On Routing Web and Multimedia Traffic in Mobile Ad Hoc Networks

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— **We evaluate the capabilities of MANETs in supporting multiple, simultaneous HTTP and multimedia streaming flows. The HTTP traffic is produced by several Web servers, each responding to a series of concurrent requests from multiple Web clients, and multimedia traffic is modelled as a variable bit-rate stream consisting of UDP packets. To see the impact of a routing protocol, we consider two on-demand (AODV and DSR) and one adaptive proactive (ADV) routing protocols. We also consider the impact of a previously proposed TCP-sender heuristic, fixed RTO, on HTTP performance. Our results indicate that ADV performs well relative to the on-demand techniques, providing higher throughput for variable bit rate UDP traffic and significantly reducing the time required to complete Web client-server transactions. Furthermore, our results show that, compared to TCP Reno alone, the fixed RTO technique yields significant gains in the HTTP performance of the two on-demand algorithms.**

I. Introduction

The successful deployment of Web-based and multimedia applications in mobile ad hoc networks (MANETs) requires routing and transport protocols to perform well for the network loads such applications generate. In a MANET, node mobility results in a dynamic network topology in which route failures may be frequent and the available bandwidth can vary dramatically over time. As a consequence, transport-layer protocols built for the wireline Internet may not give adequate performance when used in MANETs. For example, TCP's congestion control mechanisms were not designed for the mobile wireless environment, and TCP performance is known to suffer as a consequence. Poor TCP performance in turn hinders the performance of applications, such as Web browsers, which use HTTP.

A TCP sender assumes that packet losses are caused by network congestion. In the event a packet is not acknowledged by the receiver within a certain duration, called the retransmit timeout interval (RTO), the sender retransmits the unacknowledged packet and then doubles the RTO. This process is repeated until an ACK for the retransmitted packet has been received. This exponential backoff of the RTO enables TCP to handle network congestion gracefully. However, in a MANET, when the failed retransmission is due to a temporary route failure rather than congestion, this approach can hurt TCP performance. To mitigate TCP performance problems, we have recently proposed a heuristic, called fixed RTO, which can be employed by a TCP sender to respond to network changes faster and improve overall performance. We have shown that the fixed RTO technique increases TCP throughput significantly for FTP file transfers [6]. However, these results are not directly applicable for HTTP traffic, which consists of interleaving quiescent and bursty communication periods.

This research has been partially supported by NSF grant EIA-0117255.

The network loads from Web-based and multimedia applications are typically characterized by variable-bit-rate (VBR) traffic flowing over connections of variable durations ranging from a few seconds to several minutes. A MANET routing protocol must therefore be able to establish routes quickly and provide high throughput and low turnaround time over a wide range of offered traffic. For this reason, the algorithm used to discover and maintain routes in a routing table or route cache is of critical importance. Based on several studies [2], [5], 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. However, these results may not hold for HTTP and multimedia flows, which tend to generate bursty traffic.

In this paper, we evaluate the impact of Web and multimedia traffic on each other. We also evaluate the usefulness of fixed RTO technique for bursty TCP traffic in the presence of heavy UDP traffic. We simulate multiple HTTP and multimedia flows with and without interfering background traffic. We believe that these traffic scenarios are more representative of the network loads placed on a real-world MANET.

In addition, it is important to evaluate the impact of a routing protocol on the overall performance. So, we compare two on-demand algorithms called AODV [18] and DSR [14] and one proactive algorithm called ADV [1]. We also investigate the performance impact of maximum packet buffering at the routing layer for one of the routing protocols.

The results of our performance analysis demonstrate that with the use of fixed RTO, the on-demand protocols AODV and DSR achieve significantly better performance than with TCP Reno alone. The pro-active ADV outperforms AODV and DSR by a large margin when the network is subjected to interfering TCP and UDP traffic , while all three algorithms perform about the same when only TCP traffic is simulated. With VBR traffic competing for bandwidth, HTTP throughput is reduced significantly. The fixed RTO heuristic proves to be effective in the presence of heavy traffic caused by the VBR flows.

The rest of the paper is organized as follows. Section II presents the fixed RTO technique. Section III describes the simulation setup. Section IV presents our performance analysis of fixed RTO and the three routing protocols for HTTP and VBR traffic. Section V presents related work. Section VI concludes the paper.

II. TCP Reno-F

We denote the specific application of the fixed RTO heuristic to TCP Reno as TCP Reno-F.

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 retransmit timeout (RTO) value is computed adaptively when packets flow normally, but doubled whenever a timeout occurs. The sender responds to a timeout by retransmitting the lost packet until it has been acknowledged by the receiver. When a route failure occurs, multiple packets may be lost or delayed; so 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 [16]. 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 TCP Reno-F, the RTO is fixed rather than doubled when *consecutive* timeouts occur. The RTO is doubled when the first timeout occurs just as in the regular Reno protocol, but if another timeout occurs while the sender is in the backoff mode, the sender does not double the retransmit timeout interval again. Thus the TCP sender retransmits the lost packet at a constant rate, in effect probing the network at regular intervals. The probe interval is equal to the current RTO value, and thus is adaptive to network conditions.

In an earlier work [6], we have shown that, for FTP traffic, Reno-F improves TCP throughput significantly for the on-demand protocols AODV and DSR and provides much smaller benefit with ADV. Since HTTP traffic has quiescent periods interleaving short-lived, bursty communication among nodes, it will be interesting to see if Reno-F yields any performance improvements over Reno.

