Local Route Recovery Algorithms for Improving Multihop TCP Performance 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. TCP (transmission control protocol) will for sure continue to be the major transport protocol in wireless environments, due to its large install-base in existing applications. Indeed, in future ad hoc wireless networks where devices communicate among each other over multihop routes, TCP is expected to be the prominent protocol for data exchange (e.g., multihop wireless FTP). However, there is a severe performance degradation in using TCP over a wireless link. Despite that there are a large number of wireless TCP adaptation schemes proposed in the literature, improving TCP performance over a multihop wireless route is still very much unexplored. In this paper, we propose local route recovery approaches for improving the performance of multihop wireless TCP. Our simulation results generated by NS-2 indicate that the local recovery approaches outperform complete replacement approaches (i.e., using full-blown ad hoc routing protocols such as AODV and DSR) in terms of end-to-end delay, throughput, packet delivery rate, and control overhead.

Keywords: wireless TCP, multihop communications, ad hoc networks, routing, local recovery.

1 Introduction

In the past few years, we have witnessed that the Internet has extended its reach to the wireless networking environments. Thus, traditional Internet protocols [4, 7, 8] are also heavily used over wireless links. However, a straightforward migration of such protocols to wireless networks will result in poor performance [6]. In particular, the TCP (transmission control protocol), which has many salient features that are useful in a wired network, needs significant modifications in order that it can deliver packets over the wireless links efficiently. Indeed, the congestion control mechanisms are particularly problematic in wireless environments.

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

M. Danelutto, D. Laforenza, M. Vanneschi (Eds.): Euro-Par 2004, LNCS 3149, pp. 925–932, 2004.

Congestion is assumed to be the primary reason for packet losses in wired networks. When packet losses are detected, the sender's TCP exponentially throttles down the congestion window size before retransmitting lost packets. It then backs off its retransmission timer, and enters the congestion avoidance phase. All these mechanisms are aimed at reducing the load injected into the network. However, wireless networks have different characteristics compared with wired networks, such as high bit error rate and occasional blackout. Thus, packet losses can be due to channel errors rather than congestion. 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, 6], there 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, 11]. 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 [12] platform. We give some concluding remarks in Section 4.

2 The Proposed Approach

2.1 Overview

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 [5, 13]. 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 exits, 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. Figure 1 shows the flow chart of the local recovery algorithm. We explain the whole process in detail below.

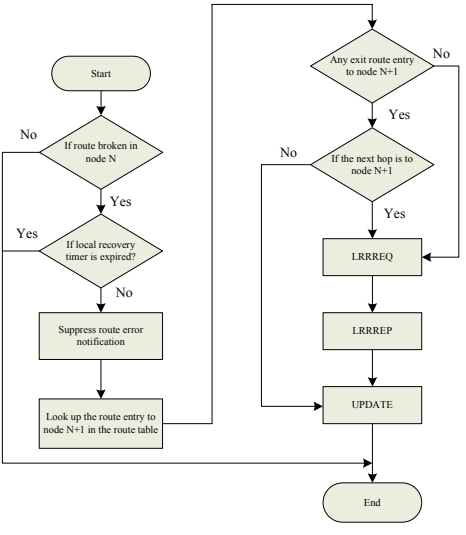


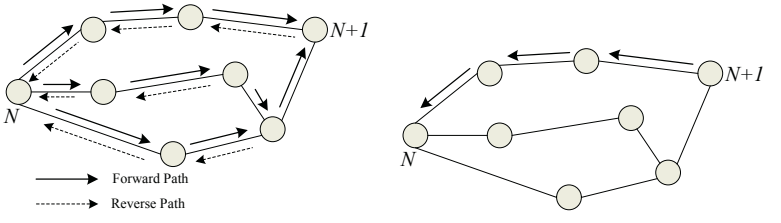
Fig. 1. The proposed local recovery algorithm.

2.2 Discovering Local Recovery Route

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. Figure 2(a) illustrates how the LRRREQ propagates and how the reverse paths are set up.

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. Figure 2(b) illustrates the process where the LRRREP is unicast through the reverse path.



(a) propagation of the LRRREQ (b) propagation of the LRRREP

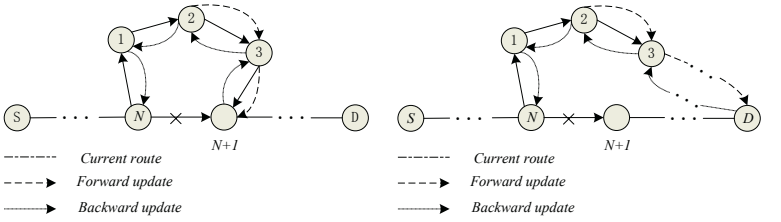
Fig. 2. The route recovery process.

2.3 Route Updating

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.

