PERFORMANCE EVALUATION OF TCP OVER ROUTING PROTOCOLS FOR MOBILE AD HOC NETWORKS

Md. Abdullah-Al-Mamun, M. Mahbubur Rahman

Department of Information and Communication Engineering
Islamic University
Kushtia 7003, Bangladesh

Hwee-Pink Tan

EURANDOM, P.O. Box 513
Eindhoven 5600 MB
The Netherlands

ABSTRACT

In this paper, we investigate the performance of TCP of various (single-hop and multi-hop) routing protocols for mobile ad hoc networks. Using *ns*-2, we evaluate the TCP window size, throughput and packet delay over a single TCP connection. Our results suggest that the various performance metrics are tightly related, and that TCP performance is tightly coupled with the stability and length of the routing path of each routing protocol.

1. INTRODUCTION

A mobile ad hoc network (MANET) is a collection of wireless nodes that can be set up dynamically anywhere and anytime without using any pre-existing network infrastructure. Nodes within each other's radio range communicate directly via wireless links, while those that are far apart use other nodes as relays in a multi-hop routing fashion [1]. Due to node mobility and variability of link quality, the interconnections between nodes change on a continual and arbitrary basis, giving rise to changing network topology.

An ad hoc routing protocol determines the way to route packets between computing devices in a MANET. Since nodes do not have apriori knowledge of the network topology, they have to discover it. To do so, a new node announces its presence and listens to broadcast announcements from its neighbors. As time goes on, each node becomes aware of all other nodes and one or more ways to reach them by maintaining and updating a routing table. To be effective, routing protocols have to (i) keep the routing table up-to-date and reasonably small, (ii) choose the best route for given destination (e.g., in terms of number of hops, reliability, throughput and cost) and (iii) converge within an exchange of a small amount of messages.

Many routing protocols have been proposed to solve the multihop routing problem in MANETs based on different assumptions and intuitions. However, little is known about the actual performance of these protocols, and very few attempts have been made to compare their performances in a realistic manner. This paper provides a realistic and quantitative comparison of the TCP performance of various multihop routing protocols (DSDV, TORA, DSR and AODV). We describe related work in Section 2 and the mechanism of the routing protocols in Section 3. We define the simulation model and present numerical results in Section 4. Finally, we draw some conclusions and outline possible future work items in Section 5.

2. RELATED WORK

Since TCP/IP is the standard networking protocol stack for the Internet, it is expected to be deployed over MANETs to allow seamless integration with the internet. However, earlier research suggested that TCP performs poorly over cellular (single-hop) wireless networks [2]. This is because the packet loss induced by the wireless link failure is erroneously interpreted as congestion-induced, which triggers an inappropriate response by the TCP mechanism. This is further exemplified in MANETs, since link failures occur more frequently due to node mobility, and give rise to packet loss.

A lot of research has since focused on mechanisms to (a) detect and distinguish link failures from congestion, and (b) initiate the proper response. Examples of such schemes include Explicit Link Failure Notification (*ELFN*) [2] (similar to Explicit Congestion Notification (*ECN*) in wired networks) and *TCP-F* (TCP-Feedback) [3], where the intermediate nodes detect packet loss and the sender can distinguish between route failures and congestion, and initiate the proper response. The difference between *TCP-F* and *ELFN* is the response to route failures: *TCP-F* relies on intermediate nodes to send a route re-establishment notification to notify the sender that the path is restored; In *ELFN*, the TCP sender needs to send probing packets periodically to detect the route recovery.

Another serious problem that link failures may cause to TCP performance is unnecessary exponential backoff of the retransmission timeout (RTO) interval. With conven-

tional TCP protocol, when a retransmission timeout happens, the TCP sender retransmits the lost packet and doubles the RTO. This procedure is repeated until the lost packet is acknowledged. Such an exponential backoff of the RTO helps TCP react to congestion gracefully. However, when link failure happens, TCP tends to increase the RTO rapidly even when there is no congestion. Wrongly applied exponential backoff significantly degrades TCP performance since, in MANETs, the TCP congestion window size is usually small and the RTO plays an important role.

