TCP Performance over Multipath Routing in Mobile Ad Hoc Networks

Haejung Lim
Telecommunication Network Division
Samsung Electronics, Seoul, Korea
Email: haejung93.lim@samsung.com

Kaixin Xu, Mario Gerla
Computer Science Department, UCLA
Los Angeles, CA 90095, USA
Email: {xkx, gerla}@cs.ucla.edu

***Abstract*— In this paper, we investigate TCP performance over a multipath routing protocol. Multipath routing can improve the path availability in mobile environment. Thus, it has a great potential to improve TCP performance in ad hoc networks under mobility. Previous research on multipath routing mostly used UDP traffic for performance evaluation. When TCP is used, we find that most times, using multiple paths simultaneously may actually degrade TCP performance. This is partly due to frequent out-of-order packet delivery via different paths. We then test another multipath routing strategy called backup path routing. Under the backup path routing scheme, TCP is able to gain improvements against mobility. We then further study related issues of backup path routing which can affect TCP performance. Some important discoveries are reported in the paper and simulation results show that by careful selection of the multipath routing strategies, we can improve TCP performance by more than 30% even under very high mobility.**

I. Introduction

In mobile ad hoc networks (MANET) [1], TCP performance is not as stable as in wired networks. Various wireless ad hoc net characteristics not found in wired networks - node mobility, unpredictable radio medium, external interference/jamming and multiple access contention - create a host of problems not all of which can been satisfactorily solved. Among them, frequent link breakages due to mobility are one of the major factors degrading TCP performance. When link breakage happens, TCP sender will encounter continuous packet losses over an extended period. Its congestion window is reduced and the TCP retransmission timeout (RTO) becomes progressively larger leading to high restart latency and very poor efficiency, especially in high mobility (eg, ground or airborne vehicles). Moreover, during the period of link breakage, the more packets sent out by the TCP sender, the more network resource is wasted. Several researchers have studied this problem. A typical solution is to detect the link failures and freeze the TCP state (including congestion window size and RTO interval) until a new path is re-established. Example schemes include Explicit Link Failure Notification (ELFN) [2] and TCP-F [3]. Both ELFN and TCP-F rely on the intermediate nodes to report the link breakage. In [4], a scheme called Fixed-RTO is presented to prevent TCP from excessive RTO backoff. The key idea is the following: when consecutive retransmission timeouts happen for the same data packet, just double the RTO the first timeout and keep the same value for the subsequent timeouts. A detailed review of related work is given in section II.

All the schemes mentioned above are targeting at preventing TCP from wrongly reacting to packet losses caused by link failures (e.g. by freezing TCP states). However, if link failures happen frequently, TCP will still suffer significant performance degradation even when the above schemes are applied since TCP sender may go to the frozen state repeatedly and simply wait for route reestablishment without sending any new data packets. To overcome this ”impasse”, another solution is to improve the path availability using multipath routing. The conventional ad hoc routing protocols usually only keep a single path to each destination. Multipath routing instead maintains several paths to the same destination simultaneously. Thus, the probability that there is no path from the TCP sender to the receiver is effectively reduced. Many multipath routing protocols indeed have been proposed in the literature to improve the network performance [5], [6], [7], [8], [9]. However, most of them only evaluate performance with UDP traffic. So far, from the best of our knowledge, no detailed investigation of TCP performance over multipath routing protocols has been reported in the literature. However, running TCP efficiently over multipath routing is not as straightforward as with UDP. One problem is that average round trip time (RTT) estimation is not accurate under multipath routing. Namely, the average RTT over several paths may be much shorter than the max RTT (on the longest path). Thus, TCP sender may prematurely timeout packets which happen to take the longest path. Moreover, packets going through different paths may arrive at the destination out of order and trigger duplicate ACKs, which in turn may trigger unnecessary TCP congestion window reductions. Indeed, from our experiments, we found that using multiple paths simultaneously will actually degrade TCP performance in most cases (see section IV). Thus, it is important to investigate TCP performance over multipath routing to understand if and when there are gains.

