# Multipath TCP in Lossy Wireless Environment

Jiwei Chen
UCLA Electrical Engineering Department
Los Angeles, CA 90095, USA
Email: cjw@ee.ucla.edu

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

***Abstract*— Random loss in wireless ad hoc nets derives from two main causes: lossy wireless channels and node mobility. Breakage and loss can significantly degrade the performance of network protocols, especially TCP. Recently, several schemes have been proposed to protect TCP from mobility induced link breaks by freezing TCP state when a link break is detected. TCP freeze however does not help with high channel loss due to random interference. Multipath solutions had been attempted to protect TCP from mobility, but with no success. In this paper, we address lossy channels. We repropose multipath routing to enhance TCP performance in lossy environment with heavy interference. To combat extremely high packet loss rates, we propose to duplicate each TCP packet and transmit a copy over each of the multiple paths provided by routing. Since each packet has several copies transmitted over multiple paths, the chance that all copies are lost is much reduced. Through simulation investigation, we observe that our scheme can improve TCP performance in a very lossy environment. This advantage persists even in presence of mobility.**

## I. INTRODUCTION

In ad hoc mobile networks, TCP faces various challenges, such as node mobility, hidden/exposed terminal problems, packet/ack contention on the same channel, unpredictable radio medium, external random interference(as caused for example by dynamic changes in the environment) and intentional interference, eg. jamming). A critical problem recently addressed in the literatures is frequent link breakage due to mobility. If link breakage happens, TCP cuts the window and progressively increases RTO(retransmission timeout) leading to unacceptably low throughput. A solution is to detect link failures and freeze the TCP state(including congestion window and RTO) until a new path is re-established [1], [2]. But, if link failures happen frequently, TCP still suffers significant performance degradation because TCP will go to the frozen state repeatedly and will mostly wait for discovery of a new path without sending any new data. To overcome this problem, multipath routing has also been proposed[3], [4], [5], [6], [7]. Multipath routing maintains several paths to the destination simultaneously, so the probability of delivering packets to the destination when one path fails is improved. Multipath routing also alleviates the hidden terminal problem by spreading packets on different available paths. However, most of the multipath results reported so far are based on UDP traffic because running TCP is on multiple paths presents problems of its own. One problem is that the RTT(round trip time) estimation is never accurate under multipath routing. Different delays on different paths lead to unstable RTT estimate and incorrect RTO setting. For example, if the maximum RTT on the longest path is much larger than the minimum RTT on another path, TCP could prematurely timeout packets on the longest path due to the incorrect RTT estimation. Moreover, it is likely that packets going through different paths arrive at destination out-of-order. Out-of-order packets trigger duplicate ACKs, which in turn cause unnecessary TCP congestion reaction, namely fast retransmit/recovery. A preliminary evaluation of TCP over multiple path routing [8] pointed out that using multiple paths simultaneously would actually degrade the TCP performance in most cases because of the duplicate acks.

In our paper, we wish to probe further the multipath TCP performance. Our purpose is not to seek performance enhancements by using parallel paths in static conditions or robust performance in the face of mobility. Rather, we wish to maintain robust operation in the presence of high channel loss. We assume an on demand multipath routing similar to Split Multiple Routing(SMR)[3], which is built on top of the Dynamic Source Routing(DSR)[9] protocol. Different from the work reported in [8], we use the multiple paths concurrently; namely, we transmit duplicate TCP packets on each path. The rest of paper is organized as following. We briefly review the related work in section II. The details of multipath TCP and multipath routing implementation are presented in section III. In section IV

we present our simulation results on TCP performance while multiple paths are used concurrently and compare with standard TCP. Several important issues related with multipath TCP are discussed in section V. In section VI we conclude our work.

