An Enhancement Scheme for TCP over Mobile Ad Hoc Networks

Jin-Hee Choi, See-Hwan Yoo and Chuck Yoo
Department of Computer Science and Engineering
Korea University
Email: {jhchoi, shyoo, hxy}os.korea.ac.kr
Telephone: +82-2-3290-3639
Fax: +82-2-922-6341

***Abstract*— In a Mobile Ad Hoc Network, temporary link failures and route changes occur frequently. With the assumption that all packet losses are due to congestion, TCP performs poorly in such an environment. This paper proposes a new mechanism called TSR, TCP-aware Source Routing, which can improve TCP performance in wireless ad hoc networks. TSR adds a hold state to an existing routing protocol to reduce consecutive timeouts, retransmissions, and out-of-ordered packets in TCP. In our simulation study, TSR achieves up to a 60% improvement in performance, without requiring any TCP stacks in end systems to be modified.**

***Index Terms*— ad hoc network; performance evaluation; routing protocol; DSR ;TCP**

I. Introduction

A mobile ad hoc network is a network, in which a group of mobile computing devices communicate among themselves using wireless radios, without the aid of a fixed networking infrastructure. Due to its dynamic property, mobile ad hoc networks have gained a lot of attention lately as a way of providing continuous network connectivity to mobile computing devices in various areas. However, because the topology of networks changes dynamically, route changes frequently and failure to find a valid route promptly would result in a significant drop in performance. When TCP is used as transport protocol, such a drop in performance is rather expected because packets sent to an invalid route are all lost.

Specifically, TCP performance can suffer due to the following reasons:

- Packet loss due to broken routes can result in the counterproductive invocation of TCP's congestion control mechanisms.
- Selecting an invalid alternative path while reestablishing a broken one, this would result in consecutive timeout.
- Longer RTO(Retransmission TimeOut) value, which results from consecutive timeout.
- Out-of-ordered TCP segments.

So, there have been a lot of research to address the routing problem in mobile ad hoc networks[1], [2], [3], [4], [5].

Routes are broken frequently as the movement of mobile terminals gets faster. Frequent route change makes TCP to have multiple packets losses. TCP has only one way to recover from multiple packets losses, and that is to expire the retransmission timer. But a consecutive timeout makes the RTO (Retransmission TimeOut) value exponentially back off [6]; and longer RTO reduces link utilization and decreases throughput dramatically [7]. So we think that consecutive timeout is a key factor that affects the TCP performance.

In this research, we explore a new way to enhance TCP performance by minimizing the consecutive timeout in the case of frequent route changes, without modifying TCP stack.

The rest of this paper is organized as follows. Section 2 describes the existing works to improve performance in TCP. Section 3 gives an explanation about the necessity of TCP-aware routing. Section 4 describes our TSR mechanism. Section 5 provides the results of our simulation and its analysis. Finally, Section 6 concludes the paper.

II. Related work

Recent studies have addressed the TCP performance problems caused by route failures in a mobile ad hoc network. Since TCP assumes that all packet losses are due to network congestion, although the cause of packet losses is route failure, TCP performs congestion control. Because this behavior is the major reason by which it shows dramatic drop in TCP performance, many studies try to distinguish between route failure and network congestion and thereby improve the performance of the routing protocols.

Chandran et al.[2] proposed a feedback-based scheme called TCP-Feedback or TCP-F. In this scheme, when an intermediate node detects route failure, it explicitly sends a RFN (Route Failure Notification) message to the TCP sender. On receiving the RFN, the TCP sender suspends packet transmissions and freezes all states, including the RTO value and CWND (the size of Congestion WiNDow). When an intermediate node learns of a new route to the destination, it sends a RRN (Route Reestablishment Notification) message to the TCP sender, which then restores its previous state and

resumes transmission. The conclusion of the study was that the average route repair time has a major impact on TCP performance.

Holland et al.[1] proposed a scheme that uses ELFN (Explicit Link Failure Notification). Also, in this scheme, when the TCP sender is informed of a link failure, it freezes its state as in TCP-F. After that, the TCP sender sends out packets at regular intervals to determine if an alternative route is available.

Dyer et al.[4] proposed a heuristic scheme called fixed RTO. In this scheme, a heuristic was employed to distinguish route failure and congestion. When timeouts occur contiguously, the sender assumes a route failure occurred, rather than network congestion. The unacknowledged packet is retransmitted again but the RTO is not doubled a second time. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged.

III. TCP-aware routing

Existing routing protocols [8], [9], [10], [11] in ad hoc networks are designed to reestablish broken routes as soon as possible. However, since there is a delay between when a route is broken and when the source node is informed of the route failure, the packets sent through the broken route will be lost. If UDP is the transport protocol, such burst packet losses would not cause a serious problem in performance. When the source node has alternative routes, the source node keeps sending the packets out through one of them. If the chosen alternative route is again invalid, TCP will suffer a series of consecutive timeouts.