III. Simulation Methods

For our simulations, we used the *ns-2* network simulator [7] with the wireless and mobility extensions by the CMU Monarch group [4]. These extensions include the modelling of an IEEE 802.11 wireless LAN [13]. We simulated an ad hoc network comprised of 50 mobile nodes on a 1000m x 1000m field. The nodes move according to a mobility pattern based on the *random waypoint* model; to avoid clustering of nodes in the middle of the field, we let a node reaching an edge of the field to wraparound (instantaneously) and continue its movement in the same direction from the opposite edge of the field [12]. Since a MANET's performance is sensitive to movement patterns, 50 different mobility patterns (scenarios) were simulated and averaged for each data point presented in the plots. Node speeds were uniformly distributed between 0 m/s and 20/ms, yielding a mean node speed of 10 m/s, and only zero-length pause times were considered. We used CMU's implementation of DSR [2]; since DSR is shown to suffer from stale routes problem for TCP traffic [11], [6], we turned of route replies from the route cache. The AODV implementation is by the AODV group [5], we implemented ADV as described in an earlier paper [1]. The maximum size of both the TCP send and receive windows is 8.

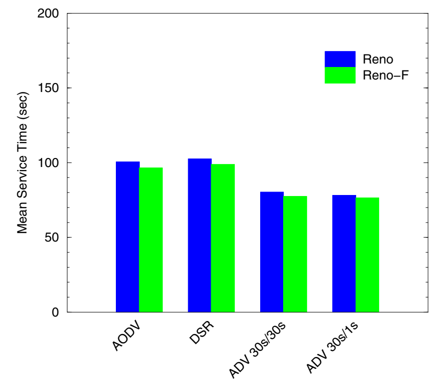
We considered two variants of the ADV protocol. In the first version, denoted ADV 30s/30s in the graphs, any data packet (TCP or UDP) may be buffered for up to 30 seconds (denoted buffer refresh time) in the source or an intermediate node when there is no route. In the second version, denoted ADV 30s/1s, 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 the impact of buffering UDP packets in intermediate nodes on HTTP and VBR traffic.

We simulated the *steady-state* conditions of a network with a background traffic load generated by constant bit rate (CBR) connections. The CBR packet sizes were fixed at 512 bytes. The CBR traffic injected was far below (about 1/3rd of) the maximum UDP traffic the network can handle without saturation, but sufficient to cause interference to other traffics of interest. Performance measurements were collected for 200 seconds following an initial warm-up time of 100 seconds. In addition to the background load, two types of network traffic were generated: Web traffic using the HTTP protocol, and variable bit rate (VBR) multimedia traffic. Both HTTP and VBR are started just after the warmup time.

HTTP traffic. Using an HTTP traffic generator [10], we 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 interleaving sequence of think and transaction modes. In the think mode, there is no traffic on the network by the Web session. In the transaction mode, the client issues a request, and the server then responds with a random number of replies of variable length; we denote this activity as a request-reply cycle. We modified the traffic generator so that the series of client-server exchanges were identical in every simulation run. The think times, the number of replies, and the length of the 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.

VBR traffic. We modelled multimedia traffic as a stream of UDP packets of variable size, generated at a constant rate. Packet sizes were assigned randomly with mean packet size, variance, and minimum and maximum sizes based on the parameters for an actual compressed video stream reported in [8]. Since our simulated 802.11 network cannot handle the average bandwidth of 5.3 Mbps required for that video stream, we scaled the packet rate and size parameters back such that a single simulated VBR connection has an average bandwidth requirement of 53 Kbps.

Performance metrics. We measured service time, response time, and throughput for the HTTP connections. The service time for a Web session is the time spent in the request-reply transaction mode. If a Web session has not finished at the end of simulation, then the service time for that session is computed by adding up the time spent in all completed request-reply cycles, and any time spent in the request-reply cycle that is still in progress; so, for unfinished Web sessions, the service time is (200 seconds - time spent in think mode). For a finished Web session, the service time is (time taken to finish the Web session - time spent in think mode). Response time is the interval between the sending of a client request and the receipt of the first packet of the server's reply in a request-reply cycle. HTTP throughput is total number of TCP bytes delivered in all 10 Web sessions divided by the total service time for all 10 Web sessions. Bytes delivered in request-reply cycles that are in progress when the end of simulation are included in calculating the throughput. The mean service time is a simple arithmetic mean of the service times for 10 Web sessions. The average response time is an arithmetic mean of all request-reply cycles in the 10 Web sessions. The graphs below present average service time for a Web session, aggregate throughput for 10 Web sessions, and average response time for a request-reply cycle. For VBR traffic, we measured packet latency, packet jitter, throughput and packet delivery fraction. For simulations with CBR traffic, we also measured CBR packet latency and throughput.



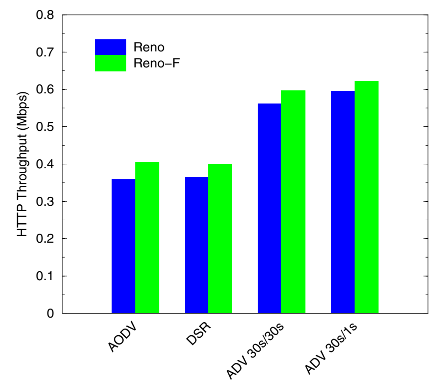


Fig. 1. Mean service time and throughput for 10 HTTP connections with a 100 Kbps background load from 10 CBR sources.

