A New Approach to Local Route Recovery for Multihop TCP in Ad Hoc Wireless Networks

Zhi Li and Yu-Kwong Kwok*

Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong**
ykwok@hku.hk

Abstract. The TCP (Transmission Control Protocol) is a critical component in an ad hoc wireless network because of its pervasive usage in various important applications. However, TCP's congestion control mechanism is notoriously ineffective in dealing with time-varying channel errors even for a single wireless link. Indeed, the adverse effects of the inefficient usage of wireless bandwidth due to the large TCP timers are more profound in a multihop communication session. In this paper, we design and evaluate local recovery (LR) approaches for maintaining smooth operations of a multihop TCP session in an ad hoc network. Based on our NS-2 simulation results, we find that using the proposed LR approaches is better than using various well-known ad hoc routing algorithms which construct completely new routes.
Keywords: wireless TCP, multihop communications, ad hoc networks, routing, local recovery.

1 Introduction

The TCP (Transmission Control Protocol) is the most widely used transport protocol and, more importantly, will continue to be a critical component when the Internet becomes completely pervasive in a wireless manner [6]. Unfortunately, due to the fact that TCP was not designed for a wireless environment, in which link transmission errors are the norm rather than the exception, its performance can be unacceptable under a time-varying communication channel [5]. Specifically, congestion is assumed to be the primary reason for packet losses in TCP. While this is true in wired networks, the throttling actions in a wireless environment can be detrimental. Indeed, unnecessary reduction in network load over a long period of time (TCP's timers are on the order of tens of seconds) leads to very inefficient use of the precious channel bandwidth and high delays.

Recently, many adaptive TCP approaches for various wireless environments have been suggested. The major objective of these schemes is to make TCP respond more intelligently to the lossy wireless links. According to [1,5], there

* This research was supported by a grant from the Research Grants Council of the HKSAR Government under project number HKU 7162/03E.
** Corresponding Author: Yu-Kwong Kwok

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 300–307, 2004.

are three major classes of wireless TCP approaches: end-to-end, link layer, and split-connection approaches. Unfortunately, all these previous approaches are only suitable for use in a single wireless link. For ad hoc networks where devices communicate in a multihop manner, these protocols are inapplicable because we cannot afford to have each pair of intermediate devices on a multihop route to execute these wireless TCP protocols [3,9]. Indeed, if a multihop ad hoc route is broken (e.g., due to deep fading in one of its links), the performance of a TCP session over such a route can be severely affected. The most obvious result is that the TCP sender will eventually discover such breakage after several unsuccessful retransmissions (i.e., after a long delay due to the large TCP timers) and then initiate a new session after setting up a new route. This can lead to unacceptably long delay at the receiver side.

In this paper, we study the performance of two local recovery approaches, which work by swiftly repairing the broken link using a new partial route. In Section 2, we describe our proposed local recovery approaches. Section 3 contains our simulation results generated by the NS-2 [10] platform.

2 The Proposed Approach

When the original route is down, we do not simply inform the source that the route cannot be used. Instead, we suppress the notification which is transmitted to the source by TCP, and then find a new partial route between the separated nodes to replace the broken part of the old route. Our approach, remedial in nature, is a *local recovery* (LR) technique [4]. The essence of LR is to shield the route error from the source in the hope that we can avoid incurring the excessive delay induced by TCP. Indeed, since the problem is found locally, the remedial work should be done locally.

For example, suppose that due to channel fading and nodes' mobility, the link between node N and $N+1$ is broken. Firstly, we suppress the upstream notification generated by TCP. Afterward, we find if the route table of node N has another route to node $N+1$. If there is a new route to the $N+1$ (i.e., the next node of such a route is not $N+1$), then the broken route is immediately repaired by using this route. If no such route exists, local recovery packets will be sent to repair the route.

A local recovery timer is set to make sure the local recovery process will not consume more time than to re-establish a new route by the source. Thus, if the local recovery timer is expired, we give up local recovery and make use of the full blown ad hoc routing protocol.

