TCP-aware Source Routing in Mobile Ad Hoc Networks

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

***Abstract*— Temporary link failures and route changes occur frequently in mobile ad hoc networks. Since TCP assumes that all packet losses are due to network congestion, TCP does not show satisfactory performance in ad hoc networks. In this paper, we propose a simple and new mechanism called TSR, TCP-aware Source Routing, which can improve TCP performance in mobile ad hoc networks. By reducing the number of invalid routes, TSR minimizes TCP's consecutive timeouts. In our simulation study, TSR achieves up to 50% TCP performance improvement without requiring any modification of TCP stack in end systems.**

***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 their dynamic properties, mobile ad hoc networks have gained a significant attention lately as a way of providing continuous network connectivity to mobile computing devices in various areas such as disaster control and military applications. In such applications, reliable packet exchange is an indispensable requirement.

However, TCP performance suffers due to the following reasons in mobile ad hoc networks.

- Packet losses resulting from broken routes tend to occur consecutively. TCP Reno has only one way, timeout, to escape from consecutive packet losses, and should wait for the expiration of retransmission timer before TCP resumes packet transmission[6].
- Selecting an invalid alternative route while reestablishing a broken one would result in consecutive timeouts. In this case, TCP RTO (Retransmission Time-Out) value is backed off, and this would degrade link utilization[7].
- Frequent route changes can make TCP segments be out-of-order[5].
- TCP data packets should contend with their ACKs for taking wireless channel (IEEE 802.11 MAC)[15]. Their collision could be the cause of TCP packet losses and low link utilization.

So, there have been many research works[1]-[5] to address the above problems, and they are summarized in two categories:

- A new TCP's reaction to deal with a route failure in which node mobility results.
- A modified TCP mechanism to adapt itself to dynamic ad hoc environment.

The first category includes TCP-F[2] and ELFN[1]. TCP-F is proposed by Chandran et al. In this scheme, when an intermediate node detects a route failure, it explicitly sends an RFN (Route Failure Notification) message to the TCP sender. On receiving an RFN message, the TCP sender suspends packet transmissions and freezes all states, including the RTO value and CWND (the size of Congestion WiNDow). After this, an intermediate node learns of a new route to the destination, and 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. proposed a scheme that uses ELFN (Explicit Link Failure Notification). This scheme is similar to TCP-F. However, there is a difference in that ELFN finds alternative routes voluntarily. ELFN sends a probing packet whenever ELFN timer expires (usually 2 seconds), and on receiving corresponding ACK, it resumes TCP's previous state.

The schemes in the second category are Fixed-RTO[4] and TCP-DOOR[5]. Fixed-RTO is a heuristic scheme that Dyer et al. proposes. By disabling TCP's backoff mechanism, this scheme prevents the retransmission timer from backing off. A problem in Fixed-RTO scheme is that it baselessly assumes that the cause of consecutive timeouts is a route failure. However, Fixed-RTO has a contribution that it shows timer's backoff is important factor, which degrades TCP performance.

Wang et al. proposed a scheme called TCP-DOOR. On receiving out–of-order TCP segments, TCP-DOOR assumes that route changes occur. When the TCP receiver detects out-of-order event, it notify the sender by setting the OOO (Out-Of-Order) bit in the TCP ACK packet. Once the TCP sender detects the OOO bit from the ACK stream, it disables congestion control. Henceforth, if the TCP sender has suffered from congestion symptoms and entered the congestion avoidance state (such as halving its CWND size), it recovers immediately to the state before the congestion avoidance action was invoked. The conclusion of this study was that the detection of the OOO bit implies that a route change event has just happened, and TCP can use it for a better performance.

In this paper, we propose a new mechanism called TSR (TCP-aware Source Routing), which is similar to ELFN but more efficient. TSR is to reduce the chance of selecting invalid alternative routes without requiring TCP stacks in end systems to be modified. By simulation study, we will show that TSR increases the goodput in TCP.

The rest of this paper is organized as follows. Section 2 describes the problem, which has influence on TCP goodput in mobile environment. Section 3 describes our TSR mechanism. Section 4 gives the comparison and analysis of TSR and ELFN, and shows the simulation results. Finally, Section 5 concludes the paper.