IV. Performance Results

We first present results on HTTP traffic with background UDP traffic simulated using constant bit rate flows. Next, we present results on VBR traffic in the presence of background CBR flows. Finally, we present results from simulations that include both HTTP and VBR traffic.

A. HTTP Traffic

We evaluated the impact of Reno-F and compared the performance of the three routing protocols for HTTP traffic with a low, 100-Kbps, CBR background traffic. We also investigated the impact of Reno-F and buffering UDP packets on CBR traffic. The observed mean service time and HTTP throughput for each routing protocol are shown in Figure 1.

TCP Reno-F improved, though not significantly, the performance of the on-demand protocols AODV and DSR. Mean service time was reduced by less than 10% for AODV and DSR. HTTP throughput gains of 22% and 14% were observed for AODV and DSR, respectively. DSR with cache turned off performed as well as AODV for both versions of the TCP protocol. Reno-F has no significant impact on ADV. For both 30s/30s and 30s/1s cases, the service times and throughputs were improved by less than 4%, and 12%, respectively. Of the two, the 30s/1s was marginally better.

Comparing AODV, DSR, and ADV 30s/30s, we note that ADV gave 61% more throughput when TCP Reno was used as the transport protocol. With Reno-F, ADV still outperformed AODV and DSR by about 47%. Furthermore, ADV completed HTTP sessions, as measured by the mean service times, 20% faster, for both TCP protocols.

Since the above analysis was based on averages of 10 HTTP flows, we calculated the mean service and response times of each flow to ensure that there is no random worst-case that is skewing the results and making AODV and DSR look worse than they appear. In Figures 2 and 3, we present the service and response times observed for each of the 10 client-server pairs with background CBR traffic. AODV and DSR performed about the same. ADV maintained its advantage in service times in *every* case. As in the case of service times and HTTP throughput, Reno-F lessened the differences in response times for the three protocols. ADV yielded shorter response times for every client-server pair, although with Reno-F the advantage over AODV and DSR was lessened. The shorter buffering time for UDP packets had little effect on ADV response times.

In an earlier work [6], we reported that for FTP traffic, AODV and DSR perform as well or better than ADV, and that Reno-F improves performances of AODV and DSR significantly. Also, it is noteworthy that with 10 FTP flows, all three routing protocols achieved an aggregate throughput of over 1 Mbps, while for 10 HTTP flows, even the best-performing ADV achieved only 60% of it. To understand why the results for HTTP traffic differ, let us take a look at the TCP traffic patterns. While FTP flows were simulated with infinite backlog, with no idling once the connection is established, HTTP flows consist of several finite-sized request-reply transactions interleaved by thinking times. The time spent in think mode is comparable to the time spent in transaction mode. Even without experiencing packet losses, an HTTP client (or server) needs go through slow start several times during the simulation. Because of the finite-sized transactions, the time spent on slow start is a significant portion of the overall transaction time. Also, the response time, during which typically small amounts of data are sent by a client, reduce the throughput calculated. A routing protocol has to be ready with routes to handle such short-lived TCP transactions. The results indicate that ADV excelled in this situation. Also, given the short amounts of data transacted, the fixed RTO heuristic was probably not invoked as frequently as in the FTP traffic simulations.

CBR Flows. Though CBR traffic was considered a background traffic interfering with the main (HTTP) traffic in these simulations, we evaluated the impact of Reno-F on

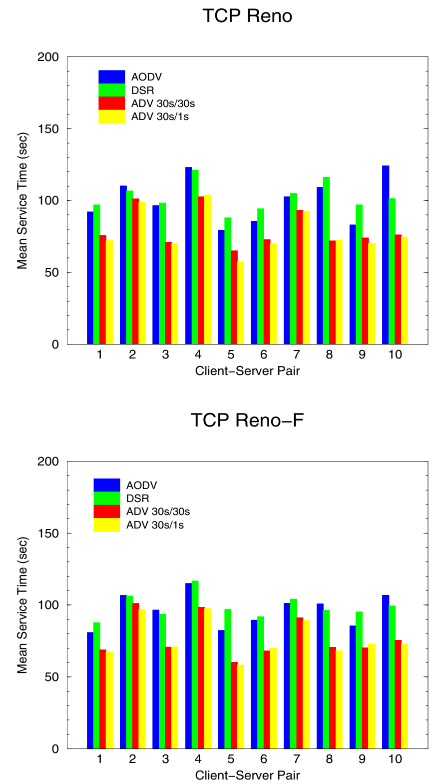


Fig. 2. Service times for 10 HTTP server-client connections using TCP Reno and TCP Reno-F.