In the remedial process, a node N generates the local recovery route request (LRRREQ) packet, which includes the following information: type of the packet, local recovery source address, local recovery destination address, original destination address, local recovery broadcast identifier (ID), and hop count. Whenever node N generates a LRRREQ, the local recovery broadcast ID is increased by one. Thus, the local recovery source and destination addresses, and the local recovery broadcast ID uniquely identify a LRRREQ. Node N broadcasts the

LRRREQ to all nodes within the transmission range. These neighboring nodes then relay the LRRREQ to other neighboring nodes in the same fashion. An intermediate node, upon receiving the LRRREQ, first checks whether it has seen this packet before by searching its LRRREQ cache. If the LRRREQ is in the cache, the newly received copy is discarded; otherwise, the LRRREQ is stored in the cache and is forwarded to the neighbors after the following major modifications are done: incrementing the hop count, updating the previous hop node, and updating the time-to-live (TTL) field.

When node $N+1$ or some other intermediate node, which has a fresh route to the node $N+1$, receives the LRRREQ, it then generates a local recovery route reply (LRRREP) packet, which includes the following information: type of the packet, local recovery source address, local recovery destination address, original destination address, hop count, and TTL. The LRRREQ is then unicast to the local recovery source along the reverse path until it reaches the local recovery source. During this process, each intermediate node on the reverse path updates its routing table entry to the local recovery destination and original destination.

Although the new partial route is found from node N to node $N+1$, updating is needed for the original route. As described above, there are two cases where updating of the original route must be done. The first case is the event that the local recovery destination receives the LRRREQ. The second case is the event that an intermediate node gets the LRRREQ and it has a fresh route to the local recovery destination in its routing table.

According to the different directions, forward and backward updatings are carried out. The forward updating process is triggered by receiving the update packet, which contains the following information: type of packet, update destination address, original destination address, hop count, and TTL. The backward updating process is triggered by receiving the LRRREP packet. In any updating, the original route should be re-established. In the first case, only backward updating is done, while in the second case, both forward and backward updatings are needed.

In the former case, node $N+1$ receives the LRRREQ, and thus, backward updating is done through the route of nodes $N+1$, 3, 2, 1, N. In the latter case, forward updating is done through the route of nodes 2, 3, $N+1$, while backward updating is done through the route of nodes 2, 1, N. The detailed updating process is as follows: when node 2 receives the LRRREQ and it has a route entry to node $N+1$, node 2 sends the update packet to node 3 according to the route entry to node $N+1$. Upon receiving the update packet, node 3 should update the route entry to the original destination node D and then check if it is the local recovery destination. The same forward updating process continues until the update packet is received by the local recovery destination. On the other hand, LRRREP is sent to node 1 following the reverse route. Upon receiving the LRRREP, node 1 should update the route entry to the original destination node D and then check if it is the local recovery source. The same backward updating process continues until the LRRREP is received by the local recovery source.

This variant of our approach is similar to the mechanism we described above. The only difference is that the goal of route reconstruction is to find a new partial route from node N directly to the destination.

3 Performance Results

In our study, we use packet level simulations to evaluate the performance of TCP in ad hoc networks. The simulations are implemented in Network Simulator (NS-2) [10] from Lawrence Berkeley National Laboratory (LBNL) with extensions for wireless links from the Monarch project at Carnegie Mellon University [2]. The simulation parameters are as follows:

- number of nodes: 50;
- testing field: $1500m \times 300m$;
- mobile speed: uniformly distributed between 0 and MAXSPEED (we choose MAXSPEED to be 4, 10, 20, 40, 60m/s, respectively);
- mobility model: *modified* random way point model [12];
- traffic load: TCP Reno traffic source;
- radio transmission range: 250m;
- MAC layer: IEEE 802.11b.

