TCP Performance over Geo-routing for High Mobile Ad Hoc Networks

Jiwei Chen\(^1\), Mario Gerla\(^2\) and Yeng Zhong Lee\(^2\)

\(^1\) UCLA Electrical Engineering Department, Los Angeles, CA 90095, USA
cjw@ee.ucla.edu

\(^2\) UCLA Computer Science Department, Los Angeles, CA 90095, USA
\{gerla,yenglee\}@cs.ucla.edu

Summary. TCP performs poorly in mobile ad hoc networks, mainly because of route breakage. Geo-routing is a routing approach which has recently received attention in large scale, mobile systems as it does not require end-to-end path establishment and pre-computed packet forwarding routing structure at nodes. For these reasons, Geo-routing is robust to highly dynamic route changes. For best performance, however, several parameters must be carefully tuned (eg, hello message exchange rate, delay timer in TCP for out-of-order delivery, etc). In this paper we study the impact of these parameters on the performance of both UDP and TCP. To improve hello efficiency in Geo-routing, we propose an adaptive hello exchange scheme based on node mobility. Then, we fix the out-of-order problem in TCP by using a receiver-side out-of-order detection and delayed ack strategy by properly calibrating the parameters. We show that these parameter adjustments are critical for TCP over Geo-routing performance in high mobile applications. With these enhancements TCP over Geo-routing easily outperforms TCP over traditional ad hoc routing schemes, such as AODV.

1 Introduction and Background

Transport Control Protocol (TCP) is unquestionably one of the most widely used protocols in the Internet. TCP was originally designed for a wired network where congestion and buffer overflow account for most packet losses. Unlike wired networks, TCP performance in wireless ad hoc networks is affected by several new factors. These factors include unpredictable channels error, medium contention which may lead to capture, and frequent route breakage due to node mobility. All these factors challenge TCP to provide efficient and reliable end-to-end communications in mobile ad hoc networks. Two recent developments have stimulated great interest in TCP in highly mobile scenarios: car to car communications, file sharing and content distribution during highway driving; and tactical communications between aircraft squadrons (manned or unmanned). In these applications, mobility can be characterized
by the inverse of “contact time” with neighbors. The lower the contact time, the more challenging the routing. For example, an aircraft flying at 360Km/hr with transmission range of 300m will maintain contact with neighbors flying in random directions for an interval of the order of seconds. Thus, neighbor acquisition and route computation must be completed well below the second. Notice that “contact time” is proportional to the ratio of transmission range over speed, thus, what counts is the “relative” speed. For instance, people walking in a shopping mall (at 3m/s) with Bluetooth or Zigbee radios with 10 m range, say, would also lead to “high mobility” ad hoc net scenarios.

Many research efforts [1, 2, 3] have been made to render TCP more robust in the face of various wireless network characteristics including mobility. In mobile ad hoc networks, most of the packet losses are due to route breakage as shown in [2]. Mobility causes frequent route interruptions. If the routing algorithm cannot track node motion and packets enroute cannot be salvaged until a new route is found, TCP goes into exponentially increasing timeout intervals with severe performance hit.

How to improve TCP performance in such mobile networks has been a hot area for years. Routing of course is one key factor, thus the interaction between TCP and routing has been thoroughly investigated in the past. Unfortunately, traditional on-demand routing schemes, such as DSR and AODV, cannot efficiently address the frequent route breakage and packet loss due to high mobility. These schemes pre-compute the route at call setup time. Every node has a predefined next-hop for the designated destination. When this next-hop node moves away (or dies), the routing scheme must find another path. Unfortunately, finding another path takes time (and generally leads to TCP to time out). To avoid this delay, multi-path routing could be adopted, allowing multiple candidate next hops for packet forwarding. However the overhead of multi-path routing grows fast and becomes intolerable as node mobility increases [4]. Another solution is to exploit routing layer feedback (eg, route broken/route repaired) to TCP. In [5, 6], TCP’s state is frozen when the sender receives the route failure signal from intermediate nodes. TCP exits the frozen state when route is re-established. Fixed-RTO was proposed in [7] with selective and delayed ack to help constraining the damage when packet loss is known to come from path breakage.

Even Fixed-RTO has only limited effect (as we will show in our experiments). When nodes are moving very fast, no traditional (proactive or on demand) routing structure can adjust rapidly enough. For these extreme situations, Geo-routing [8, 9] has recently shown remarkable promise. Geo-routing uses the destination location as the “routable” address, and forwards packets (when possible) in a greedy manner towards it. Geo-routing is highly scalable, as nodes only keep geo-locations for their local neighbors. No explicit end to end route establishment is required. Since there is no pre-computed next hop to destination at set up time (as in all traditional schemes), rather the next hop is selected opportunistically “on the fly”, Geo-routing promises
to be robust to path breakage and short term channel failure if the network is sufficiently dense (i.e., there are always nodes in the “right” direction).