CBR traffic. Figure 4 shows the packet latency and throughput observed for the background CBR flows. CBR throughput was comparable for the three routing protocols, with almost no impact from the use of Reno-F. The packet loss was low, 5% for ADV and 10% for AODV and DSR. Let us consider the packet latencies. AODV gave the lowest CBR packet latencies and is not impacted by Reno-F. For DSR and ADV, however, the CBR packet latencies were *lower* when Reno-F was used. This is counter intuitive, since Reno-F is a more aggressive protocol than Reno, we expected the latencies to increase or remain the same at best. Indeed, our earlier FTP simulations [6], which simulated TCP traffic with infinite backlog, showed that Reno-F increased CBR packet latencies measurably. Upon analyzing the data further, we observed the following. Depending on the mobility pattern and interfering traffic from other Web sessions and CBR traffic, a Web session could or could not finish within the 200 seconds simulation period. If a Web session finished with Reno, then applying Reno-F almost always resulted in quicker completion time. If a Web session did not finish with Reno, then applying Reno-F generally resulted in (a) completion of more request-reply cycles for that session, or (b) higher number of retransmissions, which is likely for sessions with pathologically bad routes. So, given that each Web session has only a finite

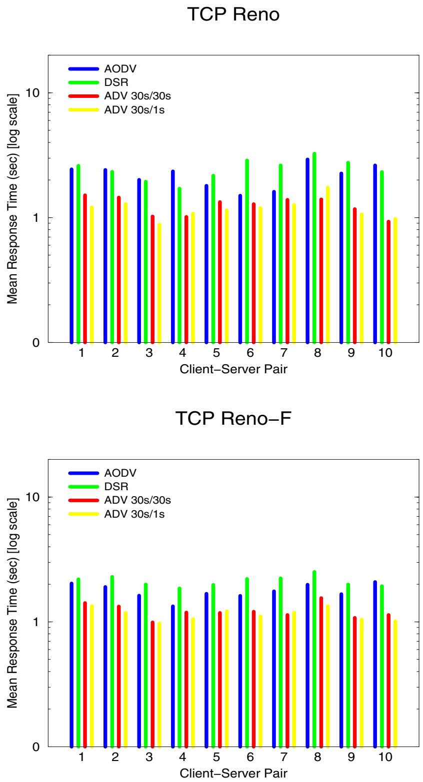


Fig. 3. Response times for 10 HTTP server-client connections using TCP Reno and TCP Reno-F.

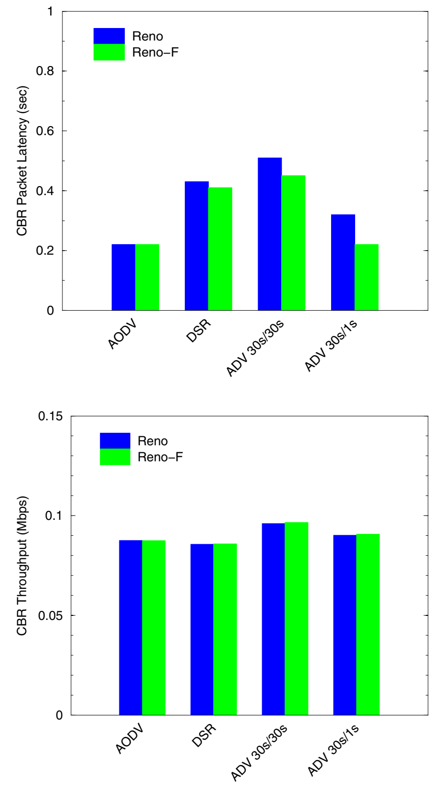


Fig. 4. CBR packet latency and throughput for 10 HTTP connections with a 100 Kbps background load from 10 CBR sources.

amount of data to transact, Reno-F, in general, minimized the duration for which TCP packets use the network. Overall, CBR packets saw less interference from TCP packets. Furthermore, with Reno, DSR and ADV had significantly higher CBR packet latencies than AODV. For these reasons, ADV and DSR benefited from Reno-F's efficient handling of TCP traffic, while AODV did not. In contrast, the CBR throughput was not improved by the reduced interference from HTTP traffic for any of the protocols, since the network was not congested and the packet delivery rates were nearly as high as they could be even without the HTTP traffic [5], [1].

Owing to shorter buffering times for UDP packets, ADV 30s/1s provided significantly lower latencies and only slightly lower throughputs compared to ADV 30s/30s.

B. VBR Traffic

We simulated 2 53-Kbps VBR flows along with 8 10-Kbps CBR flows for background traffic. The total background traffic of 80 Kbps was slightly less, but the amount of per flow is the same in the HTTP simulations and is well within the network capacity. A VBR flow, however, injects five times as much traffic, which is likely to overwhelm the network and congest the paths taken by the VBR packets.

Figure 5 presents the results of our simulations. Let us look at the packet latencies. The CBR packet latencies for the routing protocols were about the same as those observed in the HTTP-traffic simulations (see Figure 4). For the VBR traffic, AODV yielded the lowest latencies, about 57% less than DSR and ADV 30s/30s.

Now, let us look at packet delivery fractions. AODV and DSR drop more than half of VBR packets, but only about a fifth of CBR packets. On the other hand, ADV 30s/30s delivers 75% of VBR packets and over 90% of CBR packets. Overall, ADV delivered 58-65% more VBR packets than DSR and AODV. The low VBR packet delivery rates for AODV and DSR are due to the high volume of traffic produced by each VBR connection. Referring to Figure 1, even with the performance boost of Reno-F for HTTP traffic, neither on-demand protocol achieved an average per-connection throughput as high as 40 Kbps. The 53-Kbps of traffic produced by each VBR connection is a higher load than these protocols can sustain. To verify this, we reran the simulations with each VBR producing traffic at an average rate of 10.6 Kbps (about the same as the traffic load produced by a CBR flow). Figure 6 shows that, at this lower load, all three routing protocols yielded high packet delivery fractions. DSR gave packet latencies similar to ADV and throughput similar to AODV. Given that DSR is fairly competitive with AODV in CBR traffic simulations [2], [5],