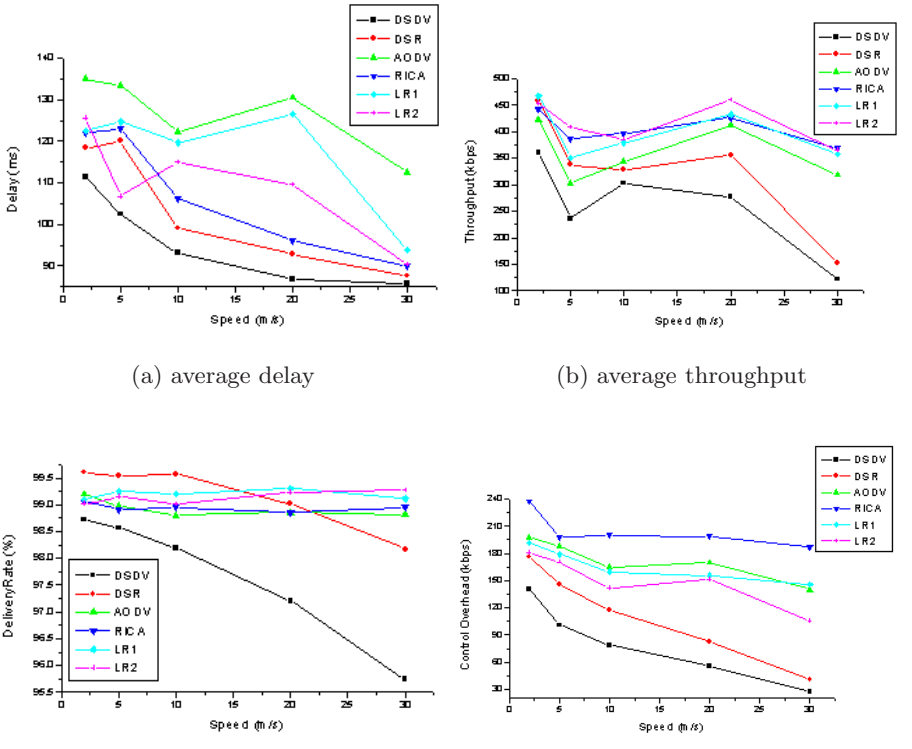
Each simulation is run for 200 seconds and repeated for ten times. We compared four protocols in our simulations. They are DSR (Dynamic Source Routing) [7], AODV (Ad Hoc On-Demand Distance Vector) [11], LR1 and LR2. LR1 is the local recovery protocol in finding the new route between node N to the destination. LR2 is the local recovery protocol in finding the new route between node N and node $N + 1$.

To evaluate TCP performance in different routing protocols, we compare them using four metrics:

1. Average End-to-End Delay: the average elapsed time between sending by the source and receiving by the destination, including the processing time and queuing time.
2. Average Throughput: the average effective bit-rate of the received TCP packets at the destination.
3. Delivery Rate: the percentage of packets reaching the destination (note that some packets are lost during the route breakage and the route reconstruction time).
4. Control Overhead: the data rate required by the transportation of the routing packets.

Our first set of simulation results are summarized in Figure 1. We compare the performance of the LR approaches against that of several well-known ad hoc routing protocols: AODV (Ad-Hoc On-Demand Distance Vector) [11], DSR (Dynamic Source Routing) [7], DSDV (Destination Sequenced Distance Vector) [11], and RICA (Receiver-Initiated Channel Adaptive) protocols [8].

Firstly, we find that TCP is idle most of the time when used with the DSDV and DSR routing protocols. On the other hand, other routing protocols can cooperate with TCP quite well. It should be noted that DSDV is table-driven routing algorithm. When the source has no route to the destination, it uses a long time to find a new route. This frequently leads to TCP timeout. Furthermore, the nodes are moving during almost the whole simulation time. Consequently, routes in the table can be stale very quickly and cannot be used. As for DSR, it is an on-demand algorithm, even though it has a route cache containing routes. When using DSR, a mobile device first checks if it has a route to the destination in the cache. Again, similar to the case of DSDV, the routes in the cache become stale very quickly and thus, new routes have to be found. However, as DSR is an on-demand algorithm, it can generally respond much faster than DSDV to find new routes.



(a) average delay

(b) average throughput

(c) average rate of successful delivery of packets

(d) average control overhead

Fig. 1. Protocol performance.