In this paper, we study TCP performance with an on-demand multipath ad hoc routing protocol named Split Multipath Routing (SMR) [5], which is built on top of the Dynamic Source Routing (DSR) [10] protocol. We firstly investigate TCP performance while multiple paths are utilized

This work is supported in part by ONR ”MINUTEMAN” project under contract N00014-01-C-0016 and TRW under a Graduate Student Fellowship.

concurrently. Simulation results clearly show that this pure multipath routing is detrimental to TCP performance. Then, we switch to another multipath routing strategy, backup path routing. Backup path routing actually uses only one path at a time but it maintains some backup paths and can switch from current path to another alternative path rapidly if current path fails. We evaluate TCP performance over backup path routing through intensive simulation experiments and then further study the issues related to backup routing, which is shown to be very important to TCP performance.

The rest of the paper is organized as following. We briefly review some related work in section II and describe the simulation environment as well as some important protocols we used in our experiments in section III. In section IV, we present our observation on TCP performance while multiple paths are used concurrently. In section V, we introduce the backup path routing and discuss some issues critical to TCP performance. The detailed investigation of TCP over backup path routing is presented in section VI and we conclude the paper in section VII.

II. Related Work

Recently, TCP performance in ad hoc wireless networks has become an active research field. Link failures due to mobility have been identified as one of the major factors degrading TCP performance . To combat this problem, Holland and Vaidya et al proposed Explicit Link Failure Notification (ELFN) scheme whereby the intermediate nodes notify the TCP sender when a link failure happens [2]. This is a scheme similar to the Explicit Congestion Notification (ECN) technique originally proposed in the wired networks. With the help of ELFN, TCP senders can tell whether a packet loss is caused by link breakage or congestion. Thus, it could properly response to different kinds of packet losses. In TCP-F (TCP-Feedback) [3], Chandran and Prakash et al proposed a scheme, very similar to the ELFN scheme, by asking the intermediate node to notify TCP sender about the network condition. When one intermediate node detects a route failure, it explicitly sends a route failure notification (RFN) to the TCP sender. The difference between TCP-F and ELFN is the response of route failures. TCP-F relies on the intermediate node to send a route reestablishment notification (RRN) to notify that the path is back up. In ELFN, the TCP sender must send probing packet periodically to detect the route recovery.

Another more serious problem that link failures may cause to TCP performance is unnecessary exponential backoff of the retransmission timeout (RTO) interval. In the conventional TCP protocol, when a retransmission timeout happens, TCP sender retransmits the lost packet and doubles the RTO. This procedure is repeated until the lost packet is acknowledged. Such an exponential backoff of the RTO helps TCP react to congestion gracefully. However, when link failure happens, TCP tends to increase the RTO rapidly even there is no congestion. Wrongly applied exponential backoff significantly degrades the TCP performance since in the ad hoc wireless networks, the TCP congestion window size is usually small and the RTO plays an important role. In [4], Dyer and Boppana et al proposed a mechanism called fixed-RTO to repair this problem. When the retransmission timeout happens consecutively, the authors think it is mainly due to route break, not congestion. Thus, after retransmitting the lost packet, fixed-RTO will freeze the RTO value utile the route is reestablished. Through simulation experiments, the authors show that with fixed-RTO, TCP can achieve throughput comparable to that of the ELFN mechanism. However, ELFN requires support from the intermediate nodes, while fixed-RTO is pure end-to-end mechanism.

The above related work mostly focuses on letting TCP detect route failures and react to them in a proper way. In contrast, , in this paper, we focus on how to improve the path availability to TCP connections under high mobility, namely by using multipath routing. Thus our work is complement to these previous work and can be combined together to help TCP achieve good performance.

III. Simulation Environment and Protocol Models

A. Split Multipath Routing

In our study, we chose the Split Multipath Routing (SMR) [5] protocol as the underlying multipath routing protocol. It is basically an extension to DSR [10]. Its basic route discovery procedure is quite similar to DSR. When the source needs a route to the destination but no route information in hand, it floods the RREQ message to the entire network. Only the destination nodes are allowed to send back the RREP packets. No cached entries are exploited. To discover multiple paths, the destination will return several RREP packets, one for each different path. The criteria for selecting multiple paths here is to build maximally disjoint paths for minimizing contention and local congestion. In detail, a destination will return the RREP packet to the first RREQ packet immediately. For later RREQs coming from different paths, it selects the maximally disjoint route and sends RREP packets back to the source via the chosen route. SMR has two ways to response to route failures. In the first scheme, it performs the route discovery when any route to the destination is invalidated. Another alternative way is to only perform the route discovery when ALL routes to the destination are invalidated. According to simulation experiments in [5], the second scheme always outperforms the first one. Thus, in this paper, we use the second scheme for responding to route failures.