Geo-routing also places some extra costs on the network. It relies on information that is not needed in conventional routing schemes (e.g., GPS positioning, Geo Location Server, accurate knowledge of neighbor locations). Moreover, if the basic Geo-routing “greedy” approach fails when the packet is trapped in a “cul de sac” (sort of a local “maximum” in the greedy search), it adopts perimeter (face) mode to go around the void area.

Many of the above issues (e.g., location determination without GPS, Geo Location server, perimeter routing to circumvent local maxima) have already been studied extensively in the literature [10, 11, 12] and will not be addressed here. However, previous studies were mostly based on UDP and lightly loaded networks. We recall that Geo-routing uses hello messages to update neighbor information. In light load, the issues of hello message O/H and of interference between hellos and data packet did not emerge. TCP is rather aggressive in increasing network load, thus, it is important to “tune” the hello message rate taking into account not only speed but also load. Regarding hello messages, the careful reader will recall that some schemes [13] discover the best next hop dynamically, with an election and thus are not encumbered with background hello message maintenance all together. However, such schemes require a change in the MAC protocol (and thus in 802.11 firmware) which we exclude in this study. Moreover, they introduce the extra election overhead. Thus, one of the important contributions of this study is the hello rate optimization. Other useful contributions are the analysis and solution of the out of order delivery problem in the specific Geo-routing context.

In the process of studying TCP performance over Geo-routing, we also compare it with TCP over the traditional “mobile” ad hoc routing scheme, namely AODV [14]. We show that indeed TCP can achieve substantial benefits from Geo-routing in terms of robustness to mobility. This confirms that Geo-routing is the preferred scheme in highly mobile scenarios.

The remainder of this paper is organized as follows. Section 2 contains an impact study of various Geo-routing parameters and comparison of UDP and TCP over Geo-routing versus traditional ad hoc routing. An adaptive hello exchange scheme for GPSR and an out-of-order enhancement for TCP over GPSR are also presented. We conclude our work in Section 3.

2 TCP Performance over Geo-routing in Mobile Ad Hoc

In this section, we analyze GPSR [8], the most popular implementation of Geo-routing. We first study the impact of high mobility on UDP and TCP over GPSR and tune GPSR parameters to optimize performance. Then, we compare UDP and TCP performance on GPSR with AODV. We do not compare GPSR with other routing schemes (e.g., DSR, DSDV, OLSR etc) for lack
of space. Besides, the latter schemes tend to perform worse than AODV in mobile scenarios [3].

2.1 Case Study: Deterministic Motion

The first motion scenario, deterministic motion model, is carefully crafted to allow high mobility in a controlled way, yet maintaining end to end topology connectivity all the time. In this scenario, shown in Fig.1, the sender and the receiver are fixed. A total of 12 intermediate nodes arranged in 3 columns are moving vertically up and down in constant but opposite velocity. We place the nodes at 200m interval of each other in each column. Neighboring columns have opposite moving directions so the relative motion is twice the node speed. Recalling that the transmission range is 250m, in this scenario a path is always available from the sender to the receiver in spite of motion throughout the experiments. This is a very important detail of this experiment that we will exploit later.

In our experiments, radio rate is set to 2Mbps. Standard TCP (TCP NewReno) is used. Data packet size (for both UDP and TCP traffic) is 1000 bytes. GPSR hello packet refresh interval is initially set to 1s. All the results are averaged over 5 simulation runs with different random seeds. We vary mobility speed from 0m/s to 100m/s.

In the first experiment, UDP delivery ratio is presented in Fig.2. Fig.2(a) shows the delivery ratio with low CBR date rate (1 packet per second). The UDP delivery ratio in GPSR is quite good, almost 100%. This performance is indeed remarkable given the relatively high speed. Eventually, at top speed (100m/s) some packets are lost because of lack of a forwarding neighbor. This problem is easy to explain. Simple geometry shows that two nodes in neighbor columns moving at relative speed 200m/s are in contact at most for 1.5s. Recalling that hello refresh rate is 1s, and that some hellos may be lost because of interference, it is very possible that for some small fraction of the time a node has no forwarding neighbors. The packet is then lost! AODV in contrast
does not work well, and the UDP delivery ratio deteriorates monotonically with the increased mobility, with more than 30% loss at 100 m/s. This is expected because of repeated path breakage and failure to find a route.

