# Effects of Multipath Routing on TCP Performance in Ad Hoc Networks

Zhenqiang Ye
Department of Electrical Engineering
University of California, Riverside
*zye@cs.ucr.edu*

Srikanth V. Krishnamurthy, Satish K. Tripathi
Department of Computer Science and Engineering
University of California, Riverside
*krish,tripathi@cs.ucr.edu*

***Abstract*—In mobile ad hoc networks, one might expect multipath routing to provide some robustness to link failures and facilitate the transmission of packets along paths that avoid regions of congestion. Consequently one would expect an improvement in network performance in terms of the achieved throughput. In this paper we consider TCP goodput as the metric of performance. We find that in contrary to the aforementioned expectations, not all TCP connections enjoy the benefits of multipath routing. Specifically, we find that while long (in terms of hop-count) TCP connections seem to benefit, short connections in fact suffer a slight degradation in goodput as compared to TCP using the single shortest path. Furthermore, we find that alternate path routing, wherein packets are routed on a secondary alternate path only upon the failure of the primary path, helps achieve almost the same goodput as when the multiple paths are used simultaneously. The main benefits of currently proposed multipath routing schemes seem to be limited to improving the efficiency of route discoveries that are initiated either due to real route failures (due to mobility) or due to false failures (due to interference effects) for long TCP connections.**

***Keywords*– Ad-hoc Networks, TCP, Multipath Routing, Simulations.**

## I. Introduction

TCP is known to perform poorly in MANETs [1][2][3]. This may be attributed to (a) mobility and (b) congestion at the medium access control layer. One might expect multipath routing to help in alleviating packet losses due to both of the above factors. If a path were to fail between a source and a destination, the presence of the other paths should still allow TCP to continue to send data. Furthermore, depending upon where the congestion occurs, it may be possible for a TCP connection to choose a route such that congested areas are bypassed.

While in prior work, there have been changes proposed to TCP to improve its performance in MANETs [2][3][4][5], and there have been many studies on multipath routing and the routing of UDP packets over multiple paths [9][11][12][13], we find that studies that combine TCP and multipath routing have been very limited. To the best of our knowledge, the only studies in which TCP in used in conjunction with multipath routing in ad hoc networks are in [8][9][10]. The authors of [8] use the strategy of scheduling packets on the multiple paths such that the fraction of packets routed on a path is proportional to the Round Trip Time (*RTT*) observed on that path. However, they do not mention how the source and destination nodes are chosen, nor do they study various approaches of using the multiple paths. Their studies are limited to node-disjoint multipath routing. In [9], the authors evaluate the performance of a single TCP connection with multiple paths. They do not study the interactions between multiple TCP connections when multipath routing is used. In [10], TCP performance over a DSR-based multipath routing framework is investigated. It is observed that in most of the cases considered, using multiple paths simultaneously may actually degrade TCP performance. Although our observations of the aggregate effects are similar, we perform microscopic studies to understand the effects of multiple paths on the connections of various path lengths. Since the earlier work only considers aggregate average throughput of all the considered TCP connections, a macroscopic quantity, as the metric for evaluation, the studies do not clearly elucidate whether the throughput of all of the TCP connections degrades or only the throughput of some of the TCP connections degrades while that of others in fact improves.

Our objective is to examine if our conjecture that TCP can benefit from multipath routing is in fact correct in ad hoc networks. We find that in static networks, while long TCP connections (in terms of hops) can enjoy some benefits, short connections, in fact, experience a degradation in performance in terms of the observed TCP goodput. Here we define TCP *goodput* as the number of sequenced bits that a TCP receiver receives per second (out-of-order packets and duplicate packets are not counted). The overall goodput of a network that contains both long and short connections is in fact lower with multipath routing than without it.

Furthermore, we observe that alternate path routing, wherein multiple paths are found but a secondary path is used only upon the failure of the primary shortest path has a performance that is almost identical to that of multipath routing in which the multiple paths are used simultaneously. As alluded to earlier, in our framework, while performing multipath routing, the fraction of the packets that are routed on a particular path is proportional to the *RTT* observed on that path [8]. The idea is to send a higher number of packets on the least congested paths and at the same time, attempt to alleviate congestion on the paths for which high *RTT*s are observed.

In mobile ad hoc networks, we observe that, as in the static case, short TCP connections enjoy no benefits from multipath routing. Longer TCP connections (between 5 and 9 hops) do enjoy similar benefits; however the performance drops if the TCP connections are even longer. We find that the benefits enjoyed by the long connections are primarily due to improvements in the

This work was supported by DARPA under contract number: FTN F30602-01-2-0535.

efficiency of the route discovery phase when a reactive routing protocol is used.