B. Exponential Backoff and Fixed RTO

As we have mentioned in the related work, the exponential backoff of the RTO will degrade TCP performance significant in ad hoc wireless networks under mobility. To make our study up to date, in our work, we also adopted the fixed RTO scheme. In the rest of the paper, unless otherwise mentioned, we use the fixed RTO scheme in experiments.

C. Simulation Platform

This paper is basically a performance study based on simulations. The simulation platform used is QualNet [11] which

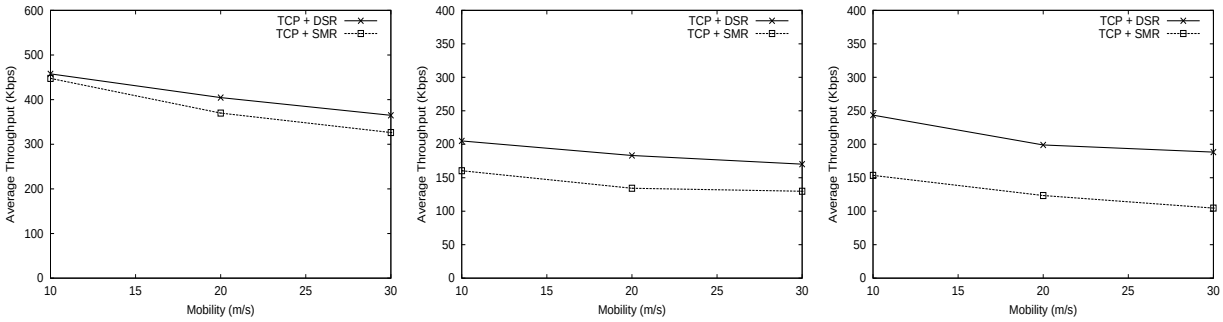

Fig. 1. *TCP performance while utilizing 2 paths simultaneously.* Left: TCP Throughput when only 1 FTP/TCP connection is used. Middle: TCP Throughput when 1 FTP/TCP and 4 CBR/UDP connections are used. Right: Average TCP throughput when 5 FTP/TCP connections are used.

is the successor of GloMoSim simulation library [12]. TCP Reno is used and we implemented both SMR routing and fixed RTO scheme. In all simulation experiments, unless explicitly mentioned, 50 mobile nodes are placed in a 1000m by 1000m terrain. IEEE 802.11 radios are adopted and each radio has a propagation range as 250 meters and channel capacity of 2Mbps. The mobility model is the random waypoint model. To simulate a high mobility environment, mobility speed is set from 0 m/s to 30 m/s with zero pause time. We simulate different scenarios with different traffic patterns such as 1 TCP connection, 1 TCP + 4 CBR/UDP connections and 5 TCP connections. The source and destination pairs are generated randomly. The TCP Maximum Segment Size (MSS) is set to 1460 bytes and we use FTP as the application for generating TCP traffic. For each simulation scenario, 30 runs with different seeds are performed to eliminate the impact of randomness. The presented results are the average of the 30 runs.

IV. TCP Using Multipath Concurrently

The first case we tested is to let TCP use multiple paths simultaneously. The multipath routing protocol, SMR, scatters TCP packets evenly on the multiple paths. For simplicity, SMR only keeps two paths at the same time. We compare TCP performance on top of SMR and the original DSR. In order to compare DSR and SMR as fairly as possible, we adjust DSR's optimization thus, except the fact that SMR uses two paths simultaneously, the route discovery and control mechanisms of both protocols are almost the same. Experiment results under various traffic patterns are plotted in Fig. 1.