Figure 1(a) shows the average end-to-end delay of each protocol. It should be noted that the delay is effective delay—the delay of the packets that actually

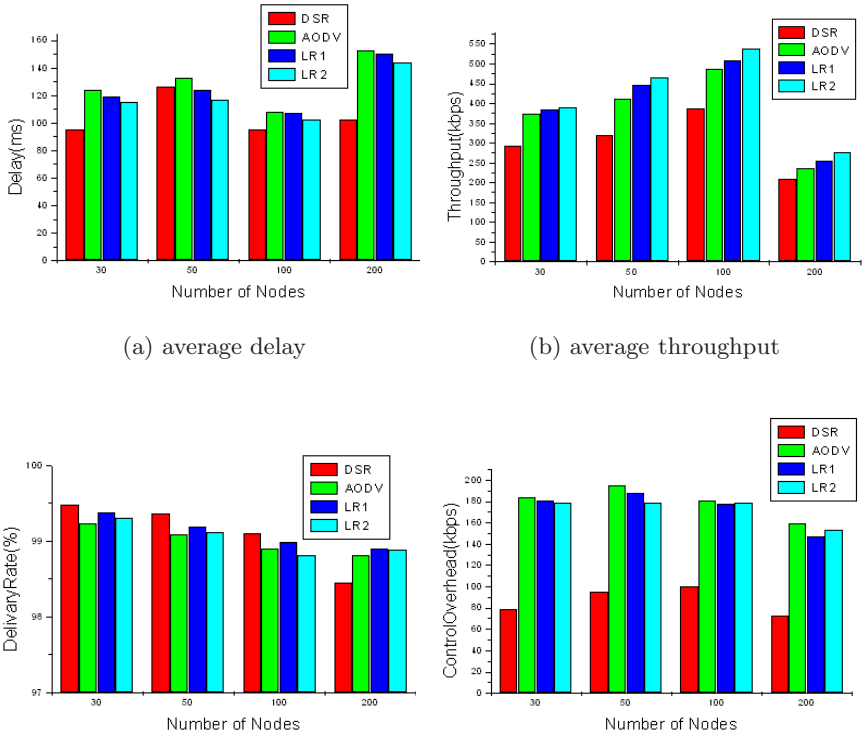
arrive at the destinations. We can see that DSDV has the lowest delay in all protocols. The reason is that each device has a table to contain routes. DSR has higher delay than DSDV because DSR caches only recently used routes. Comparing the other four routing protocols, where TCP almost does not become idle, RICA has the lowest delay and AODV has the highest delay. As RICA always chooses the best route, and the LR approaches can automatically recover the broken route locally, they generate smaller delays compared with AODV.

Figure 1(b) depicts the TCP throughput over the simulation time. Obviously, since DSDV and DSR have much idle time, they have lower throughputs. The LR approaches have higher throughputs than AODV due to the local repairing mechanisms. In particular, LR2 exhibits a better performance than LR1 because the time consumed by node N to find node $N + 1$ is less than that to find the destination on the average. Figure 1(c) shows the delivery rate. DSDV's delivery rate decreases dramatically with increasing device speed. This is because when the speed increases, stale routes are more common. This detrimental effect of mobility also applies to DSR. In the other four routing protocols, the delivery rate has little change with increasing speed. Figure 1(d) shows the control overhead required by the transportation of the routing packets. Again, because of idleness, DSDV and DSR have lower control overheads. RICA has the highest control overhead. The reason is that RICA must transmit CSI (channel state information) [8] packet to assist finding the best path. The LR approaches have less control overheads than AODV because when the route is broken the LR approaches need not inform the source to find a new route. In summary, the LR approaches are the best routing protocols for integrating with TCP in an ad hoc network.

Another set of results is about the evaluation of the TCP performance against different number of nodes (30, 50, 100, and 200 nodes). The mean mobile speed is fixed at 5 m/s and the maximum speed is fixed at 10 m/s. As can be seen in Figure 2, using the Local Recovery algorithms can greatly shorten the end-to-end delay from source to destination and lead to a higher throughput. LR1 and LR2 have higher throughputs than AODV and DSR. Furthermore, LR1 and LR2 have lower delay than AODV. In general, LR2 outperforms LR1. We can draw the following conclusions.