In a nutshell, these results seem to suggest that multipath routing helps in alleviating the effects of link failures (both real failures due to mobility and false failures that occur as a consequence of using the IEEE 802.11 MAC protocol) to a limited extent; alternate paths are made available upon route failures and the efficiency of the route discovery queries, if an on-demand routing protocol is used, is improved. These benefits are primarily enjoyed by long TCP connections. From the fairness point of view, multipath routing can alleviate unfairness between short TCP connections and long TCP connections.

The remainder of this paper is organized as follows. In Section II, we give a brief description of the multipath routing protocol that we use in this paper. In Section III, we describe the scheduling and routing policies that we use. In Section IV we describe the simulation environment and discuss various parameters of interest. The performance results of TCP with the various routing policies are evaluated in Section V. Finally, we conclude in Section VI.

## II. The Multipath Routing Protocol

In this paper, we use a multipath routing protocol that is constructed by modifying the AODV [15] protocol. Note that this work is not totally new. Similar modifications have been proposed in [11] and [13]. Our objective in this paper is not to propose a new multipath routing protocol but to study the effects of multipath routing on TCP performance. Our modifications (similar to those proposed in [13]) facilitate the discovery of multiple edge-disjoint or node-disjoint paths in a single route discovery instance. In the rest of this section we describe our modifications in brief.

When a source node needs a route to send packets to a destination node, it initiates a route discovery process. Route discovery typically involves a network-wide flood of a route request (RREQ) packet. When an intermediate node receives the first copy of this RREQ packet, it forwards the RREQ packet to its neighbors. Instead of discarding duplicate RREQ packets (as in AODV), intermediate nodes are required to record the information contained in these packets in an RREQ table. For each received copy of an RREQ packet, the receiving intermediate node records the identity of the source that generated the RREQ, the identity of the destination for which the RREQ is intended, the identity of the neighbor who transmitted the RREQ, and the number of hops that the packet has traversed thus far. Furthermore, in order to discover multiple edge-disjoint or node-disjoint paths, intermediate nodes are precluded from sending a route reply (RREP) packet directly to the source as is possible in AODV.

When the destination receives the first RREQ packet from one of its neighbors, it generates an RREP packet. The RREP packet is modified to contain an additional field called *ROUTE_ID*. Each path discovered during a single route discovery instance is assigned a unique *ROUTE_ID* by the destination. The RREP packet with a particular *ROUTE_ID* is sent back to the source via the path identified by the *ROUTE_ID*. When the destination receives duplicate copies of an RREQ packet from its neighbors, it generates an RREP packet for each of the copies; each RREP packet contains a unique *ROUTE_ID*.

When an intermediate node receives an RREP packet from one of its neighbors, it deletes the entry corresponding to this neighbor from its RREQ table and adds a routing entry to its routing table to indicate the discovered route to the destination, which is the originator of the RREP packet. The node, then, identifies the neighbor in the RREQ table via which, the path to the source is the shortest and forwards the RREP packet to that neighbor. The entry corresponding to this neighbor is then deleted from the RREQ table. By using this approach, the discovered multiple paths are edge-disjoint. In order to discover multiple node-disjoint paths, additional operations are needed. Specifically, when an intermediate node *overhears* any of its neighbors broadcasting an RREP packet, it deletes the entry corresponding to the transmitting neighbor from its RREQ table. This additional operation ensures that the discovered multiple paths are node-disjoint.

Once the source node receives an RREP packet, a route is established and is used to transmit data. If a link fails, the node that detects the link failure (through feedback from the link layer), sends a route error (RERR) packet to the source, upon the receipt of which, the source renders the particular route unusable. A route discovery is initiated only if all the routes to the destination fail.

## III. TCP with Multiple Paths

In order to facilitate the use of multiple path and ensure backward compatibility with the standard TCP protocol, in this paper, we keep the TCP layer intact and insert a Packet Scheduling layer (PS layer) between the TCP layer and the routing layer. When a TCP packet is generated, it is first sent to this PS layer. The PS layer schedules the packet according to a certain policy (the scheduling policies will be discussed later), i.e., determines the route on which the packet should be sent. The choice of the route is indicated to the routing layer via an apt interface. The routing layer then appropriately inserts the correct value to indicate the choice of the route in the *ROUTE_ID* field and forwards the packet on the corresponding route.

Once intermediate nodes receive data packets that need to be forwarded, they simply examine the *ROUTE_ID* field of the packets to identify the route in their routing tables, and forward the packet to the next hop node on that route. After the receiver receives a TCP packet, it generates an ACK packet. The ACK packet will contain the same *ROUTE_ID* as that of the received TCP packet. It is then forwarded to the sender via the reverse path.