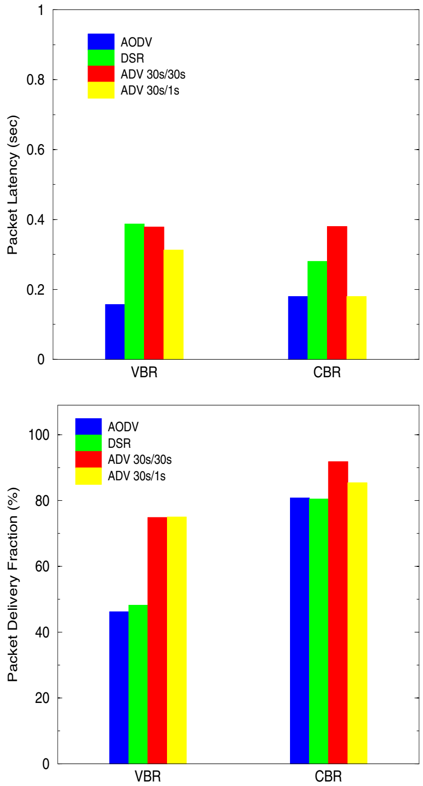


Fig. 5. Packet latency and delivery fraction for 2 53-Kbps VBR connections, and a 80-Kbps background load from 8 CBR sources.

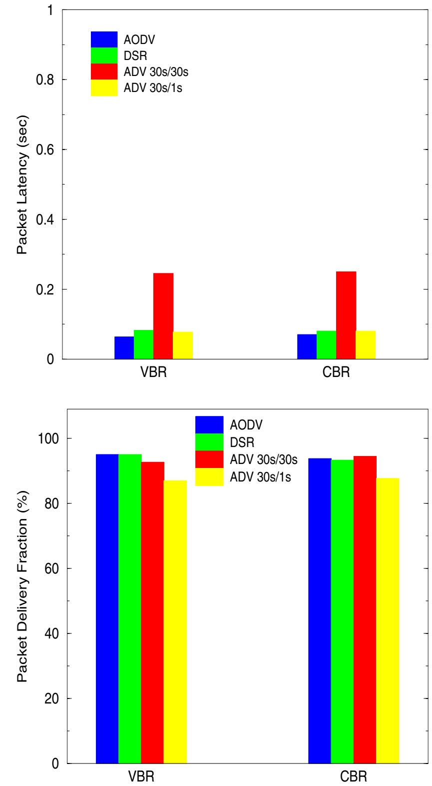


Fig. 6. Packet latency and delivery fraction for 10.6-Kbps VBR connections, and a 80-Kbps background load from 8 CBR sources.

[1], we suspect that turning off the route cache hurt its performance when the network was congested.

Comparing ADV 30s/30s and 30s/1s cases, we note the following. In situations where a connection produces extremely high traffic, for example, VBR connections in Figure 5, using shorter buffering time provides lower latencies with no measurable impact on throughput. In fact it can be argued that it is advantageous to not to buffer packets in such situations, since there are enough packets injected to use any available bandwidth. For connections that produce a low volume of traffic and do not congest the network, for example, CBR connections in Figure 5 and VBR and CBR connections in Figure 6, shorter buffering times can reduce packet latencies significantly with lower, about 10% in the results above, throughputs. A histogram analysis of packet latencies for ADV 30s/30s indicated that about 10% of delivered packets with the largest latencies increased the overall average packet latencies by a factor of 2 or more. This indicates that packet buffering time may be used as a tunable parameter to trade latency for throughput.

A MANET subjected to heavy loads from VBR flows is exactly the type of situation in which ADV excels, while the on demand algorithms AODV and DSR falter. This simulation clearly shows the detrimental impact of not maintaining routes pro-actively for high rate flows.

C. Mixed HTTP and VBR Traffic

In this set of simulations, we added the 10 HTTP flows used earlier to the 2 53-Kbps VBR and 8 10-Kbps CBR flows used above. The purpose of these simulations was to investigate the impact of TCP and VBR traffic on each other. Compared to the background traffic used in earlier HTTP simulations, the two VBR flows can cause severe disruptions to TCP traffic.

Figure 7 shows the HTTP service times and throughputs observed for each routing protocol. Compared to the results shown in Figure 1, service times were about 35% higher and throughputs were about 45-55% lower. The relative rankings of the protocols in terms of performance were unchanged, as was the effect of Reno-F. Compared to AODV and DSR, each variant of ADV provided 16-27% lower service time and 80-100% higher throughput.

Figure 8 shows the VBR packet latency and jitter observed for each routing protocol. Jitter is defined here as the standard deviation of the packet latency. Packet latencies have skewed distributions with long tails, so the standard deviation can be larger than the mean. VBR packet latencies were higher in combination with HTTP traffic than without (see Figure 5). The increase in VBR latency was as little as 40% for DSR and as much as 100% for ADV

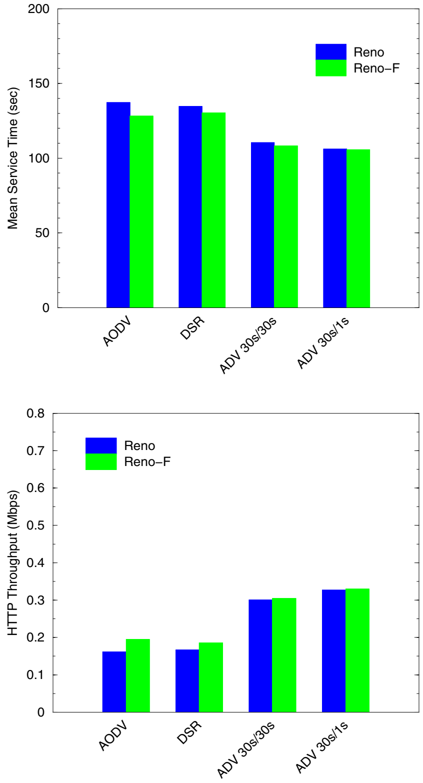


Fig. 7. Mean service time and HTTP throughput for the traffic consisting of 10 HTTP, 2 53-Kbps VBR connections, and a 80-Kbps background load from 8 CBR sources.