## II. Related Work

Recently, researchers observed that link failures in an ad hoc may cause unnecessary exponential backoff of the retransmission timeout(RTO). To alleviate this problem, Backup Routing(BR) is proposed in [8]. BR maintains two paths from a source to a destination. TCP only uses one path at a time and keep the other path as backup. When the current path breaks, BR quickly switches to the alternative path. Several choices for backup path computation were evaluated, such as Shortest-Hop Path / Shortest-Delay, Shortest-Delay Path / Maximally Disjoint Path. They pointed out that some of the negative results of TCP traffic are due to the fact that TCP reacts too quickly to RTT and other network parameters. While utilizing multiple paths concurrently, the average RTT measured by TCP sender is not accurate. Thus, premature timeouts may happen. The most serious problem is out-of order packet delivery via different paths, which will trigger duplicate ACKs and TCP congestion control intervention, i.e., fast retransmit/fast recovery. In their simulations, the packet retransmission ratio of TCP over SMR is much higher than that over conventional, single path DSR. The high packet retransmission ratio prevents the TCP congestion window from growing to the level required to achieve high throughput. Although they did observe that multiple paths can alleviate the route failure incidence, they consistently found that the use of "alternate" path doesn't provide any benefits. Rather, it tends to degrades TCP performance.

An end-to-end transport layer approach called pTCP was recently proposed in [10]. This scheme effectively performs bandwidth aggregation on multi-homed mobile hosts. That paper studies the problems involved in achieving bandwidth aggregation when an application on a mobile host uses multiple interfaces simultaneously using a transport layer approach that effectively addresses the problem. The end-to-end transport layer called pTCP (parallel TCP) manages a group of TCP-v(TCP-virtual) which is a modified version of TCP. For each pTCP socket opened by an application, pTCP opens and maintains one TCP-v connection for every interface over which the connection is to be striped on. pTCP manages the send buffer across all the TCP-v connections and decouples loss recovery from congestion control, performs intelligent striping of data across the TCP-v connections, does data reallocation to handle variances in the bandwidth-delay product of the individual connections, redundantly stripes data during catastrophic periods (such as blackouts or resets), and has a well defined interface with TCP-v that allows different congestion control schemes to be used by the different TCP-v connections. Their simulations show that pTCP can provide higher throughput to the application. This scheme is possible to be applied in the ad hoc network if multiple interfaces and mediums are available. They didn't address the routing protocol used by pTCP and didn't tested pTCP in a very lossy and mobile environment. Our work differs from pTCP in that it doesn't modify TCP, nor less it invoke the transport layer. Our multipath TCP solution is complementary and it could be used "under" pTCP.

## III. TCP over Multiple Paths

TCP is vulnerable to high packet loss due to link breakage and to interference. To address this problem, current proposed methods put focus on the single path failure recovery and TCP's proper reactions to it. Multiple path routing is recognized to be helpful to for route failure resilience and load balancing, but very few experiments of TCP using multiple paths have been reported. The major deterrence were the adverse effect of out-of-order packets on TCP, and the increased congestion caused by the use of more resources (ie. two paths instead of one). In this paper, we wish to further investigate these claims and in fact wish to determine under what mobility and error loss conditions aggressive multipath TCP can be beneficial.

### *A. Multipath Routing*

We have modified DSR routing protocol to support multipath routing protocol. The basic route discovery procedure is similar to Split Multiple Routing(SMR)[3]. It is a multipath extension to DSR [9]. When the source needs a route to the destination and has no cached route, it floods the route request(RREQ) message to the entire network. Only the destination is allowed to send back the route reply(RREP). To discover multiple paths, the destination will return several RREP packets, one for each completely disjoint path with the first RREP. We adopt a different RREQ forwarding scheme from that proposed in the original DSR. Instead of dropping all duplicate RREQs, intermediate nodes will always forward the duplicate RREQs that traverse fewer hops. This approach is slightly different with [3]. In [3], the

node forwards the duplicate RREQ that arrives from a different link than the first RREQ is received, and whose hop count is not larger than that of the first received RREQ. However, it would not help in constructing complete disjoint paths since the routes in all forwarded RREQs by this node will overlap at the node itself. It turns out that the number of RREQs during the route discovery will be much higher than the original DSR, and their simulation shows problems with routing overhead. Our method will incur less overhead by only forwarding route requests which have a *shorter* path.