### A. Routing Policies

In this section, we introduce various routing policies that TCP is used in conjunction with, in this paper:

- **Single Path Policy (SP)**: The routing protocol finds only a single path per route discovery instance. All packets from the TCP session that stimulated the route discovery traverse this path.
- **Alternate Path Policy (AP)**: The routing protocol discovers multiple *edge-disjoint* paths per route discovery instance. All packets from the TCP session in discussion are initially routed on the shortest path. If this path were to fail, an attempt is made to route packets on the next shortest path and

so on until no paths remain, i.e., all of the computed paths fail.

- **Edge-Disjoint Multipath Routing Policy (EDM)**: The routing protocol discovers multiple *edge-disjoint* paths per route discovery instance. All the available paths are simultaneously used to transmit TCP packets as determined by the PS layer. A new route discovery is initiated only if all of the paths discovered in the previous instance fail.
- **Node-Disjoint Multipath Routing Policy (NDM)**: The routing protocol discovers multiple *node-disjoint* paths per route discovery instance. All of the available paths are simultaneously used to transmit TCP packets (as per the PS layer specifications) until none of the paths remain. When all the paths fail, as in EDM, a new route discovery is initiated.

## B. Packet Scheduling

As mentioned earlier, if multiple paths are used simultaneously, a packet scheduling strategy is needed to decide upon the path on which a packet should be routed. The scheduling policy would be typically based on the information available with regards to each path. In this paper we adopt the strategy proposed in [8] which is to schedule packets on each path based on the *RTT* observed on the various paths. The policy essentially attempts to route packets on the least congested paths. The approach seems reasonable since a higher *RTT* would indicate a larger extent of congestion. Furthermore, this approach is attractive since it does not require the maintenance of state at the intermediate relay nodes. Below we give a brief description of the policy.

Let the number of available paths between a source and a destination be $N$ and the average *RTT* on path $k$ ($1 \leq k \leq N$) be $RTT_k$. $RTT_{max}$ is defined as the maximum of all the *RTT*s, i.e.,

$$RTT_{max} = MAX(RTT_1, RTT_2, ......, RTT_N) \ . \qquad (1)$$

The weight of each path $W_k$ is defined as

$$W_k = \frac{RTT_{max}}{RTT_k} \quad k = 1, 2, ....., N \ . \qquad (2)$$

Thus, the weight of a given path is inversely proportional to the average *RTT* observed for the packets routed on that path. The smaller the average *RTT*, the larger the weight. The TCP traffic routed on a path is proportional to the path's weight. This policy attempts to increase the traffic on the under-utilized paths while reducing the traffic on the over-utilized paths, i.e., balance load dynamically. Notice that this may in turn cause variations in $RTT_k$ for each $k$, thereby changing each path's weight. This may cause swapping between routes frequently. Thus, in [8] the weight equation is refined through experiment as follows:

$$W_k = MIN(\frac{RTT_{max}}{RTT_k}, U) \cdot R \quad k = 1, 2, ....., N \ . \qquad (3)$$

where U is a bound to ensure that $W_k$ does not reach a very large value. R is a factor that is used to control the frequency with which routes are swapped. Once a path is chosen, $\lfloor W_k \rfloor$[1] consecutive packets will be sent out via this path.

[1] $\lfloor x \rfloor$ represents the largest integer that is less than or equal to $x$.

## IV. Simulation Environment

We implement a simulation model in *ns-2* [16] to evaluate the performance of TCP used in conjunction with the multipath routing strategies listed earlier. The distributed coordination function (DCF) defined in the IEEE 802.11 standard [17] is used at the MAC layer. The radio model is similar to a commercial radio interface, Lucent's WaveLAN, which is a shared-media radio with a nominal bit-rate of 2Mb/sec and a nominal radio range of 250 meters. The performance metrics that we are interested in are the absolute goodput of TCP and the "goodput ratio" which is the TCP goodput observed with a multipath routing strategy as compared with the single path routing strategy.

In our simulations we place 200 nodes in a 2500m x 1000m region. In each simulation iteration, a random scenario is generated; 10 <source, destination> pairs are randomly chosen and TCP connections are established between these pairs. These TCP connections begin sequentially. The initiation instances of consecutive TCP sessions are separated by 1 second. Each TCP connection lasts for 100 seconds and is then terminated. The simulation results reported in Section V represent the average results over 1000 different scenarios.