Figure 3(a) illustrates how the updating process is carried out in the two cases. In the former case, node $N+1$ receives the LRRREQ, and thus, backward



(a) forward and backward updating (b) local recovery algorithm for finding a new partial route directly to the destination

Fig. 3. Illustration of the two route recovery approaches.

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, LRRRREP 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.

2.4 Local Recovery of a Route to Destination

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. Figure 3(b) illustrates this algorithm.

3 Performance Results

3.1 Simulation Environment

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) [12] 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 [14];
- 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) [9], AODV (Ad Hoc On-Demand Distance Vector) [10], 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 $N + 1$.

3.2 Performance Metrics

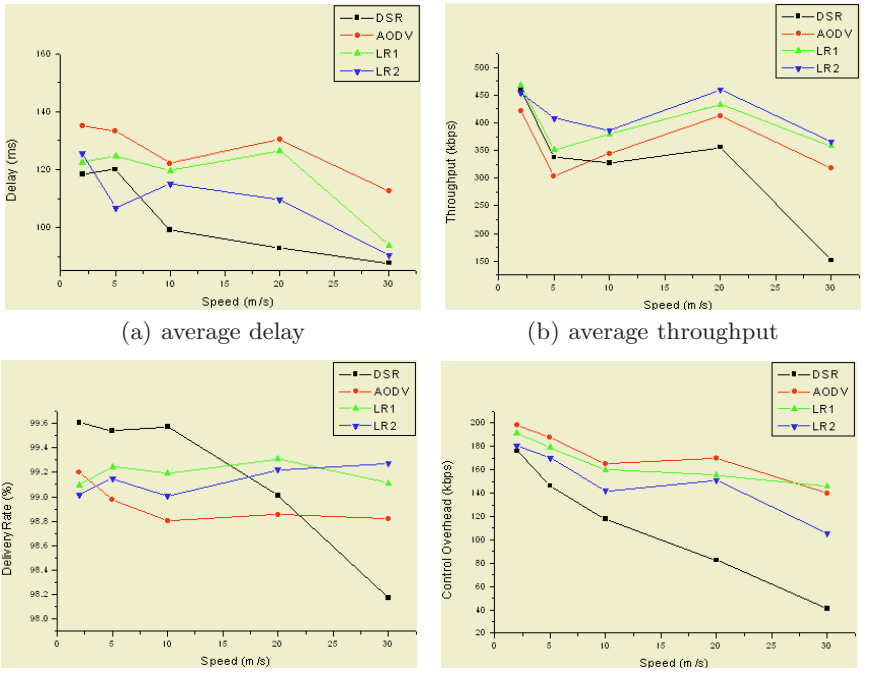
To evaluate TCP performance in different routing protocols, we compare them using four metrics:

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.

3.3 Simulation Results

Due to space limitations, our simulation results are summarized in Figure 4. Compared with the AODV protocol, the two LR protocols have lower end-to-end delays, higher throughput, and lower control overhead. The reason is that when the route is broken, the local recovery process can efficiently reconstruct the route, without incurring excessive TCP time-out delays. LR2 exhibits a better performance than LR1. The reason is that on average, the time consumed by node N to find node $N + 1$ is less than that to find the destination.

DSR has the worst performance among all compared protocols. Since it drops much more packets than other protocols. In particular, when the mobility is increased, the delivery rate decreases rapidly. However, the other three protocols can keep a stable delivery rate with increasing speed.



(a) average delay (b) average throughput

(c) average rate of successful delivery of packets (d) average control overhead

Fig. 4. Simulation results.

4 Conclusions

In this paper, we study the problem of improving multihop wireless TCP performance in an ad hoc network. TCP performance over a single wireless link is notoriously poor due to its long retransmission delays in error recovery. The long delay problem is even more acute in a multihop wireless environment. We compare four approaches in route error recovery: two of them are based on complete replacement of the old route, while the other two are based on local reconstruction of the broken route. Our simulation results generated by the NS-2 platform using TCP Reno traffic sources show that the local recovery approaches significantly outperform the complete replacement approaches.

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. D. Comer, *Internetworking with TCP/IP*, vols. 1-3, Prentice Hall, 1991.
5. 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.
6. H. Elaarag, "Improving TCP Performance over Mobile Networks," *ACM Computing Surveys*, vol. 34, no. 3, Sept. 2002.
7. K. Fall and S. Floyd, "Simulation Based Comparisons of Tahoe, Reno, and SACK TCP," *ACM Computer Communications Review*, vol. 26, no. 3, pp. 5–21, 1996.
8. S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communications Review*, vol. 24, no. 5, pp. 10–23, 1994.
9. 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.
10. 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.
11. 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.
12. The UCB/LBNL/VINT Network Simulator (NS), `http://www.isi.edu/nsnam/ns/`, 2003.
13. D. Tian and N. D. Georganas, "Energy Efficient Routing with Guaranteed Delivery in Wireless Sensor Networks," *Proc. WCNC 2003*, vol. 3, no. 1923–1929, 2003.
14. J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *Proc. INFOCOM 2003*.