If multiple paths are used concurrently, even if they are completely disjoint, they still could interfere with each other if they are within interference range of each other's transmissions. To minimize contention and local congestion during data phase, one must assure that the paths are separated by at least two hops. To this end we assume that each node maintains neighbor information and insert the neighbor information in the RREP. This requires a simple modification to the DSR code, and some extra space in the route reply. As for the neighbor information, this is gathered in a passive way, by listening to neighbors.

When the sender gets the replies, it will build a partial map of the topology that interconnects source and destination. The criterion for selecting multiple paths is to find completely disjoint paths, so that the intermediate nodes in one path are two-hop disjoint with those in the other path(ie, two-hop disjoint paths).

In detail, the source will initiate route discovery when no route is available. The intermediate nodes only forward the route requests which have a shorter path. The destination will return the RREP packet to the source immediately when the first RREQ packet arrives. For later arrived RREQs, a RREP packet is sent back to the source only if the route is disjoint from that reported in the first RREP. On the way back, each intermediate node will attach its own neighbor information into the RREP and forward it. When the sender receives the RREP, it will build the topology map from the RREP and it will try to generate route two-hop disjoint from the first one. If more than one such route exists, the one with the shortest hop distance is chosen. If multiple routes meet the shortest hop condition, the most recent route is chosen. This way, nodes on different paths are not in transmission range of each other's and the contention is alleviated. However, there can still be interference since the sensing and interference ranges are higher than the transmission range. The complete solution to this problem is a topic for future research.

### *B. MultiPath TCP(M-TCP)*

Now that multipath routing provides multiple paths, the next question is how to use exploit these multiple paths. One approach is use only one path, such as backup routing in [8], in which TCP uses primary path all the time until it fails and TCP switches to backup path. Another approach is to use the multiple paths concurrently, but it could have serious out-of-order problem as reported in [8]. In a highly lossy and mobile network, we found that using a single path is not enough, even when a backup route on standby. Suppose the primary path fails, the lost packet is retransmitted on the backup path only after TCP detects the loss after 3 duplicate acks or RTO. Moreover, the backup path could have become obsolete due to mobility. As a consequence, TCP performance may still be severely degraded. In our solution, instead of using the backup only when the primary fails, we choose to send copies of same packet on both paths to increase redundancy and thus probability of success.

One detail is where to duplicate the TCP packet. When TCP sends a single packet out, MR will stamp the source route in each packet header and duplicate packet if alternate path is available. We didn't modify TCP to send duplicate data packet, it is the MR routing layer at the sender that duplicates the TCP data packets. Our method is a routing layer approach and it is consistent with our philosophy not to change TCP. Moreover, TCP doesn't have the knowledge of alternate paths. The network layer also will add delay jitter between duplicate packets (albeit they travel on disjoint paths) to minimize their contention around the sender. In the next section, we compare multipath TCP performance with standard TCP. The results show that multipath TCP can achieve better performance in very lossy conditions. Because we duplicate packets on each path, the out-of-order problem reported in [8] dramatically reduced. However, another problem is the increase in duplicate acks because of duplication. We discuss this problem and its solutions in the next section.

We compare our multiple path performance with standard TCP over DSR. Our initial results show it can achieve more throughput under some degree of packet loss rate. Because we duplicate packet on each path, the out-of-order problem reported in [8] is not obvious. However, we still has duplicate ack problem caused by our packet duplication. We explain this phenomena in the next section.