II. TCP PROBLEM IN MOBILE AD HOC ENVIRONMENT

In mobile ad hoc networks, the routing protocols try to reestablish broken route as quickly as possible. However, it takes time to reestablish a new route. DSR protocol[8] is a popular routing protocol in ad hoc networks, and it is an on-demand routing protocol that is based on the concept of source routing. DSR is the basis of our TSR.

DSR 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 message. When either the destination node or an intermediate node that contains a valid route to the destination in its route cache, receives the ROUTE REQUEST message, a ROUTE REPLY message 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 message. ROUTE ERROR message is generated at a node when the data link layer encounters a fatal transmission problem. Nodes that receive ROUTE ERROR messages remove these invalid routes from their route cache.

When DSR selects an invalid route, it will cause consecutive timeouts. When a source node is informed of a route failure, it removes the used route from its route cache. If there exist another routes for the destination node in route cache, the source node tries to transmit packets through one of them. In case that the chosen alternative route is again invalid, TCP will suffer a series of consecutive timeouts. Note that the chance of selecting invalid routes is quite high when 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 nodes move faster in mobile ad hoc networks.

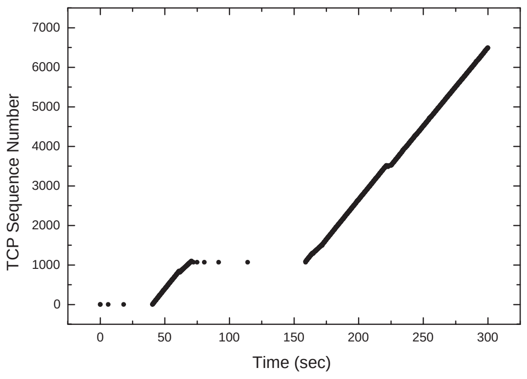

Fig. 1. Impact of consecutive timeout.

If timeouts occur consecutively, the RTO value of TCP's retransmission timer exponentially increases. Since all packet transmissions are suspended until the timer is expired, link utilization is significantly reduced. Figure 1 shows the sequence number of TCP Reno and DSR when consecutive timeouts occur from 0 to 47 seconds and from 70 to 155 seconds. As we can see, packet transmission is suspended for a long period (about 80 seconds). 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 reduces the chance of selecting invalid routes, when the source node is informed of a route failure. In order to achieve this objective, we propose a new mechanism called TSR.

III. TSR MECHANISM

In this section, we describe TSR mechanism in details. TSR is initiated when the source node receives a message that notifies a route failure event. TSR checks the type of transport protocol being used when a route failure occurs. If it is TCP, TSR holds the buffer of routing protocol and starts to find reliable alternative routes. Probing a reliable route should be performed in parallel so as to reduce searching time. Also, only the destination node should respond to each probing packet. Upon receiving a probing packet, the destination node transmits a responding packet through the reverse route. Without requiring a reverse route, the destination node tries to transmit a responding packet through the route in its route cache. However, the chance that this route is also invalid is quite high. TSR in source node escapes from the previous hold state when the source node receives the message that responds to probing packet, and resumes packet transmission through the first responding route. The other responding routes are used for the purpose of updating route cache. 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 routing protocol unnecessary. However, because multiple packet losses are so fatal to TCP performance, we argue that TSR would be effective in most cases, which is proved in our experiments.

In order to implement TSR, we define two additional control messages: ROUTE PROBE and REPLY PROBE. The ROUTE PROBE message is used to probe a reliable new route in the hold state. And REPLY PROBE message is used to reply to the ROUTE PROBE message. The transition diagram of TSR is illustrated in Figure 2.

- TSR activation: TSR is activated when the source node receives ROUTE ERROR, and the state of routing protocol transits to the hold state. Then TSR selects n alternative routes in its route cache and tries to find a reliable and fastest route. Note that intermediate nodes could be activated when they receive ROUTE ERROR. Of course, intermediate nodes assume conditions that they have a TCP session that has used invalid route with ROUTE ERROR message.
- Parallel probing: The source node transmits ROUTE PROBE messages through n alternative routes, and only a destination node could respond to these messages. The chance of invalid routes could be quite high when the source node depends on the routing information from its neighbors that are not updated with the route failure. Actually, it was reported that DSR without the route cache shows better performance in TCP[1]. The destination node that receives ROUTE PROBE messages generates REPLY PROBE messages, and transmits them through the reverse routes.
- Escape from the hold state: On receiving REPLY PROBE messages, the source node escapes from the hold state, and resumes packet transmission. However, if all ROUTE PROBE or REPLY PROBE messages are lost in intermediate nodes, TSR may wait in the hold state infinitely. So, we introduce the TSR timer that is used to escape from the hold state. This TSR timer uses timeout value (TTO) set to

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

where α is a delay variance factor with a recommended value of 1. (Since TCP's RTT moving average adapts itself to mobile ad hoc networks slowly[9], we add α, a delay variance factor, to RTO.)