TCP New-reno is used in all our simulations. The length of each TCP packet is 1460 bytes. In order to avoid inactivity due to a sequence of TCP back-off operations which are triggered by a series of packet losses due to link failures, in our simulations, the TCP sender disables its retransmission timer and enters a standby mode upon receiving a route error (RRER) packet. Subsequently, the sender sends out a packet periodically until an acknowledgement is received; the period is equal to the current value of its retransmission timer. Our approach is similar to the ones used in [2] and [3]. The maximum congestion window size is set to be 8 [3]. If multiple paths are used simultaneously, packets that traverse different paths may reach the receiver in different order. In order to reduce the possibility of out-of-order packet delivery, in [8], $R$ is set to 3 (See Equation 3). As a consequence, when a route is chosen, at least 3 packets are sent out on that route. However, in our simulations, we found that setting this parameter to 3 does not help much in avoiding out-of-order packet delivery. In the reported simulation studies of this paper we set $R = 1$ and $U = 5$. We varied the number of duplicate ACKs that the sender must receive before it enters the congestion avoidance state. We found that when this number is set to 5, TCP with EDM and TCP with NDM perform the best. So in our simulations, we set this number to 5. We also find that changing this value from 3 to 5 has no significant impact on the performance of TCP with SP and TCP with AP.

We limit the maximum number of paths that may be discovered in a single route discovery instance to 3, since, it has been observed that additional routes only provide marginal benefits [14]. Furthermore, in order to avoid extremely long paths, we require the alternate paths to be at most 1.3 times the shortest path in terms of hop count.

In our simulations with mobile nodes, we use the random way point model to simulate node mobility. The pause time was set to zero to facilitate continuous motion. The velocity of each node is uniformly distributed over $[0, V_{max}]$. Since our objective is to study the effects of multipath routing on both long (in terms of hop-count) and short TCP connections, we would need to keep the distance between the source and the destination of a TCP connection relatively unchanged during a simulation run in order to classify the connection as either long or short. In our simulations, we thus make the 20 TCP end nodes static, while all the other nodes are allowed to move in accordance with the mobility model.

## V. Performance Evaluation

In this section, we evaluate the performance of TCP when used with multipath routing and compare it with that of TCP over a single path. We interpret our observations and investigate the impact of various parameters such as the distance between the end nodes, the number of TCP connections and mobility on the performance.

### *A. Performance of TCP with Multipath Routing in a Static Network*

Figures 1(a), 1(b) and 1(c) show the goodput of TCP with the various multipath routing strategies and their goodput ratios as compared with TCP with SP in a static network. From these figures, we see that TCP with multipath routing performs worse than TCP with SP if the distance[2] between the <source, destination> pair is less than 5 hops. If the hop count is increased beyond 5, using multiple paths can improve the TCP goodput by a certain ratio as compared with TCP with SP; the ratio increases with the distance (in hops). When the distance between the <source, destination> pair is 12 hops, using multiple paths can improve TCP goodput by as much as 60%. On the other hand, TCP with AP, which simply uses the multiple paths one by one, performs almost identically, as compared with TCP with EDM. Thus, one might infer that the multiple paths between the two nodes are usually closely coupled with each other and the spatial diversity benefits obtained by using these multiple paths simultaneously is very limited. Note that TCP with NDM performs the worst, since, although using node-disjoint paths can potentially minimize the interference effects of the transmissions of one path on another, the paths are usually much longer and hence, cause an overall reduction in TCP goodput.

### *B. Impact of the Distance between End Nodes*

From Figure 1 we see that not all TCP connections can benefit from using multiple paths. There are three reasons why multiple paths fail to improve TCP goodput when the two end nodes are close to each other.

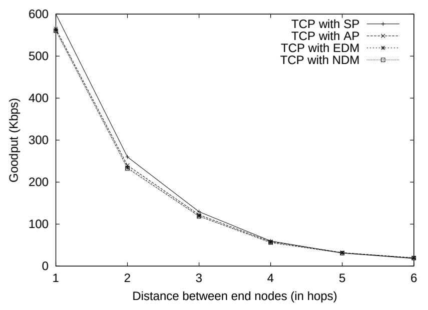
First, all the nodes in an ad hoc network share the same wireless bandwidth; if a node is transmitting, other nodes within a certain range of the transmitting node cannot transmit. There are two ranges typically used in simulation models [16]: the transmission range and the sensing range. The transmission range is the maximum distance between two nodes such that one node can receive the other node's signal and decode it correctly. A node is said to be within the sensing range of another node if it can hear the other node transmitting, but the Signal-to-Noise Ratio (SNR) is so low that the receiver cannot correctly decode the signal. The sensing range is much larger than the transmission range. In the 802.11 standard, the nominal transmission range is defined to be 250m. The sensing range is assumed to be 550m and is more than twice the transmission range. The 802.11 MAC protocol ensures that while a node is transmitting, other nodes within the sensing range of that node cannot transmit [17].