## IV. Performance Evaluation

We implemented the M-TCP in the NS2[11] simulator, and evaluated the performance of the following protocols:

1. Standard TCP over original DSR(O-TCP)
2. Standard TCP over multipath DSR(M-TCP)

We have two simulation scenarios, one is the static network without mobility, another is dynamic network with mobility. In both cases we assume there is "random" channel interference. This interference is distinct from packet collisions caused by competing traffic streams. It corresponds to temporary obstruction of the path caused for example by people traffic(indoor scenarios) or vehicle traffic(outdoor scenarios). Also to this category belongs the radio interference from external sources such as microwave ovens, cordless phones, Bluetooth transmitters and (in hostile environments) jamming sources. We collectively model the environment interference with random packet drops on links, ie, each time a packet is transmitted, it suffers a constant loss probability. For simplicity we assume independent losses from link to link. This assumption is valid for losses due to obstructions and local interference. The severity of external interference clearly depends on the radio configuration (antennas, modulation and coding scheme). Commercial 802.11b radios are very vulnerable while, at the other extreme, MIMO OFDM radios can effectively combat interference by a combination of beamforming and diversity strategies. In our experiments, we assume IEEE 802.11b radios using the Distributed Coordinate Function(DCF) MAC protocol. Thus, high random loss rates are expected, as reported in [12]. The size of TCP packet is set to 500 bytes. The default carrier sensing range is 550m and transmission range is 250m in NS2.

### *A. Static Network*

If no multiple paths are available, M-TCP is exactly same with O-TCP. We consider the topology in which at least one alternate path is available. In Fig.1, it is a "ring" network to illustrate how M-TCP works. It consists of 14 nodes, with a source and a sink and two paths of 8 nodes each. The distance between two nodes is 200m. First we compare M-TCP with the original TCP in absence of random loss at nodes. Fig.2 shows the sequence number acked by TCP. Here M-TCP cannot ourperform O-TCP. To understand this, suppose transmission range is equal to interference range, recall that the sender duplicates a packet and puts it on one path and, after two transmission time, on the other path. Thus, packets are fed to each path at an interval of 4 transmission times(as opposed to 3 tx times in a single path scenario). Additional allowance is required also for the returning ACKs. In more realistic topologies even a higher degradation is expected because of potential interference between the paths. Even if the paths are disjoint, they still can interfere. This is reflected in the NS2 simulator where the carrier sensing range is more than two times of transmission range. In this "ring" topology, the packets fed to one path are delayed by interference on the other path. More exactly, in Fig.1, packets are fed to each path at an interval of 6 transmission times(as opposed to 5 tx times in a single path scenario) without considering acks. In balance, about 17 percent degradation is predicted as confirmed in Fig.2.

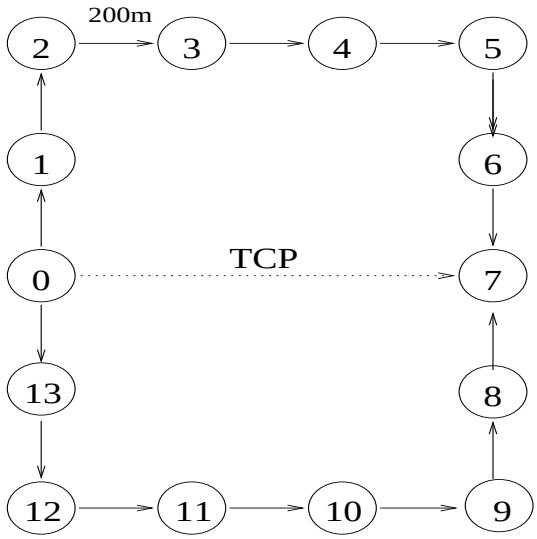



Fig. 1. Ring Topology with Two Paths

Another problem that degrades M-TCP is the increase of duplicate ACKs caused by duplicate packets. The receiver must ack each packet even if it is a duplicate because it cannot tell if the duplicate is a retransmission after timeout from the source(in which case it must be sent); or it is a duplicate coming from the alternate path(in which case it should be dropped). There are actually simple techniques[13] that allow selective duplicate ACK drops, but we have not implemented them here in order to keep TCP receiver unchanged. The extra duplicate acks could easily trigger more retransmit/recovery episodes because TCP will enter fast retransmit/recovery whenever it receives 3 duplicate acks.