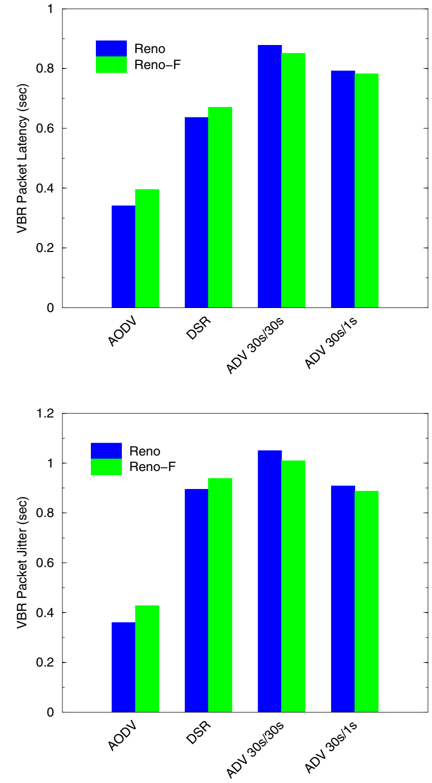


Fig. 8. VBR packet latency and jitter for the traffic consisting of 10 HTTP, 2 53-Kbps VBR connections, and a 80-Kbps background load from 8 CBR sources.

and AODV. AODV had the lowest packet latencies, about 55-75% lower than those given by DSR and ADV. Also, AODV provided significantly lower packet jitter than DSR and ADV. ADV 30s/30s had the worst packet latencies and jitter, while the 30s/1s version had lower latencies with about the same VBR throughput.

Figure 9 shows the VBR packet delivery fraction and throughput observed for each routing protocol. Compared to the results shown in Figure 5, all three protocols delivered 1/4th to 1/3rd fewer VBR packets. Once again, ADV outperformed the other two by at giving 33-50% higher throughput. ADV delivers more packets, even those taking 4 or more hops, while AODV and DSR drop most of those packets. Combining the VBR throughputs in Figure 5 with the HTTP throughputs in Figure 7, we find that the total throughput is significantly less than the throughput achieved by 10 HTTP connections alone in Figure 1. For ADV, the reduction in total throughput is around 200 Kbps.

It is interesting to examine the impact of Reno-F on VBR traffic. For AODV and DSR, VBR packet latencies and throughput were adversely impacted, while for ADV, they were improved, much the same way CBR traffic performance was improved by Reno-F in HTTP simulations (see Figure 4). With AODV and DSR, 53-Kbps VBR flows caused network congestion, and Reno-F improved throughput of HTTP traffic at the expense of VBR traffic. ADV, however, did not suffer as much from the heavy traffic flows. So reducing the duration for which HTTP transactions appear on the network still seems to be beneficial to VBR traffic.

Additional simulations involving 4 VBR and 10 HTTP flows with no background CBR flows are given in Figures 10 through 12. HTTP flows suffer significantly from VBR flows. Reno-F helps AODV and DSR's performances significantly. Among the three routing algorithms, AODV and DSR perform much worse than ADV for both HTTP and VBR flows.

These simulations indicate clearly that MANETs may not be able to support interactive video and audio flows satisfactorily, when there is competing HTTP and other background traffic. However, adaptive transport protocols (for example, a datagram protocol with rate control) for MANETs should be able to support non-realtime streaming flows with buffering successfully.

V. Related Work

Several mechanisms have been proposed for improving TCP performance in MANETs. Each of the studies cited

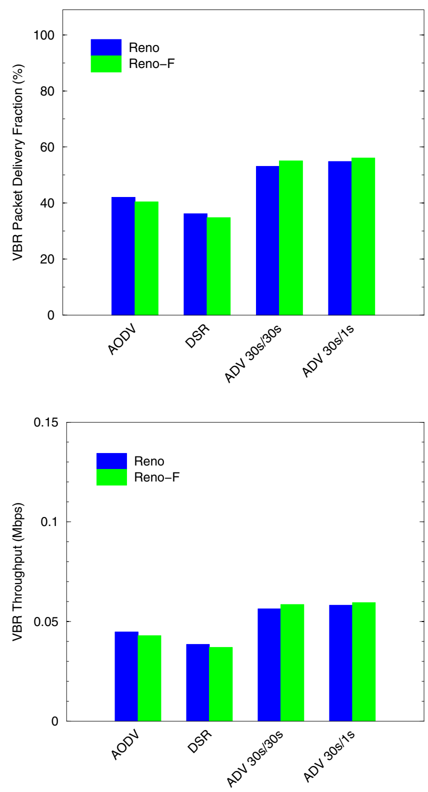


Fig. 9. VBR packet delivery fraction and throughput for the traffic consisting of 10 HTTP, 2 53-Kbps VBR connections, and a 80-Kbps background load from 8 CBR sources.