Fig. 2(b) shows the delivery ratio for high rate UDP (40 packets per second). Surprisingly, GPSR collapses! The delivery ratio is even worse than in AODV for speed larger than 40 m/s. From simulation results we find that the major reason for this problem is the loss of hello packets due to interference. Since hello packets are broadcast with an “unreliable MAC” (no RTS/CTS/ACK), when the UDP rate is high and congestion builds up, hello packet mortality is high. This leads to inaccurate neighbor information. If
Fig. 3. TCP performance over deterministic motion

GPSR finds no neighbors in the forward directions, it initiates “perimeter routing” which can lead to loops, and to hop count timeout. In fact, in this experiments the lost packets all had very high hop count except packet losses at the source! Another experimental observation is that, when congestion builds up, most of the packets are dropped at the source node. This information could be useful to adjust (ie. reduce) CBR rate. We will further discuss this property later. To improve the chance that some hello messages are received, we increase the hello exchange rate from 1 hello per second in the initial GPSR to 1 hello per 0.2 second (named as GPSR(0.2)). Fig.2 shows that delivery ratio is significantly improved, especially for high CBR rate. However, high rate hello exchange brings more overhead. In Fig.2(b), the delivery ratio at speed 0m/s in GPSR(0.2) is slightly less than that for GPSR(1.0). From these results, high hello exchange rate is most effective when mobility speed is high.

Suspecting that in general there will be a trade off between routing efficiency and extra network O/H in the hello rate selection, we propose an adaptive hello interval scheme that increases hello rate based on mobility, also taking into account that hello packets will be lost due to interference. Specifically, we select the adaptive hello interval according to the following formula:

$$I = \frac{R}{k \times \text{speed}}$$

(1)

Where $I$ is the hello interval, $R$ is the transmission range and $k$ is a tunable parameter. The rationale behind the formula is the following. On average two randomly moving neighbors see each other in a window equal to $R$ meters. Thus, the contact interval is $R$/speed. During the contact interval, a node needs to send several hellos to announce its presence. The factor $k$ should be adjusted to balance the overhead and effectiveness and to account for hello
loss. In our simulation, \(k\) is set to 16. Additionally, there are two limits for hello interval, an upper limit of 2 second and a lower limit of 0.1 second. Thus, the adaptive hello interval used in the paper is \(max(min(I, 2), 0.1)\).

The results for GPSR with adaptive hello exchange rate, named as GPSR(ha), are also shown in Fig.2. Equipped with adaptive hello rate in GPSR, UDP delivery ratio keeps around 100% for low rate UDP, and it is much better than AODV in high rate UDP as well. For high rate UDP, GPSR(ha) is slightly less than GPSR(0.2) for speeds from 20m/s to 60m/s. A more aggressive choice of parameters should be explored. In all, however, the difference (<3%) is quite small.

After studying UDP performance, we turn to TCP performance in Fig.3. Since TCP consists of two way traffic and a data packet can collide with an ack packet, TCP traffic press much more stress on the routing. From Fig.3, TCP over GPSR is only moderately affected when nodes moves, even at very high speed, due to the GPSR robustness to mobility. The throughput of TCP over GPSR only drops 21\% from 0m/s to 100m/s and still achieves high throughput (around 230k at 100m/s), while TCP throughput over AODV drops by 50\% and degrades fast for high mobility, just about 120k at 100m/s. Due to the frequent route breakage and more packet losses from two way traffic, TCP never has the chance to perform well and the performance degrades quickly with mobility in AODV. We recall that Fixed-RTO was proposed in TCP as a remedy to path breakage. We note in Fig.3 that Fixed-RTO only has minimal impact on performance over AODV and GPSR (the Fixed-RTO curves are practically overlapped with original TCP curves).

We note that, as a difference from UDP, TCP does not perform too bad with GPSR(1.0). This is due to the fact that TCP does congestion control and thus limits the interference onto hellos. Nevertheless, a reduction from hello interval from 1s to 0.2s brings significant benefits for high mobility as shown in 3, however the performance is worse in low mobility. This calls for adaptation. GPSR with adaptive hello (GPSR-ha) eliminates the overhead problem at low mobility. TCP over GPSR(ha) traces the upper envelope of performance of both GPSR(1.0) in low mobility and GPSR(0.2) for high mobility. Therefore, it satisfies our quest to improve TCP performance under varying mobility.