[2]We define the distance between two end nodes to be the length of the shortest path (in terms of the number of hops) that is observed between these two nodes during the simulation run.

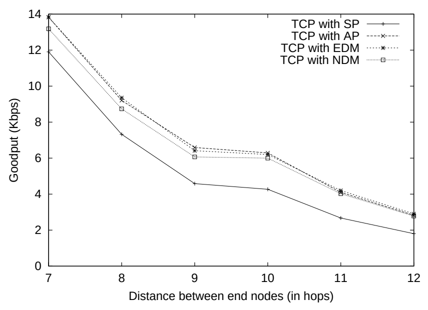
We consider an example in Figure 2 to demonstrate why the existence of multiple paths does not translate to an increase in goodput for short TCP connections. Suppose a TCP connection is established between node A and node E. Note that the distance between consecutive relay nodes on a path can be *at most* equal to the transmission range. As such, in this example two paths are discovered: ABCDE and AFGHE. However, notice that communication is possible only on a single link at any given time since each node is within the sensing range of the others. Thus, even though there are two four-hop paths, only a single path can be used at a given time. Thus, the existence of multiple paths does not translate into a spatial diversity benefit that can lead to an increase in goodput. Specifically, when the source-destination pair are close to each other (within 4 hops), only one of the links of the TCP connection can be active at any given moment. There are no possible spatial reuse benefits regardless of the number of paths that might exist between the source-destination pair. In this regime (short distance between the source-destination pair) the TCP goodput is limited by the inherent exponential capacity drop-off as the number of nodes in the connection increases. As an example, if let a single one hop TCP connection exist in isolation and the capacity of the link be $C$. If now we consider a two hop TCP connection, since only one of the three component nodes forming the connection can transmit at any given time, only one of the two links can be active at any time. Thus, the maximum capacity enjoyed by the connection drops to $C/2$. Similarly the addition of a third hop drops the maximum achievable capacity to $C/3$. This has been reported in prior work as well [19]. The effect cannot be alleviated via multipath routing and is an inherent constraint. From Figure 1(a) we see that the average TCP goodput decays exponentially when the distance between the source-destination pair increases from 1 hop to 4 hops.

A second reason for the degradation in the observed goodput for short connections is that long TCP connections *steal* some bandwidth from short TCP connections. Long TCP connections respond much more slowly to route failures and packet losses than short TCP connections. If multiple paths are available for long TCP connections, the recovery from route failures is much faster and this helps the long TCP connections compete better with short TCP connections for wireless bandwidth.

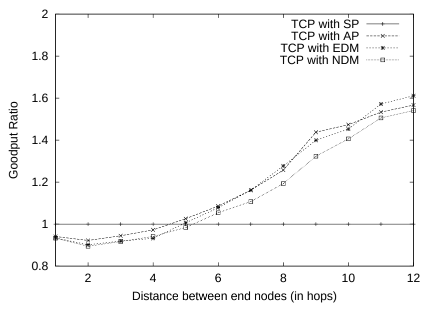
Finally, the use of sub-optimal paths if the primary path were to fail, is another reason for the degradation in the goodput of short TCP connections. Notice that in this static case all the link failures are "false" link failures, a direct consequence of the capture effect of the IEEE 802.11 protocol. The multipath routing protocols discard the shortest path at the first instance of failure and attempt to use alternate longer paths. However, single path routing initiates a route discovery upon failures and in many cases re-discovers the shortest path. Since the TCP goodput is very sensitive to path length (See Figure 1(a)), this, turns out to be to TCP's advantage.



(a) Average goodput of TCP connections (distance: 1-6 hops)



(b) Average goodput of TCP connections (distance:7-12 hops)



(c) Comparison of goodput ratio for various routing policies

Fig. 1. Average TCP goodput and comparison of goodput ratio (10 TCP connections, $V_{max} = 0m/s$).