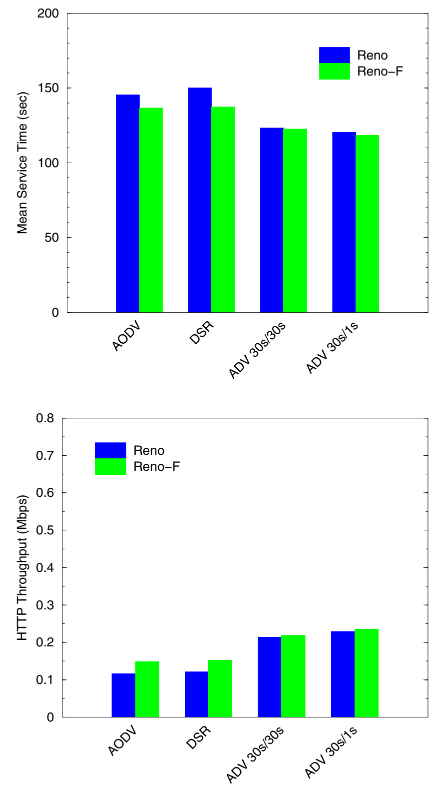


Fig. 10. Mean service time and throughput for the traffic consisting of 10 HTTP and 4 53-Kbps VBR connections with no background load from CBR sources.

below utilizes simulation or emulation to demonstrate the performance benefits of the proposed technique. In each case, only a single TCP connection is considered, TCP traffic is limited to file transfers, and no background network load is included.

TCP-F [3] uses a feedback scheme in which an intermediate node, upon detecting the disruption of a route due to the mobility of the next host along that route, sends a Route Failure Notification (RFN) to the TCP sender. When it receives the RFN, the sender stops transmitting packets and freezes its state, including the retransmission timeout interval and the congestion window. Eventually, an intermediate node will learn of a new route to the destination and send a Route Re-establishment Notification (RRN) to the source. After receiving the RRN, the sender restores its previous state and resumes transmission. TCP's congestion control mechanism is not invoked and the consequent performance penalty is avoided.

A similar scheme has been proposed in which an intermediate node sends an explicit link failure notification (ELFN) to the TCP sender when a route failure is detected [11]. As in TCP-F, the sender freezes its state upon receipt of the ELFN. However, instead of relying on the receipt of a further route re-establishment message, the TCP sender transmits packets at a regular interval to probe the network until the availability of a new route is detected. This mechanism is similar to the Reno-F proposal in that probing is used to learn that the route has been repaired as opposed to waiting for a notification that might get lost in the network. However, Reno-F has the advantage that its probe interval is not fixed, but rather is tied to the current estimate of the RTT and is thus adaptive to network conditions.

In the TCP-BuS proposal [15], an explicit route disconnection message (ERDN) is generated at an intermediate node upon detection of a route failure. This message is propagated to the source which then stops transmission. Packet transmission is resumed after a partial path has been re-established from the node which detected the route failure to the destination and that information is relayed to the TCP sender in an explicit route successful notification (ERSN). During the course of a TCP connection, packets are buffered at the intermediate nodes along the path from sender to receiver. Nodes upstream from the failed link are able to forward these packets on to the destination once the route has been repaired, relieving the sender from having to retransmit these packets. This scheme is somewhat complex and would seem likely to have trouble with multiple route failures in quick succession, as in a high mobility network.

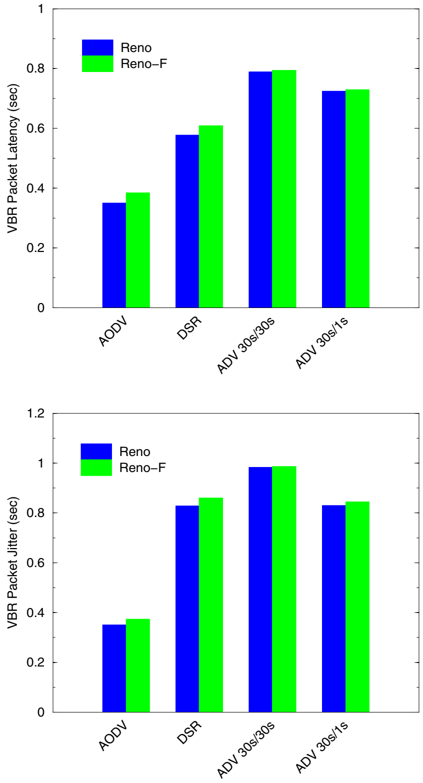


Fig. 11. VBR packet latency and jitter for the traffic consisting of 10 HTTP and 4 53-Kbps VBR connections with no background load from CBR sources.

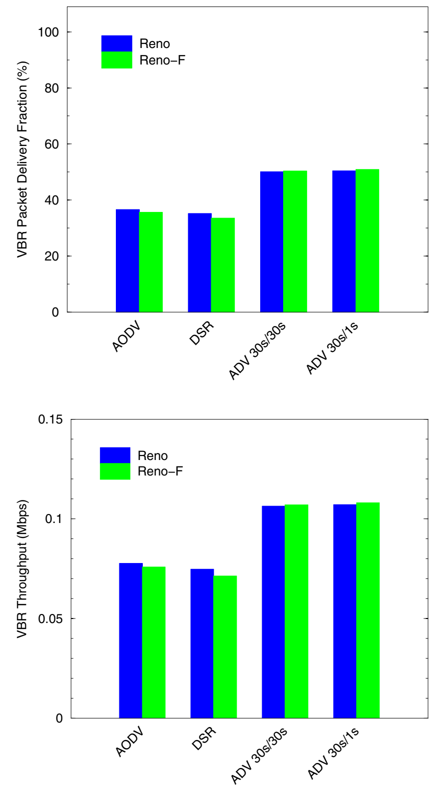


Fig. 12. VBR packet delivery fraction and throughput for the traffic consisting of 10 HTTP and 4 53-Kbps VBR connections with no background load from CBR sources.