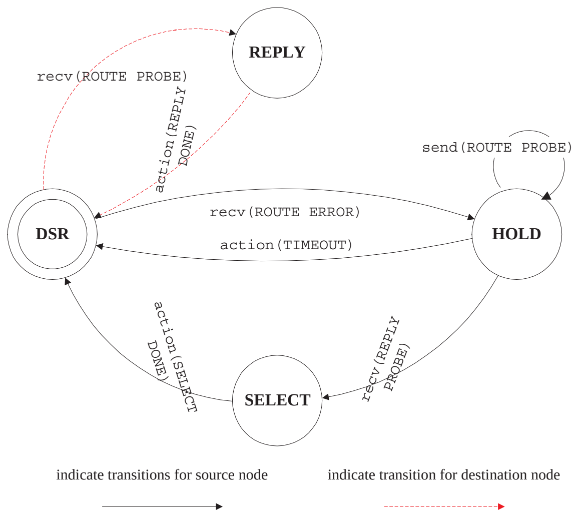

Fig. 2. State transition diagram.

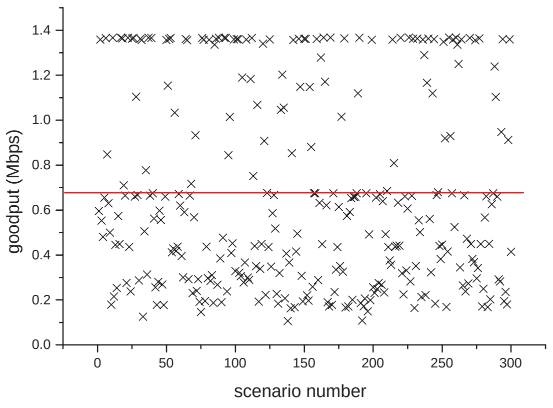

Fig. 3. TCP goodput distribution in 2 m/s.

IV. Simulation results and analysis

The simulation study is done in the NS-2 network simulator[10]. NS-2 is a discrete event simulator that was developed as part of the VINT project at the Lawrence Berkeley National Laboratory. The extensions implemented by the CMU Monarch project[11] enable it to simulate mobile nodes connected by wireless network interfaces. We modified the CMU extension of the DSR code as described in Section 3 to implement TSR.

A. Simulation environments

We generate scenario files by using the CMU random scenario generator, and select TCP sessions randomly in each scenario. Table 1 shows simulation environments. In total 1500 scenario files, goodput results are distributed from 0.01 to 1.36 Mbps. Figure 3 shows the distribution of goodput results when mobile nodes with DSR move at the mean speed of 2 m/s. The red line represents the average value.

Previous works did simulations only with random scenarioes. But we argue that the simple average value of total goodput is not sufficient because the goodputs in Figure 3 have such a wide range. So, we do simulation by two methods.

- Purely random scenario (by CMU scenario generator)
  Almost all previous works did simulation in this way. In order to compare with ELFN, we show the average value of the total goodput.
- Random scenario by limited hop count
  This scenario is similar to the purely random scenario,

Application protocol	FTP
Transport protocol	$TCP\ RENO$
Routing protocol	$DSR,\ TSR$
Link protocol	$IEEE\ 802.11\ MAC$
Link bandwidth	2 $Mbps$
Network model	$1500 \times 1500\ meter\ flat$
Transmission range	$250m$
Node number	50
Scenario number	$300\ by\ each\ case$

TABLE I
SIMULATION ENVIRONMENTS.

but differs from it on the point that this has to keep the hop count of end-to-end nodes.

B. Comparison of TSR and ELFN

The difference between TSR and ELFN is summarized as follows.