If timeouts occur contiguously, the RTO value of TCP's retransmission timer is exponentially backed off. Since all packet transmissions are suspended until the timer is expired, link utilization is significantly reduced. Figure 1 shows the sequence number when consecutive timeouts occur from 0 to 47 seconds and from 70 to 155 seconds. Packet transmission is suspended for a long period (about 80 secs); so that the entire throughput degrades very poorly.

A key observation is that invalid routes cause consecutive timeout. Also, note that the chance of invalid routes is quite high because the source node depends on the routing information from its neighbors that are not updated with the route failure. The chance will be higher when mobile terminals move faster in mobile ad hoc networks.

Also note that the congestion control of TCP does not help improve the performance because too many packets are lost and the timer expiration is the only way to start retransmission.

Therefore, we need a mechanism that prevents the alternative routes from being invalid, when the source node is informed of a route failure. So, we propose the mechanism called TSR that consists of adding a hold state

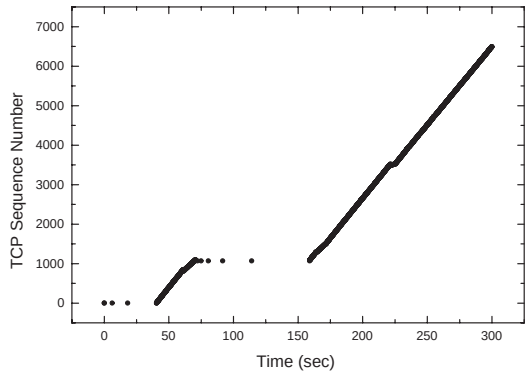

Fig. 1. Impact of consecutive timeout.

and refreshing alternative routes in its routing table. Here is how TSR works. When the source node is informed of a route failure, TSR checks whether the transport protocol is TCP or not. If it is, TSR makes the routing protocol transits to the hold state. Hold state is the state in which the routing protocol does not forward any data packets. In the hold state, TSR starts probing alternative route, and such probing is performed in parallel. In order to get the fastest path, the destination node should reply to such probing packets as soon as possible. And finally, when TSR gets the fastest route by n-parallel probing, it escapes from its hold state and starts packet transmission by using that path.

Of course, when the source node is informed of a route failure, there is chance that the alternative routes are valid, which makes holding the state of the routing protocol unnecessary. However, because multiple packet losses are so fatal to TCP performance [1], [5], we believe that TSR would be effective in most cases, which is proved in our experiments.

An advantage of TSR is that it does not require the modification of TCP stack. In mobile ad hoc network, it is accepted that mobile terminals need a new routing protocol because of the unique nature of ad hoc networks. But, it is desirable that the transport layer stays independent of the underlying networks. To the best of our knowledge, all the previous routing protocols in ad hoc network required TCP stack modification, and TSR is the first attempt that achieves no modification.

IV. Design and implementation of TSR

We add a hold state to DSR [8], one of the existing routing protocols, in order to implement TSR. The reason we choose DSR is that 1) it is widely accepted in ad hoc networks, 2) its architecture fits the framework of TSR. Although we implement TSR by adding a hold state on DSR, we believe that TSR can be applied to any other routing protocol.

The DSR protocol is an on-demand routing protocol that is based on the concept of source routing. The protocol consists

of two phases: route discovery and route maintenance. When a mobile node wishes to send a packet to a destination node, it first checks its route cache to determine whether it already has a route to the destination node. If the mobile node does not have such a route, it initiates route discovery by broadcasting a ROUTE_REQUEST packet. When either the destination node or an intermediate node that contains a valid path to the destination in its route cache, receives the route request packet, a ROUTE_REPLY packet is generated. As the source node receives ROUTE_REPLY, route discovery is done, and DSR initiates packet transmission. Route maintenance is accomplished through the use of ROUTE_ERROR packets. ROUTE_ERROR packets are generated at a node when the data link layer encounters a fatal transmission problem. Nodes that receive ROUTE_ERROR packets remove these invalid paths from their route cache.

In order to implement TSR, we define two additional control packets: ROUTE_PROBE and REPLY_PROBE. The ROUTE_PROBE packet is used to probe fresh paths when the source node receives a ROUTE_ERROR packet. And REPLY_PROBE packet is used to reply to the ROUTE_PROBE packet.

TSR consists of probing routes in n-parallel and adding a hold state. When DSR reports ROUTE_ERROR, TSR does not immediately select an alternative path in the route cache, but transits to a hold state and start parallel probing. The probing process is as follows. 1) TSR selects n paths from the route cache (we select 3, heuristic value). 2) TSR sends a ROUTE_PROBE message to each path. 3) Destination nodes, which have received a ROUTE_PROBE, send REPLY_PROBE messages using reverse paths. 4) When the first REPLY_PROBE message is received, TSR releases its hold state and resumes packet transmission using that path. It is because the path that has the shortest round trip time is the best path at that given moment.