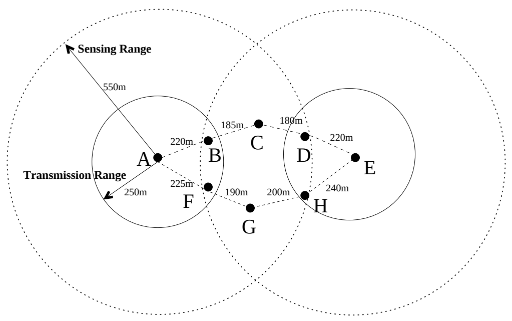


Fig. 2. Transmission range and sensing range

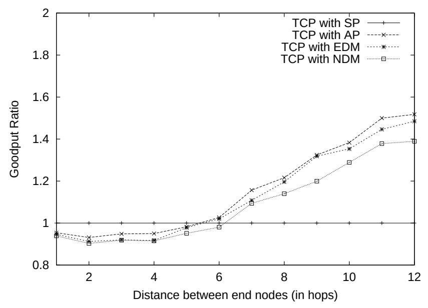


Fig. 3. Comparison of goodput ratio for various routing policies (10 TCP connections, $V_{max} = 0m/s$. For short TCP connections, only one path is discovered per route discovery instance)

### C. Modifications to the Routing Protocol to Help Shorter Connections

. In order to quantify benefits (if any) that can be obtained by multipath routing without deteriorating the goodput of the short TCP connections, we modify the multipath routing protocol as follows: if the destination receives an RREQ packet that has traversed less than 6 hops, it sends only a single RREP packet to the sender and ignores duplicate RREQ packets that might be received. Thus, only a single path is discovered per route discovery instance if the source-destination pair are within 5 hops of each other. If the distance is no less than 6 hops, the multipath routing protocol works in its normal mode (it discovers multiple paths per route discovery instance). Figure 3 shows the simulation results with this modification. We see that even though the goodputs of short TCP connections are improved marginally, they are still less than the goodputs achieved with TCP with SP. We observe that the long TCP connections continue to enjoy some benefits due to fast recovery from link failures. Thus, the use of multiple paths for routing TCP packets seems to alleviate TCP unfairness to some extent.

In order to understand why short TCP connections suffer even when the single shortest paths are used by these connections, we further investigate the average *RTT* experienced by the various TCP connections. The observations show that for the TCP connections that are less than 6 hops, the average *RTTs* increase by up to 5% when multipath routing is used. The average *RTT* of a “one-hop TCP connection” increases by about 3%. The increase in *RTT* for short TCP connections is due to the increased levels of congestion caused due to an increased level of packet injection rates by the longer TCP connections, thanks to the faster recovery time experienced upon link failures due to multipath routing. The increased congestion causes longer packet queueing delays for short TCP connections even though they simply use the single shortest path.

In order to see if there are benefits (due to exploitation of spatial diversity) other than those due to fast recovery from link failures, we eliminate false link failures (the only link failures since the nodes are static) by setting the RTS retry limit of the IEEE 802.11 MAC protocol to infinity (usually this number is set to be 7). Figure 4 shows that without false link failures, short TCP connections still suffer. At the same time, the benefits of multipath routing dwindle for long TCP connections. This reduction in benefits is because the scheduling mechanism that we use (as in [8]) cannot take into account the temporal correlations between the *RTTs* experienced on the multiple paths. The traffic on one path adversely affects the traffic on the other path. This effect worsens with our changes since now packets are not dropped at the MAC layer at all. Note that the bandwidth consumed by a $k$ hop TCP connection is at least $k$ times that being consumed by a one hop TCP connection. Thus, in Figure 4, even though the goodput of short TCP connections decrease by about 10%, the goodput of long TCP connections only increase marginally. Since the performance of TCP with AP performs is identical to that of TCP with SP if there are no link failures, we do not include the performance of TCP with AP in Figure 4.

### D. Impact of the Number of TCP connections

In order to examine the impact of varying number of TCP connections on the performance of TCP over multiple paths, we fur-

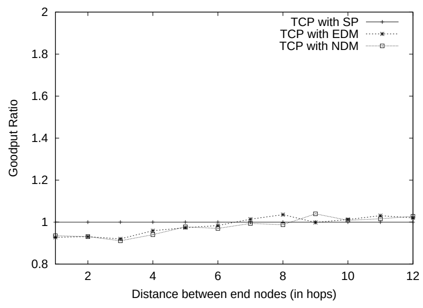


Fig. 4. Benefits due to spatial diversity when multiple paths are used simultaneously (10 TCP connections, $V_{max} = 0m/s$, no false link failures)

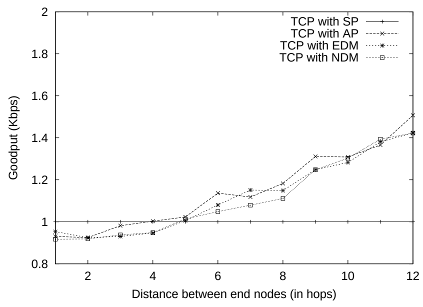


Fig. 5. Comparison of goodput ratio for various routing policies (5 TCP connections, $V_{max} = 0m/s$)