However, if we scatter the packets on two same paths, the sender could send faster onto two paths than using one path and thus increased the capacity. In Fig.3, M-TCP equally scatters packets on two paths and the

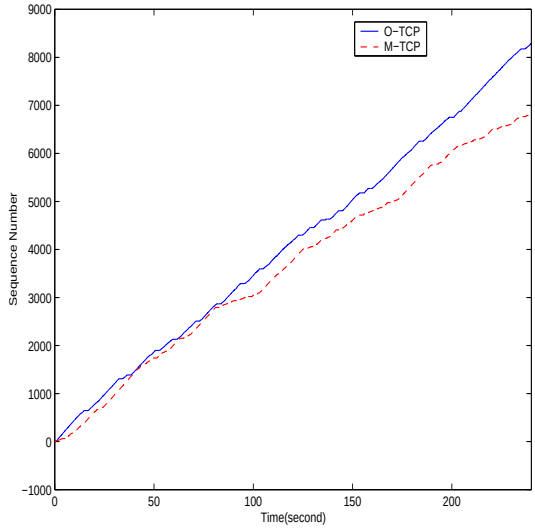



Fig. 2. Sequence Number Acked in Ring Topology, No Random Loss

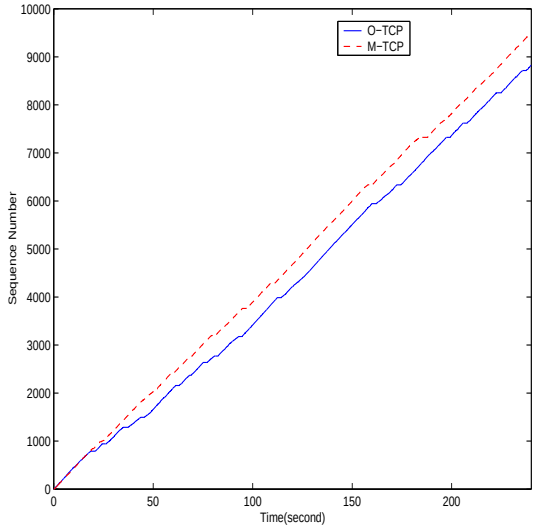



Fig. 3. Sequence Number Acked in Ring Topology by Scattering Packets, No Random Loss

performance of M-TCP is better than O-TCP. Scattering packets on multiple paths will increase the capacity and decrease the congestion than using one single path, so M-TCP can outperform O-TCP. On the other hand, serious out-of-order problem could happen by using two path concurrently[8], mostly because the RTTs on two paths differ much and the packets on different paths arrive at the receiver out of sequence. It reminds us that using two similar separated paths simultaneously could potentially increase TCP performance.

Next, we introduce packet loss. The loss is uniformly distributed on links over the whole network. Original TCP cannot perform well because high packet loss rates result in frequent timeouts. However, the probability of a loss to happen simultaneously on paths separated by, say, at least 250m is small. So, duplicated packet have a much better chance of survival and can yield significant advantage. Fig.4 corresponds to the static ring topology with 5% error rate. The M-TCP throughput improvement is significant, about 5 fold up to t=70s second in total throughput. Fig.4(b) shows the M-TCP doesn't stay in timeout state as often as O-TCP. Moreover, one notices that O-TCP "dies" at t=80sec, after exceeding a max timeout threshold. M-TCP instead survives.

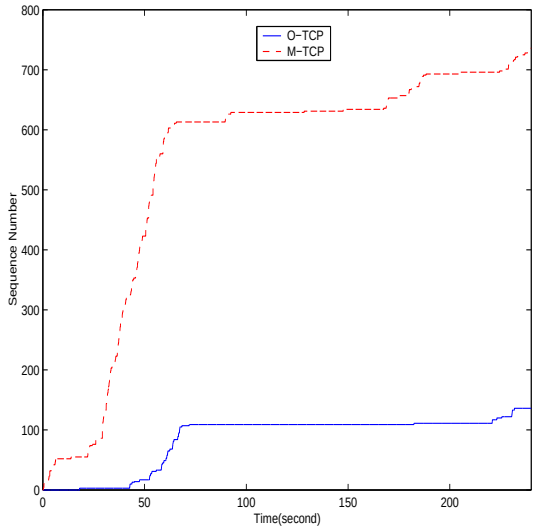
We didn't scatter packets on two paths in lossy environment. With high packet loss probability, the packet scattering will not bring benefit so we didn't adopt this approach in this paper.