1) In receiving ROUTE ERROR message,
   - TSR: TSR transits to the hold state and tries to find a reliable route.
   - ELFN: ELFN stores CWND and RTO values and transits to the standby state. In this state, it periodically sends probing packet.
2) In route probing step,
   - TSR: TSR ignores the route cache of neighbor nodes since it is probing a reliable route.
   - ELFN: If the route cache of the source node contains any alternative route, ELFN selects it and resumes packet transmission. Also, if there is a reply from the route cache of neighbor nodes, ELFN does not check whether new route is invalid or not, and just selects this.
3) In reply step,
   - TSR: TSR transmits response messages through the reverse routes when it receives ROUTE PROBE messages.
   - ELFN: ELFN responds to a probing packet by using an alternative route from its route cache. Therefore, it may send ACKs through invalid route.
4) In overall step,
   - TSR: Intermediate nodes can activate TSR when they receives ROUTE ERROR message. Of course, each node must have had TCP session that used just broken route.
   - ELFN:ELFN is an end-to-end mechanism since it modifies TCP mechanism directly.

C. Simulation results

The main objective of our simulation study is to prove that TSR shows a noticeable improvement in goodput. First, we show the average goodput by varying the mobile node's mean speed as many researchers did.

We define the average goodput as follows:

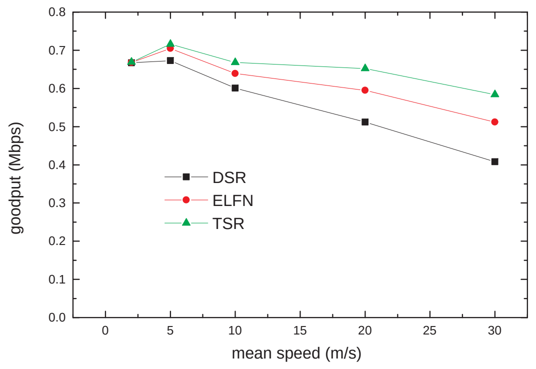

Fig. 4. Average goodput of one TCP session(CMU random scenario).

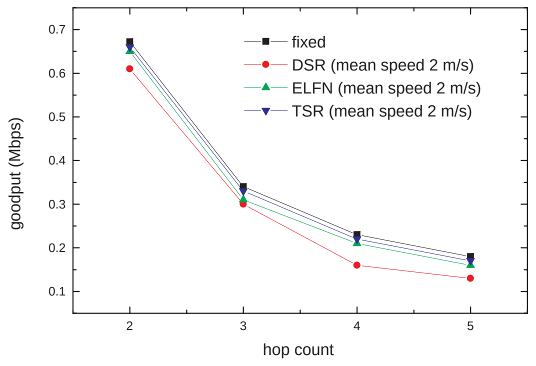

Fig. 5. Average goodput of one TCP session(by hop count, 2m/s).

$$Goodput_{avg} = \frac{\left(\sum_{i=1}^{N} \frac{last\_recv\_seqno_i \times packet\_size}{simulation\_time} \right)}{N} \quad (2)$$

where i is scenario identifier and N is total scenario number.

Figure 4 shows the average goodput of TSR compared with ELFN and normal DSR. X-axis represents the mean speed of mobile nodes and Y-axis represents the average goodput. The mean speed of mobile node is varied by from 2 to 30 m/s. Significant improvements in goodput is observed in the best case (e.g., when the mean speed is 30 m/s). Note that the goodput in the mean speed 5 m/s gets higher than that in 2 m/s. But, overall, the goodput decreases as the mobility increases in Figure 4. We believe that the increase at the mobility of 5 m/s is similar to what[12] found in that the capacity of ad hoc networks may increase with a certain level of mobility.

Second, we measure the average goodput of TCP varying the end-to-end hop count from 2 to 5. We compared DSR and ELFN with TSR for different mean speeds (2 m/s to 30 m/s) in Figure 5 to 9. Again, we found that the graphs in Figure 6(5 m/s) show a higher goodput than those in Figure 5(2 m/s).

Third, we measure the number of invalid routes in Figure 10. X-axis represents the mean speed of mobile nodes and Y-

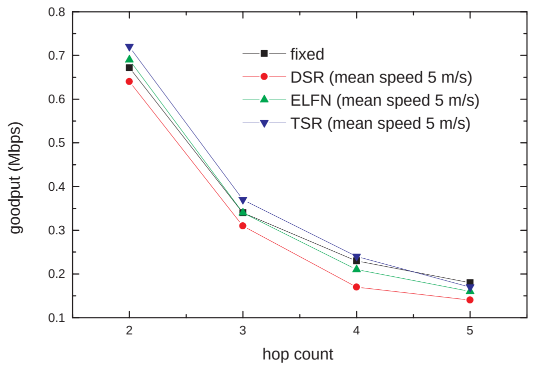

Fig. 6. Average goodput of one TCP session(by hop count, 5m/s).