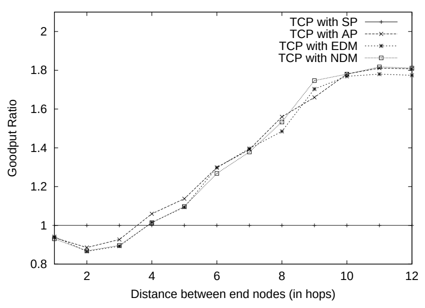


Fig. 7. Comparison of goodput ratio for various routing policies (10 TCP connection, $V_{max} = 10m/s$)

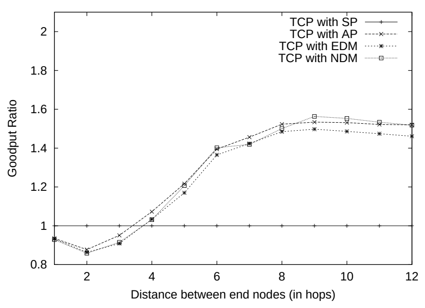


Fig. 8. Comparison of goodput ratio for various routing policies (10 TCP connection, $V_{max} = 20m/s$)

ther simulate two cases. In the first case, five TCP connections are established in the network (Figure 5); In the second case, only one TCP connection is established in the network (Figure 6). We see that the behavior in terms of the increase/decrease in goodput ratio of the TCP connections, on average, is independent of the offered load. Short connections tend to do worse with multipath routing while long connections tend to benefit. Note that when there is only one TCP connection in the network, the decrease in goodput of short TCP connections is solely caused by using sub-optimal paths (the last factor discussed in Section V-B).

## *E. Impact of Node Mobility*

Next we examine the case wherein nodes are now mobile. As mentioned earlier, we keep the 20 TCP end nodes static[3] during the simulation runs. All the other nodes are allowed to move around. Figures 7 and 8 compare the goodput achieved with the various routing strategies when $V_{max} = 10m/s$ and $V_{max} = 20m/s$ respectively. We see that TCP goodput is fairly high for all cases when the hop count between the source and destination pair is small ($\leq$3 hops). Furthermore, the difference in performance for the various routing policies is insignificant. However, for higher hop counts the goodput is low but the multipath policies outperform TCP with SP. From Figures 7 and 8, we see that with mobility the goodput ratio of TCP with any of the multipath routing strategies to that observed for TCP with SP tends to first increase with hop count. However, beyond a certain hop count the increase is less significant. In fact, when the mobility is high ($V_{max} = 20m/s$) the goodput actually decreases a little if we increase the hop count beyond 9. In general, note that the behavior is similar irrespective of whether the maximum speed is $10m/s$ or $20m/s$.

In the experiments thus far, since, the performance difference between TCP with AP and TCP with EDM or TCP with NDM is insignificant, one might infer that the latter schemes are unable to enjoy spatial reuse benefits in spite of attempting to bypass congestion by scheduling a larger number of packets on the paths that experience lower *RTTs*. Thus, the benefits are primarily due to the alleviation of the effects of link failures either due to mobility or congestion.

In particular, the primary reason for the observations in Figures 7 and 8 is that when the distance between two nodes is large and when nodes move with high speeds, the lifetime of a path is short. In such cases, the lifetime of the longest life path is almost equal to the life time of the shortest life path. In other words, if a path breaks, it is highly probable that the other paths will also break in the very near future. Therefore, one might expect that

[3]The positions of these nodes are randomly chosen for every simulation run.

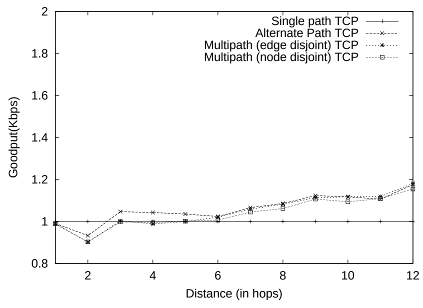


Fig. 6. Comparison of goodput ratio for various routing policies (1 TCP connection, $V_{max} = 0m/s$)

if two nodes are extremely far from each other, using multiple paths does not provide any noticeable benefits over simply using a single path. We also observe that TCP with NDM performs the worst in the static case (Figure 1); however it performs better than TCP with AP or TCP with EDM in the mobile case. This is because the mobility of a node can cause multiple paths to fail in the latter two cases. Thus, these policies are more susceptible to mobility than TCP with NDM.