TSR has two ways to escape from the hold state and return to the DSR state. Normally, it transit to $hold\ state \triangleright select\ state \triangleright DSR\ state$. However, if all ROUTE_PROBE or REPLY_PROBE messages are lost in immediate nodes, TSR may wait in the hold state infinitely. So, we introduce the TSR timer that is used to back to the DSR state. This TSR timer uses timeout value (TTO) be set to

$$TTO = TCP's \quad RTO + \alpha, \tag{1}$$

where a is a delay variance factor with a recommended value of 2. (Since TCP's SRTT adapt itself to mobile ad hoc networks slowly in a way [12], we add a, a delay variance factor, to RTO).

The transition diagram of TSR is illustrated in Figure 2.

V. Simulation and Analysis

The simulation study is done in the NS2 network simulator [13]. NS-2 is a discrete event simulator that was developed as part of the VINT project at the Lawrence Berkeley

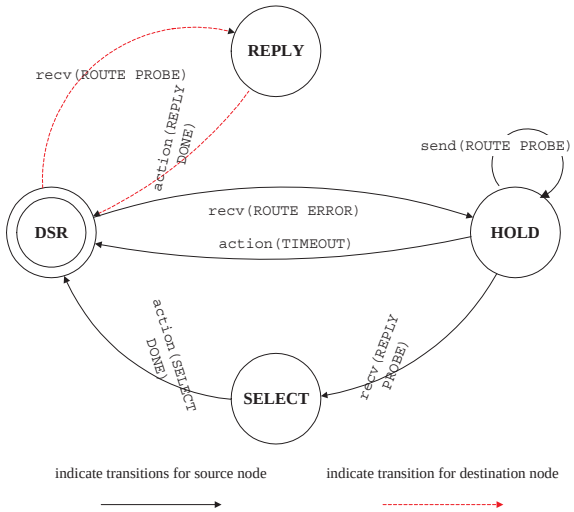

Fig. 2. State transition diagram.

National Laboratory. The extensions implemented by the CMU Monarch project[14] enable it to simulate mobile nodes connected by wireless network interfaces. We extended the NS-2 DSR protocol implementation to include TSR as described in Section 4. All results are based on a network configuration consisting of TCP-Reno over IP on an IEEE 802.11[15] MAC layer. Our network model consists of 30 nodes in a 1500x300 meter flat, rectangular area. Each node uses a wireless channel model with a transmission range of 250m. The nodes move according to the random waypoint mobility model. All of our simulation results are based on the average value of 50 scenarios.

We measured the throughput of TCP with and without TSR, varying the mean speed from 2 to 30m/s; and we got the count of the retransmission timer's backoff in each case. Also, we measured the sum of each mobile node's throughput, in case the network has 5 and 10 TCP sessions. The results are shown in Figure 3, 4 and 5

Figure 3 shows the throughput of TSR compared to DSR. Significant improvements in throughput can be observed in the best case (e.g., when the mean speed is 30 m/s). The improvement in throughput is largely due to avoiding consecutive timeouts. We can see that the throughput gain increases as the mean speed increases. It is because, as the mobility of nodes in mobile ad hoc network increases, the probability of route failures gets higher. In the current design, TSR initiates only when the protocol of the transport layer is TCP and the routing protocol of the source node received a ROUTE_ERRROR message.

Figure 4 shows the number of timeouts and backoffs depending on the routing protocol: TSR and DSR. In general, the number of timeouts is reduced, the TCP throughput increases. However, Figure 4 shows that if timeouts occur contiguously, no matter how many timeouts may be,

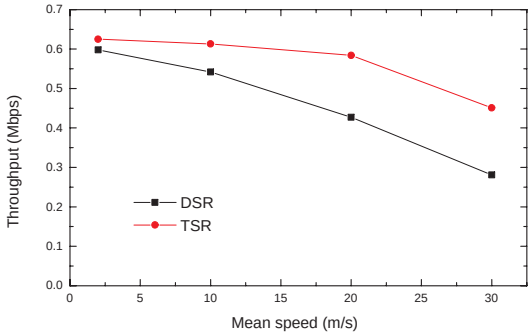

Fig. 3. Comparison of TCP performance based on routing protocol (DSR, TSR).