### *B. Dynamic Network*

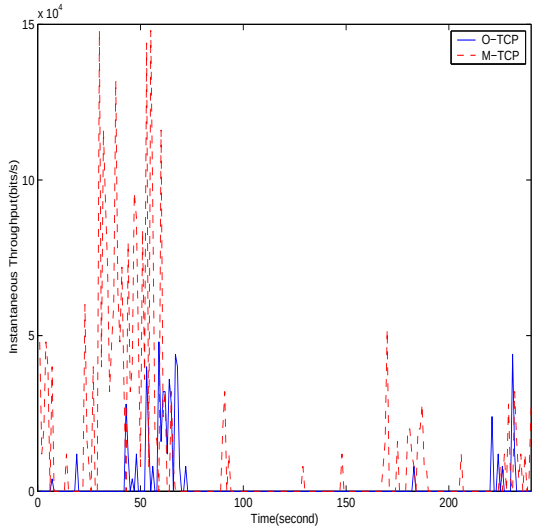
In this section we studies the TCP performance under lossy environment with node mobility. The experiment consists of 50 nodes randomly placed within a 3000mx500m area shown in Fig.5. Each node will continuously move within this area. Different experiments were run at maximal speeds ranging from 10m/s to 40m/s. The link loss rate ranges from 0 to maximum 20 percent. We use random waypoint model for node mobility. For each mobility speed, 25 different mobility trace files are generated. Further, 25 different simulation are run for each trace file. We take the average of total 625 simulation runs to generate one point in all the following graphs. In all simulations, we only show one TCP connection for clarity.

In the lossy environment, the throughput of TCP is usually low. The total throughput of O-TCP and M-TCP are shown in Fig.6 under different loss rates ranging from 5 to 20 percent. The performance of M-TCP is clearly better than O-TCP. In Fig.7, we show the total throughput gain of M-TCP over O-TCP. The graph shows that M-TCP has at least 30 percent throughput improvement when the loss rate is beyond 5 percent. The throughput gain is defined as the average throughput of M-TCP divided by average throughput of O-TCP. Each point in the graph can be viewed as the averaged data over more than 600 different runs with different mobility trace files.

When loss rate increases, M-TCP typically achieves more throughput than O-TCP because duplicate packets enable TCP to fight with packet loss. The throughput gain increases as the loss rate increases. For a fixed loss rate, the throughput gain generally decreases when the mobility speed increases. This is because when nodes