- Local Recovery suppresses the notification to the source so that the saved time can be utilized for local construction of new routes.
- The source does not need to setup a new route so that the buffered packets in the intermediate nodes need not be sent again.
- The unnecessary TCP timeout can be avoided in the source.

However, DSR has the lowest delay in all compared protocols. This is because the delay is calculated by considering the received packets only. In fact, DSR has much longer idle time. For the same reason, DSR has the lowest control overhead. In general, LR has lower control overhead than AODV. Finally, we can see that LR has a higher delivery rate than AODV. Furthermore, in LR and AODV, the delivery rate does not change much with the the number of nodes.



(a) average delay

(b) average throughput

(c) average rate of successful delivery of packets

(d) average control overhead

Fig. 2. Simulation results for various number of nodes.

We also consider the route setup delay of the protocols. The results are shown in Figure 3. In general, DSR has the lowest setup delay in all compared protocols. This is because the DSR source has the route cache to set up connection immediately. However, if a suitable cached route cannot be found, DSR will use a long time to find a new route. Moreover, the setup delay is rather unacceptable, e.g., more than 6 sec. In that case, the route failure causes the TCP sender to become idle until a new trial begins after a long time. AODV and the LR approaches have nearly the same route request and reply mechanisms. They require some time to find the route to set up connection. In particular, LR1 and LR2 have the same setup delay. When the number of node is small, i.e., 30 or 50, AODV and the LR approaches have similar setup delay. However, with a larger number of nodes—100 or 200, the setup delays between AODV and LR become obviously different. The setup delay of the LR approaches are smaller than that of AODV. The reason is that with increased number of nodes, the route re-establishment process is more frequent, especially when TCP synchronization packet has not been received by the destination. Since the LR approaches can suppress the route

error notification to the source and locally recover the route, the unnecessary timeout can be avoided or the duration of timeout is much reduced in the TCP source.

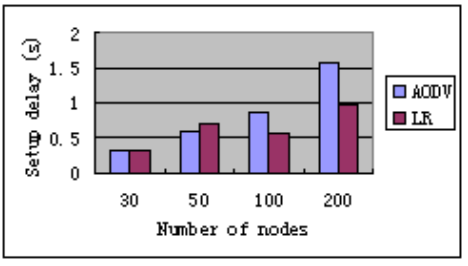


Fig. 3. The proposed local recovery algorithm.

References

1. H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, Dec. 1997.
2. J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. MOBICOM*, pp. 85–97, Oct. 1998.
3. 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*, vol. 8, no. 1, pp. 34–39, Feb. 2001.
4. R. Duggirala et al., "Performance Enhancements of Ad Hoc Networks with Localized Route Repair," *IEEE Trans. Computers*, vol. 52, no. 7, pp. 854–861, July 2003.
5. H. Elaarag, "Improving TCP Performance over Mobile Networks," *ACM Computing Surveys*, vol. 34, no. 3, Sept. 2002.
6. S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communications Review*, vol. 24, no. 5, pp. 10–23, 1994.
7. D. B. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, T. Imielinski and H. Korth (eds.), Chapter 5, Kluwer Academic Publishers, 1996.
8. X.-H. Lin, Y.-K. Kwok, and V. K. N. Lau, "A Quantitative Comparison of Ad Hoc Routing Protocols with and without Channel Adaptation," *IEEE Trans. Mobile Computing*, vol. 3, no. 4, Oct.-Dec. 2004.
9. D. A. Maltz, J. Broch, J. Jetcheva, and D. B. Johnson, "The Effects of On-Demand Behavior in Routing Protocols for Multihop Wireless Ad Hoc Networks," *IEEE J. Selected Areas in Comm.*, vol. 17, no. 8, pp. 1439–1453, Aug. 1999.
10. The UCB/LBNL/VINT Network Simulator (NS), URL:`http://www.isi.edu/nsnam/ns/`, 2003.
11. C. E. Perkins, E. M. Royer, and S. R. Das, "Ad Hoc On-Demand Distance Vector(AODV) Routing," *IETF Internet Draft*, draft-ietf-manet-aodv-10.txt, 2002.
12. J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *Proc. INFOCOM 2003*.