From Fig. 1, we observe that when TCP is using multiple paths, it always behaves worse than using only single path in all the investigated scenarios. This is surprisingly different from the results given in [5], where UDP based CBR traffic achieve good performance improvements over SMR. With careful analysis of the trace file, we found that the negative results of TCP traffic is due to the fact that TCP reacts to RTT and other network parameters very sensitively. While utilizing multiple paths concurrently, the average RTT measured by TCP sender is not accurate. Thus, more premature timeouts may happen. The more serious problem is out-of-order packet delivery via different paths, which will trigger

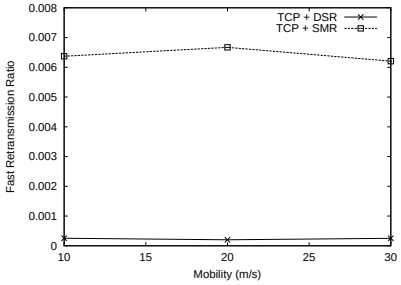

Fig. 2. The packet retransmission ratio of TCP under SMR and DSR

duplicated ACKs, which in turn triggers TCP congestion control scheme (e.g. reducing the congestion window size to half when 3 duplicated ACKs arrive).We then plotted the packet retransmission ratio of TCP flows in the experiments. The packet retransmission ratio is defined as [# of packet retransmissions / total # of packet transmissions], where each retransmission of the same packet is counted as a separate transmission. A typical figure is shown in Fig. 2. Clearly the packet retransmission ratio of TCP over SMR is much higher than that of DSR. The high packet retransmission ratio prevents the TCP congestion window from growing large enough to achieve high throughput. To be precise, in few isolated experiments, we indeed observe that multiple paths alleviate route failures. However, the overall throughput always consistently implies that simple use of multiple paths doesn't provide prominent benefit to TCP performance. Instead, it degrades TCP performance in some degree.

V. TCP Using Backup Path

In this section, we investigate an alternative way of utilizing multipath routing. That is, to let TCP only use one path at a time and keep the other paths as backup routes, which we refer as backup path multipath routing. Backup path multipath routing still maintains several paths from a source to a destination. However, it only uses one path at any time. When current path breaks, it can quickly switch to other alternative paths. With this backup path routing, we are facing two important issues regarding TCP performance on top. The first one is how many backup paths are optimal. The second

one is how to select the primary and alternative paths. We will study them in detail in the next sections.

A. Optimal Number of Backup Paths

In a highly mobile environment, the routes replied to the same Route Request are likely to be invalid after one of them gets stale because of the mobility. Thus, although many routes for the same destination may be stored in the cache, this doesn't provide prominent advantage. When too many paths are stored in the cache, TCP sender actually tends to waste much time in trying stale routes one by one and such useless effort eventually results in low TCP throughput. In this paper, we are not trying to derive the optimal number of backup paths using theoretical arguments. Instead, we did a lot of simulations trying to experimentally determine this optimal number. From our extensive simulation experiments, we observed that two paths for each destination usually show the best TCP performance. Thus, in rest of our simulations, we only store one primary path and one alternate path for each requested destination. Due to page limitation, we didn't plot figures here.

B. Strategies of Selecting Primary and Alternative Paths

Another important issue involved would be how to select the primary as well as the alternative paths. Generally, the routes can be selected by either the source or destination since our backup path routing is a sort of source routing. If the destination replies to all copies of a Route Request, the source could select two of them for caching. However this approach would consume a lot of network resources since many paths will be discarded at the source. Thus, selectively replying two route replies at the destination is strongly preferred. There are many criteria for the destination node to choose the good paths. Here, we mention three criteria: the shortest-hop path, the shortest-delay path and the maximally disjoint path. The shortest-hop path means the number of hops from the source to the destination is the smallest. The shortest-delay path refers to the path from which the destination receives the first RREQ packet. The maximally disjoint path is for selecting the secondary path after the primary path has been decided. That is the secondary path is the one which has the fewest overlapped intermediate nodes with the primary path. These three criteria can be combined together for various schemes in terms of selecting the primary and the secondary paths.