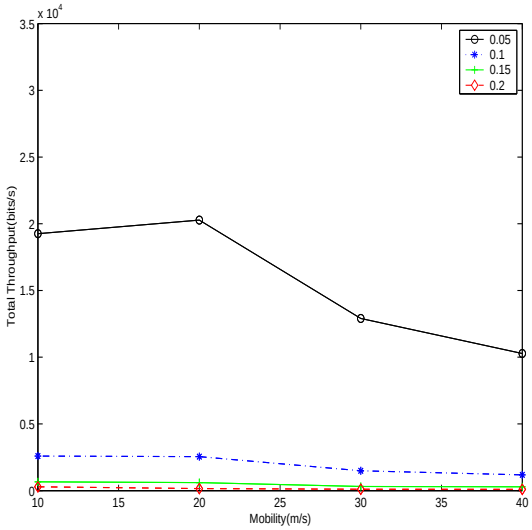



(a) Sequence Number Acked

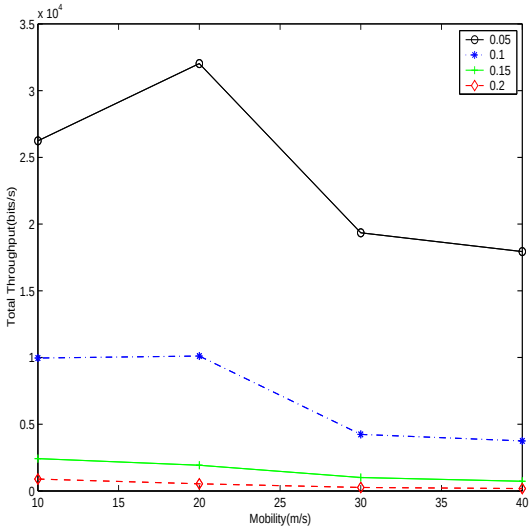



(b) Instantaneous Throughput

Fig. 4. Throughput in Ring Topology with Loss Rate=0.05




(a) Total Throughput of O-TCP




(b) Total Throughput of M-TCP

Fig. 6. Total Throughput for Variable Mobility with Different Loss Rates

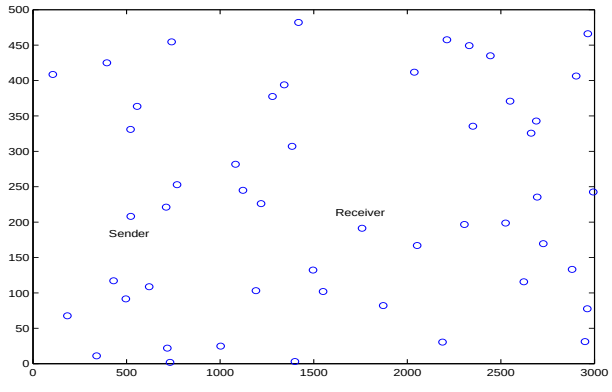



Fig. 5. Dynamic Network Topology

move faster, paths between the sender and receiver will break often. Even using two paths concurrently cannot protect TCP from frequent route failures. The result is somewhat counterintuitive. We would expect that using two paths would be an insurance against path breaks. If one path breaks, the other is still up! In fact, UDP experiment show some improvement when multiple paths are used. Instead, M-TCP performs progressively worse as speed increases and breaks become more frequent. This confirms earlier published results showing that the multiple paths do not help, rather degrade TCP throughput. The cause of this multipath degradation must be sought in the fact that the path recovery is more