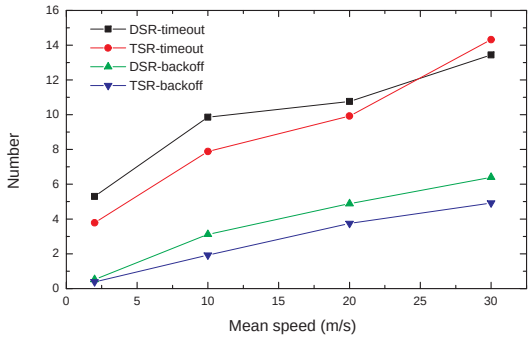

Fig. 4. Comparison of the number of timeout and backoff based on routing protocol (DSR, TSR).

throughput is reduced. Actually, the number of timeouts for TSR is lower than DSR except when the speed of movement is 30 m/s. However, TSR has higher throughput than DSR even at 30 m/s because the RTO value increased due to the selection of invalid paths, consequently decreasing throughput.

Finally, Figure 5 demonstrates the effect of the number of TCP sessions. It says that when there are 10 TCP sessions in ad hoc networks, TSR does not achieve an outstanding improvement in performance compared with DSR. There are two reasons. The first one is the MAC contention. All mobile terminals contend for the right to transmit frames, in an IEEE 802.11 network. The winner could transmit frames, but others should await the next contention during random intervals (CSMA/CA)[15]. The more sessions are in mobile ad hoc networks, the more data and MAC contention grows. Thus, when the number of TCP sessions increases, MAC contention becomes a bottleneck before TCP timeouts. The second reason is that the routing table tends to be refreshed often as the number of TCP sessions increases. It is just because more packets are exchanged. Therefore, there is less chance that the alternative routes become invalid.

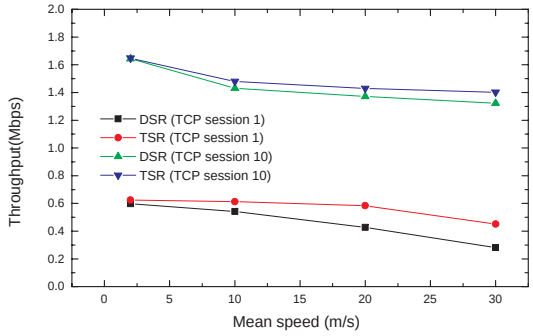

Fig. 5. Comparison of TCP performance based on the number of session.

VI. Conclusion

Normal TCP performs poorly in mobile ad hoc networks because of frequent route changes. In our scheme, TSR does not send out packets until it discovers a reliable route. By holding the state of routing protocol, TSR reduces consecutive timeouts, retransmissions, and out-of-ordered packets in TCP.

This protocol achieves up to a 60% improvement in performance compared with DSR. Also, it shows more outstanding improvements in performance as the mobility of mobile terminal increases. We also experimented with UDP and got a similar result as TCP. In addition, TSR enhances TCP performance just by modifying the routing protocol without any modification of TCP.

References

[1] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *in Proc. ACM MOBICOM 2001*.

[2] K. Chadran, S. Raghunathan, S. Venkatesan and R. Prakash, "A feedback based scheme for improving TCP performance in ad-hoc wireles networks," *in Proc. 18th International Conference on Distributed Computiong Systems.*

[3] A. Ahuja, S. Agarwal, J. P. Singh and R. Shorey, "Performance of TCP over different routing protocols in mobile ad-hoc networks," *in Proc. IEEE VTC 2000.*

[4] T. Dyer and R. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," *in Proc. ACM MOBIHOC 2001.*

[5] Y. Zhang and F. Wang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response," *in Proc. ACM MOBIHOC 2002.*

[6] R. Stevens, "TCP/IP Illustrated, Volume I," *Addison-wesley*, 1994, pp. 304-306.

[7] N. Vaidya, "Mobile ad hoc networks: routing, MAC and transport issues," *ACM MOBICOM 2001, tutorial.*

[8] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing.*

[9] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," *in Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications.*

[10] Z. J. Haas and M. R. Pearlman, "ZRP: A hybrid framework for routing in ad hoc networks," *Addison-wesley*, 2001, chap. 7.

[11] M. S. Corson and V. Park, "Link reversal routing," *Addison-wesley*, 2001, chap. 8.

[12] M. K. Kim and B. Noble, "Mobile network estimation," *in Proc. ACM MOBICOM 2001.*

[13] K. Fall and K. Varadhan, "NS notes and documentation," *the VINT Project, UC Berkeley, LBL USC/ISI, and Xerox PARC, available from http://www-mash.cs.berkeley.edu/ns*, Nov. 1997.

[14] CMU Monarch Group, "CMU Monarch extensions to the NS-2 simulator," *Available from http://monarch.cs.cmu.edu/cmu-ns.html*, 1998.
[15] IEEE 802.11 working group, "IEEE Std 802.11, 1999 Edition," *available from http://standards.ieee.org/catalog/olis/lanman.html*, ISO/IEC 8802-11: 1999.