1) Scheme 1: Shortest-Hop Path / Shortest-Delay Path: The first scheme we propose is to select the shortest-hop path as the primary route and use the shortest-delay path as the backup. In detail, during the route discovery, the destination node replies RREP packet immediately upon receiving the first RREQ packet. This first route is the shortest-delay path. As soon as the source receives the first RREP, it temporally marks this path as a primary route and checks its outgoing buffer. If there are data packets targeting for that destination, it starts transmitting them via this shortest-delay path. After sending the first RREP packet, the destination waits certain duration of time for receiving more RREQ packets and learns all possible paths. It then selects the shortest-hop path among all possible routes (except the first one which it has replied RREP packet) and sends another RREP packet to the source via this shortest-hop route. Sometimes this secondary path may be longer than the first one. When receiving the second RREP packet, the source compares it with its current primary path and resets the primary path to the shortest-hop path and the alternate path to the other one.

2) Scheme 2: Shortest-Delay Path / Maximally Disjoint Path: In the second scheme, we intend to investigate how useful it is to select the alternate path maximally disjoint with the primary path. The first scheme doesn't consider overlapping between multiple paths. It is possible that when the primary path is removed, the alternate one might also have expired if they share the same broken link. In such case, caching alternate paths doesn't provide any advantage. In order to have a valid alternate path even upon the failure of the primary path, the second scheme tries to select a maximally disjoint alternate path. In detail, when receiving the first RREQ packet, the destination records the entire path and sends a RREP packet to the source via this path (e.g. This is the shortest-delay path). It then follows the similar steps in the first scheme except that it selects a path that is maximally disjoint with the first one, instead of the shortest-hop path. If there are more than one paths which are maximally disjoint with the first route, the one with the shortest hop distance is chosen. Unlike scheme 1, the source node in scheme 2 never exchanges the primary route when the second RREP is received. In other words, scheme 2 always uses the shortest delay path as the primary route and keeps the maximally disjoint route as the alternate route.

VI. Performance Evaluation

In this section, we evaluate TCP performance on top of the backup path multipath routing mentioned in the above section. The same simulation scenario used in section IV is adopted.

A. Improvements of TCP Throughput

Fig. 3 shows TCP throughput over backup path routing under various traffic patterns. The left figure of Fig. 3 illustrates the case where only 1 TCP connection is used. We observe that scheme 1 constantly outperforms DSR at various mobility speeds ranging from 10m/s to 30m/s, while scheme 2 shows lower throughput than the original DSR. The performance gain of scheme 1 ranges from about 23% to 30%. This is a visible improvement. The reason why scheme 2 performs worse than DSR is mainly because that scheme 2 doesn't make use of the shortest hop route. As described earlier, scheme 2 uses the shortest delay route as the primary path and upon its failure, it switches to the secondary route which is maximally disjoint to the primary route. We observed that in many cases, the shortest delay route used for the primary route is not the shortest hop route. And the maximally disjoint route also tends to be longer than other routes because of its disjointness requirement. This conclusion is confirmed by simulations using 1 TCP with 4 CBR background traffic (e.g. middle figure of Fig. 3) as well