### 2.2 Random Movement

In what follows, we study TCP performance in a general case where nodes move randomly. The random waypoint mobility model [15] is used in the simulation. Fig.4 illustrates the random moving topology. The sender and receiver are kept at fixed positions, while the remaining 40 nodes are moving randomly in a region of 1000m × 1000m. The speed of the nodes ranges from 10m/s to 90m/s.

The performance of UDP over GPSR and TCP with Fixed-RTO in this scenario is pretty similar to that observed in deterministic motion, so we omit their results here. Fig.5 presents TCP results over AODV and GPSR. As
expected, TCP over GPSR still outperforms TCP over AODV. However, its performance is by far worse than in the deterministic motion case. We discover that TCP over Geo-routing has a serious out-of-order (OOO) delivery problem in the random movement. In contrast, TCP over AODV suffers no significant OOO problems.

In GPSR, a route is selected on a packet by packet basis and thus can change very rapidly. If a new route is shorter or has lower delivery delay due to lighter load, data packets on the new route could arrive before packets on the old route. OOO packets cause throughput degradation. In fact, TCP receiver responds to OOO data packets with duplicate acks which potentially trigger fast retransmits leading to congestion window reduction and extra inefficiency. This OOO problem was discussed in [16] and some approaches for OOO detection and response were presented. However, the approaches proposed in [16] require the modifications of packet header and cooperations at TCP sender and receiver. In the paper we propose a novel approach, which only involves TCP receiver without modifying the packet header format and TCP sender. TCP receiver determines if a non-in-order packet has come from a different route by simply checking the TTL value in the packet header. If the packet is from a path different from that of the latest in-order packet, there was a route change and the OOO event is detected. The missing packets between this OOO packet and in-order packet could arrive latter. The receiver could wait for some time before issuing a duplicate ack. A challenge here is to estimate the "optimal" waiting window at the receiver. We propose to let the receiver passively estimate RTT and use it to decide the period for the waiting timer. The receiver could use the TCP timestamp option in the packet header to estimate RTT, though such an estimate may be inflated if the sender does not send data packets immediately after receiving an ack [17]. However, the waiting timer is only a coarse timer for predicting when the missing packets will arrive, thus RTT inflation errors can be tolerated.

The receiver computes the waiting time based on this RTT measurement. The following formula is used:

\[
S_{RTT}^r_{k+1} = \frac{7}{8} S_{RTT}^r_k + \frac{1}{8} RTT^r_{k+1}
\]

\[
RTT^\text{var}_{k+1} = \frac{3}{4} RTT^\text{var}_k + \frac{1}{4} |RTT^r_{k+1} - S_{RTT}^r_{k+1}|
\]

\[
RTO^r_{k+1} = S_{RTO}^r_{k+1} + 4 \times RTT^\text{var}_{k+1}
\]

Where $RTT^r$ is the RTT estimation at the receiver, $S_{RTT}^r$ is the smoothed RTT. $RTT^\text{var}$ and $RTO^r$ are RTT variance and waiting timer period at the receiver. This ack waiting timer is started after detecting OOO event, and is canceled after all missing packets arrived.

As shown in Fig.5, the OOO packet handling strategy (GPSR-OOO) enhances TCP performance by about 10%. Incidentally, we also tested OOO delivery in deterministic motion, however we did not find significant OOO delivery effects because all direct paths have the same length. Next, we en-
hance TCP over GPSR by adjusting hello intervals. Fig. 6 presents the TCP performance over GPSR with fast and adaptive hello exchange. Note: we only show results with OOO response. As expected, GPSR(0.2) only provides performance gain for high mobility, while GPSR(ha) integrates the advantages of GPSR in low mobility and GPSR(0.2) in high mobility. TCP over GPSR with adaptive hello interval is considerably better than TCP over AODV. The throughput of TCP over GPSR(ha) only drops from 315kbps at 10m/s to 225kbps at 90m/s (about 28% performance degradation), while AODV drops from 260kbps at 10m/s to 95kbps at 90m/s (about 63% performance degradation).

Fig. 4. Random Movement Topology

Fig. 5. TCP performance over random motion
3 Conclusion

We have studied TCP and UDP performance over Geo-routing in highly mobile ad hoc network. As expected, Geo-routing introduces substantial benefits. However, several Geo-routing parameters must be carefully tuned for achieving best performance. First, we have proposed a hello scheme adaptive to the mobility. Second, we have proposed a novel scheme of handling out-of-order delivery which requires TCP receiver-only modification. These two enhancements improve TCP performance by 50%, from 150kbps to 225kbps at diving bird speed (90m/s) in GPSR. In contrast, TCP over AODV delivers less than 100kbps at this speed.

References