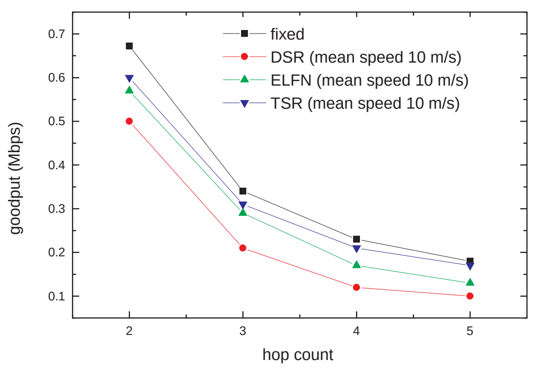

Fig. 7. Average goodput of one TCP session(by hop count, 10m/s).

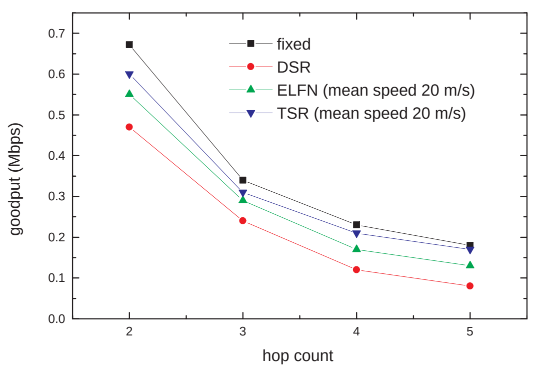

Fig. 8. Average goodput of one TCP session(by hop count, 20m/s).

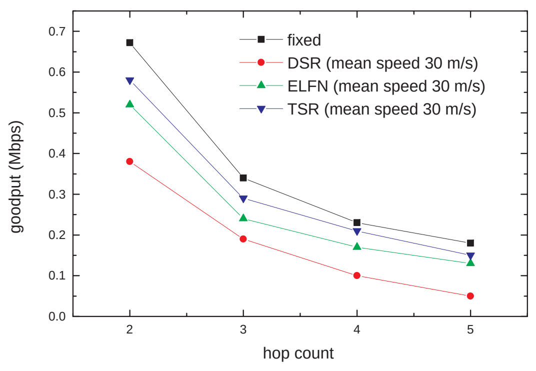

Fig. 9. Average goodput of one TCP session(by hop count, 30m/s).

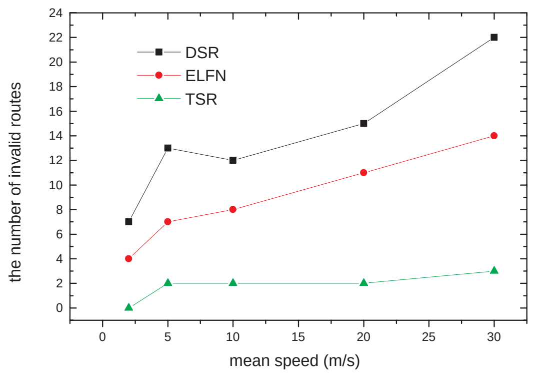

Fig. 10. The number of invalid routes(DSR, ELFN and TSR).

axis represents the average number of invalid routes. Figure 10 says that the average number of invalid routes has a close relationship with TCP goodput. TSR has a lower number of invalid routes than ELFN and DSR, and so it shows a better goodput in almost all cases.

Fourth, we measure the number of timeouts and backoffs with and without TSR in Figure 11. ELFN is excluded in this simulation because it suspends all TCP states by force. In general, as the number of timeouts is reduced, TCP goodput increases. However, in Figure 11, the number of timeouts of TSR at 30 m/s is higher than that of DSR, and in Figure 4, TSR shows better performance then DSR at 30 m/s. It is because the number of backoffs(a result from consecutive timeouts) decreases. From Figure 10 and Figure 11, we confirm that the number of consecutive timeouts have a close relationship with the number of invalid routes.