An advantage of Reno-F is its simplicity; to the extent that comparable performance can be attained, a simple scheme is preferable to a more complicated mechanism.

In ATCP [17], a layer between TCP and the routing agent is proposed which, among other things, shields TCP from packet loss that is perceived to be non-congestion related. Upon learning of a route failure (by means of an ICMP *Destination Unreachable* message), ATCP places the TCP sender into *persist mode*, thus avoiding the invocation of congestion control measures. While in persist mode, TCP generates probe packets at exponentially increasing intervals up to a maximum of 60 seconds. Once the route is re-established and an ACK is received for one of the probe packets, TCP moves out of persist mode and resumes packet transmission. Since exponential backoff is a problem in MANETs, the fixed RTO technique may be used by ATCP in determining a more suitable probe interval.

VI. Conclusions

Since the capacity of a mobile ad hoc network (MANET) changes rapidly due to node movements and random wireless link errors, both TCP and UDP protocols perform poorly, albeit for different reasons. In this paper, we have analyzed the impact of a simple transport layer mechanism, called fixed RTO, on the performance of TCP traffic and the impact of buffering packets at intermediate nodes when routes are lost on UDP packets. We simulated a variety of traffic loads, involving HTTP and VBR multimedia traffic with an interfering background CBR traffic.

For a more complete understanding of the modifications to transport protocols, we used representative on-demand and proactive routing algorithms. Of the three routing protocols we have simulated, ADV (proactive type) performs as well or better than AODV and DSR (on-demand type) when TCP Reno is used as the transport protocol. With the fixed RTO heuristic, which prevents doubling of the retransmit time out interval for consecutive timeouts, we have shown that AODV and DSR perform much better, while ADV, though does not improve, still outperforms them.

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.

Comparing the Reno-F results for FTP traffic in [6] and those for HTTP traffic in this paper, we note that while AODV and DSR outperform ADV multiple for FTP flows, ADV is markedly superior to the other two for multiple HTTP connections when there is interfering non-TCP friendly traffic. The primary reason is that the FTP traffic has infinite backlog with no idling once the connection is setup, while the HTTP flows have a finite amount of data transacted with several slow start periods. This sort of situation favors a routing protocol that can supply routes quickly. So maintaining routes pro-actively like ADV does is beneficial in such situations. Our results show that handling finite-sized TCP transactions efficiently using the fixed RTO heuristic can lead to better performance for UDP traffic, as long as the UDP traffic does not congest the network.

Another major result of our study is that current MANETs may have difficulty in sustaining interactive audio and video flows. We evaluated the capability of MANETs to support congestion-causing UDP traffic using 2-4 53-Kbps VBR flows. The traffic rates of 53-Kbps per VBR is at the low-end of low quality video and high-fidelity audio rates. This is the type of traffic that favors routing protocols that maintain active routes efficiently. Once again we show that on-demand routing protocols that depend on elaborate route request and reply mechanisms do not perform well. On the other hand, the proactive ADV performs well, though it has high packet latencies.

Among the three routing protocols, ADV offers superior performance for HTTP traffic both in terms of service times and throughputs. For VBR traffic, AODV offers the best latencies at the expense of very low delivery rates, while ADV offers very high delivery rates, but also high latencies. Our results show that reducing buffering of packets at intermediate nodes improves ADV latencies significantly, but also lowers delivery rates by about 10%. DSR with route replies from cache turned off seems to perform as well as AODV for HTTP traffic, but not for VBR traffic.

Our simulations involving mixed HTTP and VBR traffic show that the impact of VBR on TCP traffic quite significant. So it is necessary to use more TCP-friendly datagram protocols for multimedia flows.

REFERENCES

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

[2] J. Broch et al., " A performance comparison of multi-hop wireless ad hoc network routing protocols" in *ACM Mobicom '98*, 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*, 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] S. R. Das, C. E. Perkins, and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *IEEE Infocom 2000*, Mar. 2000.

[6] T. Dyer and R. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks" *MobiHoc '01*.

[7] 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.

[8] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SigComm*, 1994.

[9] M. Gerla et al., "TCP Performance in Wireless Multihop Networks" IEEE WMCSA '99.

[10] 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.

[11] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *ACM Mobicom '99*.

[12] Y. Hu and D. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," in ACM Mobicom 00, Aug. 2000.

[13] IEEE Computer Society LAN/MAN Standards Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications." IEEE Std. 802.11-1997. IEEE, New York, NY 1997.

[14] 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.

[15] D. Kim et al., "TCP-BuS: Improving TCP performance in wireless ad hoc networks" in *IEEE ICC 2000*.

[16] C. Liu and R. Jain, "Using Congestion Coherence to Enhance TCP over Wireless Links," manuscript, June 2001.

[17] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks" *IEEE J-SAC*, vol. 19, no. 7, pp. 1300–1315, Jul. 2001.

[18] 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.