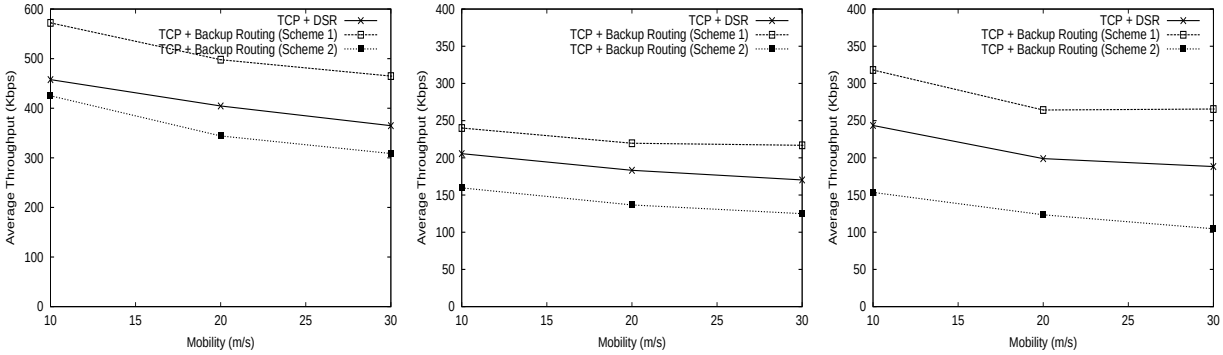

Fig. 3. *TCP performance while utilizing backup path multipath routing.* Left: TCP throughput when only 1 FTP/TCP connection is used. Middle: Average TCP throughput when 1 FTP/TCP and 4 CBR connections are used. Right: Average TCP throughput when 5 FTP/TCP connections are used.

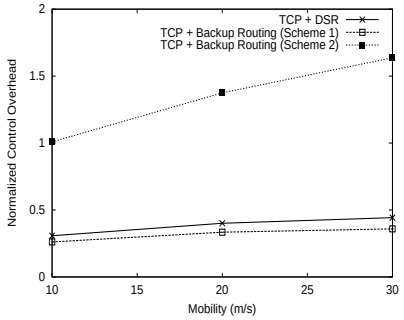

Fig. 4. Normalized Control Overhead of the Backup Path Multipath Routing with 5 FTP/TCP connections

as when 5 TCP connections are used (e.g. right figure of Fig. 3). Scheme 1 always performs best and DSR does second and scheme 2 is the worst. From these simulation results, we can conclude that backup path multipath routing with careful selection of the primary path and the alternative path is capable to improve TCP performance substantially under high mobility environments.

B. Control Overhead

In addition to investigation of throughput, in this subsection, we also investigate the control overhead of the two backup path routing schemes. Fig. 4 illustrates the normalized routing control overhead for the scenario with 5 TCP connections. Very similar results are got for other scenarios. Normalized control overhead is defined as the ratio of the number of control packets including RREQ, RREP and RERR propagated by all nodes and the number of data packets received at the destination nodes. Fig. 4 shows that scheme 1 disseminates control packets even less than the original DSR. However, scheme 2 propagates a lot of RREQ to build maximally disjoint route. Scheme 1 shows slightly fewer control overhead than DSR since it encounters fewer path failures (It only initiates new route requests when both the primary and alternative paths are broken).

VII. Conclusion

This paper is an experimental study of TCP performance over multipath routing. Its major contribution is a detailed investigation of TCP performance over various multipath routing schemes including backup path routing. It has certain importance since this is a good way to combat mobility and improve TCP performance. Moreover, unlike UDP traffic which usually gets uniformly good performance over multipath routing, TCP performance over multipath routing may show negative results as we reported in the paper. Thus, our work will prove to be of value for future investigations in this direction.

References

[1] S. Corson and J. Macker, "Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations," *RFC 2501*, Jan. 1999.

[2] G. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Proceedings of ACM MobiCom'99*, Aug. 1999.

[3] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless neworks," *IEEE Personal Communications Magazine*, vol. 8, no. 1, Feb. 2001.

[4] T. D. Dyer and R. V. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," *Proceedings of ACM MobiHoc'01*, Oct. 2001.

[5] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," *Proceedings of IEEE ICC'01*, June 2001.

[6] A. Nasipuri, R. Castaneda, and S. R. Das, "Performance of multipath routing for on-demand protocols in mobile ad hoc networks," *Mobile Networks and Applications*, vol. 6, no. 339-349, 2001.

[7] M. Marina and S. Das, "On-demand multipath distance vector routing in ad hoc networks," *Proceedings of IEEE International Conference on Network Protocols (ICNP)'01*, Nov. 2001.

[8] L. Zhang, Z. Zhao, Y. Shu, L. Wang, and O. W. Yang, "Load balancing of multipath source routing in ad hoc networks," *Proceedings of IEEE ICC'02*, Apr. 2002.

[9] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: A distance-vector multipath routing protocol," *Proceedings of IEEE INFOCOM'01*, Apr. 2001.

[10] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing, edited by T. Imielinski and H. Korth, Chapter 5*, 1996.

[11] "Qualnet simulator," *Available at http:www.qualnet.com*.

[12] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," *Proceedings of PADS'98*, May 1998.