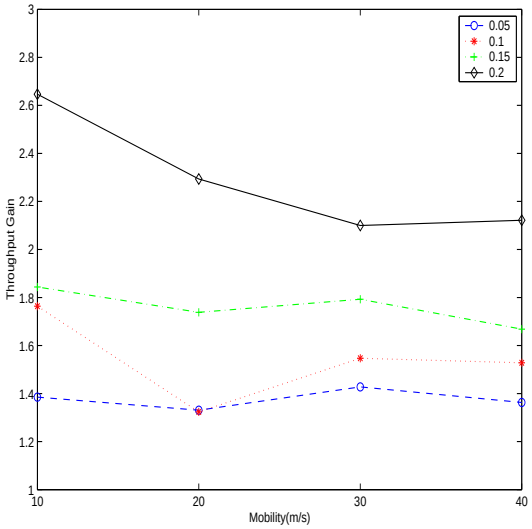



Fig. 7. Throughput Gain

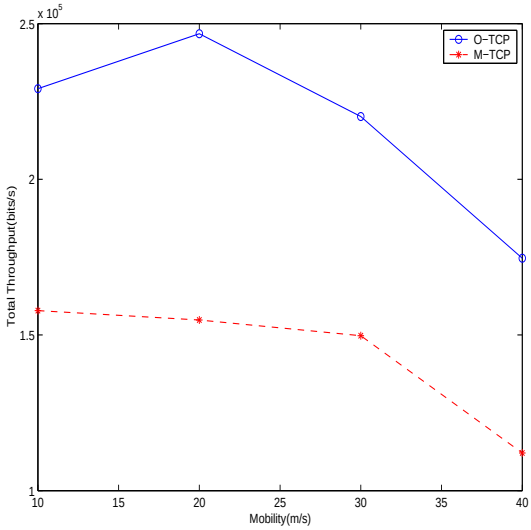



Fig. 8. Total Throughput for Variable Mobility without Random Loss

expensive (overhead wise) in multipath than in single path. The frequent path changes caused by high mobility will invalidate path selection at the sender. Two paths could interfere with each other more often when nodes move fast, and duplicate packets possibly go through the same path near the destination. Moreover, TCP is much more sensitive to sporadic losses than UDP. Thus, whatever small gain was obtained with UDP multipath, it is lost with TCP.

Multiple paths help with error but do worse in pure mobility. This effect is shown in Fig.8, where M-TCP performs worse than O-TCP. This reminds us that we should use multiple paths only in an error mode. For instance, in an urban or CAMPUS mesh network with static routers and nodes we certainly use multipath. In a very mobile battlefield, we use single path. Our next step is to design a dynamic scheme which can automatically switch between single path and multiple path by monitoring random error rate.

## V. Discussion and Future Work

There are two major results from this study. One is a positive result, stating that multipath helps TCP in high interference scenarios. The second result is a partially negative result: multipath hurts TCP in high mobility. However, the silver lining here is that, is spite of the mobility problem, when random error and mobility are combined, M-TCP is still a cost-effective option.

An issue that requires further work is interference between the paths. If two paths selected can interfere with each other, the performance can be worse than using only one path under normal conditions. We plan to further investigate the issue, taking into account the sensing versus transmission ranges.

Another issue is ACK duplication. Duplicate ACKs are not required for redundancy(the sporadic loss of ACKs is not critical because of the cumulative significance of ACKs). On the other hand, duplicate ACKs introduce more O/H and interfere with data traffic. Moreover, they can mislead TCP and force it to cut the window. As proposed in [13], a new sequence number or the TCP timestamp option wound enable TCP receiver to determine if a duplicate packet is a multipath duplicate packet or duplicate retransmission from the source. However, it requires an intelligent receiver. In this paper, we present M-TCP only by send-side modification, the cooperation of receiver will be further investigated. Another option could be to use TCP Westwood(TCPW)[14] since duplicate acks will not affect TCPW's bandwidth estimation and window control, the extra duplicate ack will not has dramatic impact on performance.

## VI. Conclusion

In this paper, we proposed a multipath TCP (M-TCP) scheme which can enhance the performance of TCP in a lossy wireless environment. In such environment, the conventional single path based TCP shows very poor performance due to high packet losses and link breaks on the path. By utilizing multiple paths from sources to destinations and duplicating TCP data packets on these multiple paths, the M-TCP can greatly reduce the packet loss rate from end to end, which in turn improves TCP performance. Simulation investigations also confirm that

M-TCP can achieve much better performance than the conventional TCP. The mobility results show that the M-TCP advantage over O-TCP still exists. However, the throughput gain decreases as speed increases. This confirms earlier results reporting the fact that multipath routing does not make TCP more robust in mobile situations.

## REFERENCES


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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] H. Lim, K. Xu, and M. Gerla, "Tcp performance over multipath routing in mobile ad hoc networks," *Proceedings of IEEE ICC'03*, May. 2003.

[9] 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.

[10] R. S. H.Y. Hsieh, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," *Proceedings of ACM MobiCom02*, Sep. 2002.

[11] "The network simulator - ns2," *Available at http://www.isi.edu/nsnam/ns/*.

[12] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Proceedings of ACM MobiCom'03*, September 2003.

[13] Y. Zhang and F. Wang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response," *Proceedings of ACM MobiHoc'02*, June 2002.

[14] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," *Proceedings of ACM MobiCom'01*, July 2001.