative impact of doubling the RTO. However, the reduction in retransmissions would be somewhat limited. For example, if an increase of 5% were applied to the RTO at each consecutive timeout, only about a 20% decrease in the number of packet retransmissions would accrue after ten timeouts. Moreover, a reduction in packet retransmissions may reduce the degree to which Reno-F is able to stimulate route repair.

Comparing the results for FTP traffic in [5] and those for HTTP traffic in this paper, we note that the performances differ substantially for HTTP traffic: ADV outperforms AODV and DSR significantly when there is interfering non-TCP friendly traffic. Another result of our study is that background CBR traffic can adversely impact the TCP performance when TCP Reno and AODV are used as transport and routing protocols.

Our results show that TCP performance must be evaluated with some background non-TCP friendly traffic. In the future, we would like to expand the study by including UDP-based multimedia traffic.

REFERENCES

[1] R. Boppana and S. Konduru, "An adaptive distance vector routing algorithm for mobile, ad hoc networks," in *IEEE Infocom 2001*, vol. 3, pp. 1753–1762, Mar. 2001.

[2] J. Broch et al., " A performance comparison of multi-hop wireless ad hoc network routing protocols," in *ACM Mobicom '98*, pp. 85–97, Oct. 1998.

[3] 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*, pp. 472–479, May 1998.

[4] CMU Monarch Group, "CMU Monarch extensions to the NS-2 simulator." Available from http://monarch.cs.cmu.edu/cmu-ns.html, 1998.

[5] T. Dyer and R. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," in *MobiHoc '01*, pp. 56–66, Oct. 2001.

[6] K. Fall and K. Varadhan, "NS notes and documentation." The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC. Available from http://www-mash.cs.berkeley.edu/ns, Nov. 1997.

[7] T. Henderson, "HTTP/1.0 background traffic generator for ns." Computer Systems Engineering Group, Lawrence Berkeley Laboratory. Available from http://www.tomh.org/software/httptrafficgen.tar.

[8] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *ACM Mobicom '99*, pp. 219–230, Aug. 1999.

[9] D. B. Johnson et al., "The dynamic source routing protocol for mobile adhoc networks." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, 2002.

[10] D. Kim et al., "TCP-BuS: Improving TCP performance in wireless ad hoc networks, " in *IEEE ICC 2000*, vol. 3, pp. 1707–1713, June 2000.

[11] C. Liu and R. Jain, "Using congestion coherence to enhance TCP over wireless links," manuscript, June 2001.

[12] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on demand distance vector routing." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-10.txt, 2002.