Finally, Figure 12 demonstrates the effect of the number of TCP sessions. It says that when there are multiple TCP sessions in mobile ad hoc networks, both ELFN and TSR do not achieve any significantoutstanding improvement in performance compared with DSR. There are two reasons. The first one is the MAC contention. All mobile nodes contend to transmit frames, as in IEEE 802.11 networks. The

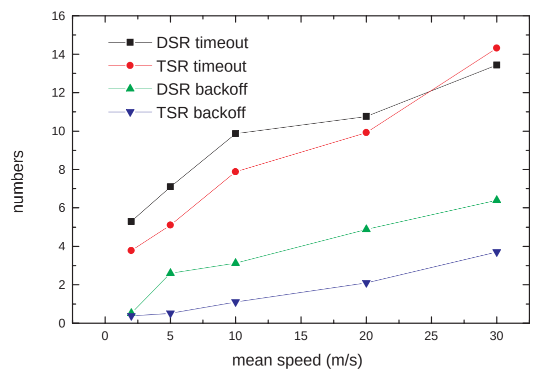

Fig. 11. The number of backoffs and timeouts(DSR and TSR).

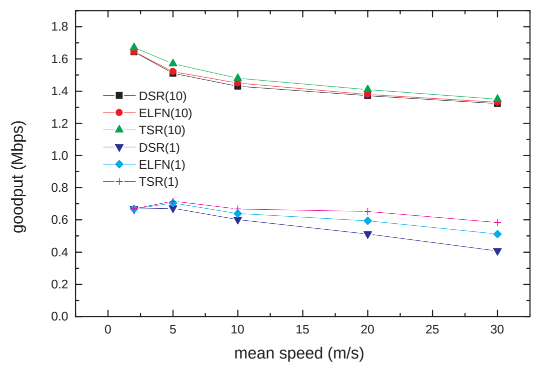

Fig. 12. Average goodput of ten TCP sessions(CMU random scenario).

winner could transmit frames, but others should await the next contention during random intervals. The more sessions are in mobile ad hoc networks, the more data and MAC contention grows. Thus, MAC contention becomes a bottleneck before TCP timeouts when the number of TCP sessions increases. The second reason is that the routing cache tends to be refreshed as 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.

D. Discussion

Even though the number of invalid routes is quite high, ELFN shows good performance in many cases. It is because that ELFN suspends retransmission timer and repeats alternative route probing in every 2 seconds. In other words, ELFN eliminates the bad effect of backoff process by modifying the original TCP mechanism. However, the effect of restoring CWND value has not been verified yet. In this paper, we show there are several cases that only holding timer shows better performance than ELFN. Also, it was reported that larger CWND couldn't always show better goodput[13]. Specifically, some researches connected with TCP Vegas demonstrate that TCP could get the best goodput with the small and steady traffic in IEEE 802.11 ad hoc networks (TCP Reno shows this point when buffer size is 4[14]). We believe that the effect of restoring CWND value has to be examined through the TCP's accurate adaptation for the network capacity.

V. Conclusion

Normal TCP performs poorly in mobile ad hoc networks because of frequent route changes. The cause of this problem is that DSR uses an invalid route as the alternative route. In order to solve this problem, we propose TSR that detects the change of networks and does not send out packets until it discovers a reliable route. As a result, the chance that TSR selects an invalid route and tries to transmit packets could be dramatically reduced than DSR or ELFN. This protocol achieves up to the average 50% improvement compared with DSR, and the average 12% improvement in performance with ELFN. Also, it shows more improvements in goodput as the mobility of mobile nodes increases. It is because that invalid routes are selected more often as mobile nodes move faster. In addition, TSR enhances TCP goodput 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] M. K. Kim and B. Noble, "Mobile network estimation," *in Proc. ACM MOBICOM 2001.*

[10] 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.

[11] CMU Monarch Group, "CMU Monarch extensions to the NS-2 simulator," *Available from http://monarch.cs.cmu.edu/cmu-ns.html,* 1998.

[12] M. Glossglauser and D. Tse, "Mobility increases the capacity of ad-hoc wireless networks," *in Proc. IEEE INFOCOM 2001.*

[13] S. Xu ,T. Saadawi and M. Lee, "Comparison of TCP Reno and Vegas in wireless mobile ad hoc networks," *in Proc. IEEE LCN 2000.*

[14] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?," *IEEE Communications Magazine June 2001.*

[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.