## VI. Conclusions

In this paper we examine the effects of multipath routing on TCP goodput in wireless ad hoc networks. We consider different multipath routing strategies: (a) finding multiple edge-disjoint paths and using the shortest one until it breaks, then switching to the next shortest path and so on, (b) simultaneously using multiple edge-disjoint paths and (c) simultaneously using multiple node-disjoint paths. When we use multiple paths simultaneously, we use a previously proposed scheduling policy, wherein, the number of packets that are scheduled on a route is inversely proportional to the average *RTT* experienced on that route. In contrary to our expectations, not all TCP connections are seen to benefit or degrade from the use of multiple paths. Long TCP connections benefit to a certain extent while short TCP connections may even suffer a slight degradation in goodput as compared with TCP using the single shortest path. We find that spatial diversity benefits of using multiple paths simultaneously is very limited, since the transmission on one of the paths interferes with the transmissions on the other paths. Thus, only a single path may be used at a given time. We observe that as a consequence, the different multipath routing strategies behave identically in terms of TCP goodput. Furthermore, we observe that the benefits of multipath routing in coping with (real or false) link failures are in terms of improving the efficiency of the route discovery process for long TCP connections in on-demand routing protocols. From the fairness point of view, multipath routing can alleviate unfairness between short TCP connections and long TCP connections.

## References

[1] M. Gerla, K. Tang and R. Bagrodia, "TCP Performance Wireless Multihop Networks", *IEEE WMCSA*, 1999.

[2] K. Chandran, S.Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks", *IEEE Personal Communications Magazine*, pp.34-39, Feb. 2001.

[3] G. Holland and N. Vaidya "Analysis of TCP Performance over Mobile Ad Hoc Networks", *ACM MobiCom*, 1999.

[4] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-Of-Order Detection and Response", *ACM MobiHoc*, 2002.

[5] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks", *IEEE Journal on Selected Areas in Communications*, Vol.19, No.7, July 2001.

[6] K. Sundaresan, V. Anantharaman, H. Hsieh and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad-hoc Networks", *ACM MobiHoc*, 2003.

[7] M.R. Pearlman, Z.J. Haas, P. Sholander, and S.S. Tabrizi, "On the Impact of Alternate Path Routing for Load Balancing In Mobile Ad Hoc Networks", *ACM MobiHoc*, 2000.

[8] L. Wang, L.F. Zhang, Y.T. Shu, M. Dong, and O.W.W. Yang, "Adaptive Multipath Source Routing in Wireless Ad Hoc Networks", *IEEE ICC*, 2001.

[9] M.R. Pearlman, Z.J. Haas, P. Sholander, and S.S. Tabrizi, "Alternate Path Routing in Mobile Ad Hoc Networks", *MILCOM*, 2000.

[10] H. Lim, K. Xu and M. Gerla, "TCP performance over Multipath Routing in Mobile Ad Hoc Networks", *IEEE ICC*, 2003.

[11] M.K. Marina and S.R. Das, "On-demand Multipath Distance Vector Routing in Ad Hoc Networks", *ICNP*, 2001.

[12] K. Wu and J. Harms, "Performance Study of a Multipath Routing Method for Wireless Mobile Ad Hoc Networks", *IEEE MASCOTS*, 2001.

[13] Z. Ye, S.V. Krishnamurthy and S.K. Tripathi, "A Framework for Reliable Routing in Mobile Ad Hoc Networks", *IEEE INFOCOM*, 2003.

[14] A. Nasipuri, R.Castaneda, and S.R. Das, "Performance of Multipath Routing for On-demand Protocols in Mobile Ad Hoc Networks", *ACM/Kluwer Mobile Networks and Applications (MONET)*, Vol.6, No.4, pp.339-349, 2001.

[15] C.E. Perkins and E.M. Royer, "Ad Hoc On-demand Distance Vector Routing", *IEEE WMCSA*, 1999.

[16] K. Fall, and K. Varadham, "The ns Manual", http://www.isi.edu/nsnam/ns/ns-documentation.html/.

[17] IEEE Standards Department, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications", *IEEE standard 802.11*, 1997.

[18] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", *IEEE INFOCOM*, 2003.

[19] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee and R. Morris, "Capacity of Ad Hoc Wireless Networks", *ACM MobiCom*, 2001.

[20] S. Xu and T. Saadawi, "Revealing the Problems with 802.11 Medium Access Control Protocol in Multihop Wireless Ad Hoc Networks", *Computer Networks*, Vol.38, Issue 4, March, 2002.

[21] D.S.J. De Couto, D. Aguayo, B.A. Chambers, and R. Morris, "Performance of Multihop Wireless Networks: Shortest Path is Not Enough", *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, New Jersey, October 2002.