In [4], a mechanism called *fixed-RTO* is proposed, where consecutive occurrence of retransmission timeouts are assumed to be caused by route break, and not by congestion. After retransmitting the lost packet, *fixed-RTO* freezes the RTO value until the route is re-established. Through simulation experiments, it is shown that *fixed-RTO* can achieve TCP throughput comparable to that of the *ELFN* mechanism. However, *ELFN* requires support from the intermediate nodes, while *fixed-RTO* is purely an end-to-end mechanism. In [5], the authors investigated TCP performance over a *multipath* routing protocol. They showed that multipath routing can improve path availability in a mobile environment. Thus, it has a great potential to improve TCP performance in MANETs. A backup path routing scheme is proposed, where TCP can achieve improvements against mobility. Backup path routing actually uses only one path at a time but maintains some backup paths that the routing protocol can switch to rapidly when the current path fails.

As suggested above, most related work focuses on devising mechanisms for TCP to detect route failures or link breakages and react to them accordingly. In this paper, we evaluate TCP performance over various routing protocols in terms of metrics such as TCP window size, throughput and end-to-end packet delay, and show that these metrics are strongly interrelated: when the window size is high (low) then throughput is also high (low) and delay is low (high). For analyzing the TCP window size, we considered combined slow-start with congestion avoidance algorithm [6], which reduces packet dropping and achieves better performance than conventional TCP. Thus, our work complements previous work and can be combined to help TCP achieve good performance.

3. MULTI-HOP ROUTING IN AD HOC NETWORKS

When link failures occur in MANETs, it is important for the routing protocol to detect and restore routing paths to minimize packet loss, since TCP responds to packet loss by reducing window size, which unnecessarily reduces throughput. In this paper, we consider the following routing protocols: DSR, DSDV, AODV and TORA, and their key features are briefly described next.

Destination Source Route (DSR) [7] is composed of two entirely on-demand mechanisms: Route Discovery and Route Maintenance. When a source node wants to send packets to a destination, and a route is not available in its cache, it initiates a Route Discovery by broadcasting a route request. Upon receiving a route request, a node sends a route reply to the sender if a route to the destination is available in its cache; otherwise, it adds its address to the source route in the packet header and re-broadcasts the route request. When the route request eventually reaches the destination, it sends a route reply to the source containing the route from the source to the destination. A connection is thus established and all subsequent packets contain the complete route in the packet header. No routing information is maintained at the intermediate nodes. In Route Maintenance, when the data link layer at a particular node encounters a transmission failure, it issues a route error notification to the source and a new route search is initiated.

Destination Sequence Distance Vector (DSDV) [8] is a distance vector routing protocol. Each node maintains a routing table which contains the next-hop and number of hops to each destination. Each node periodically broadcasts routing updates. A sequence number is used to tag each route, where a higher sequence number indicates a more updated route. Between two routes with the same sequence number, the one with fewer hops is more favorable. If a node detects that route failure to a destination, its hop number is set to infinity and its sequence number is assigned an odd number: even numbers are assigned to connected paths.

Ad-Hoc On-demand Distance Vector routing (AODV) [9] algorithm borrows its salient features from DSR and DSDV. When a source needs a path to the destination, it broadcasts a route request until it reaches a node that has a route to the destination. A route reply generated by the destination propagates along the reverse route and establishes the forward route information at the intermediate nodes. Each node records only the next hop for a destination and not the entire route as in DSR. Failure of a link can be detected via hello messages or link layer detection. When a link fails, the upstream nodes are notified of the failure and the destination is marked as unreachable in the routing tables of these nodes.

In the Temporary Order Routing Algorithm (TORA) [10] when a node needs a route to a particular destination, it broadcast a query packet that propagates till it either finds a node with a route to the destination or the destination itself. That node responds by broadcasting an update packet containing the nodes height. A node receiving the update packet updates its height accordingly and broadcast another update packet. This may result in a number of directed paths from the source to the destination. If a node discovers that a particular destination to be unreachable, it sets the corresponding local height to a maximum value. If the node fails to locate any neighbor with finite height with respect to this

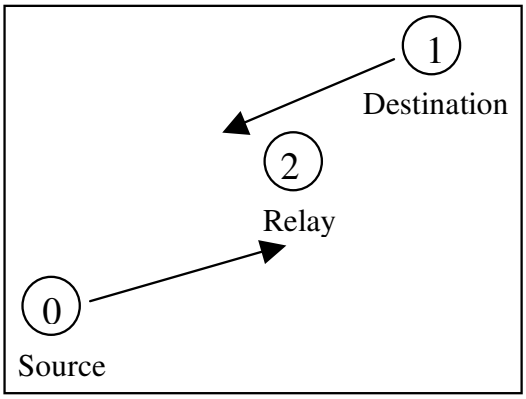

Fig. 1. 500x400m Network Scenario for Topology **A**.

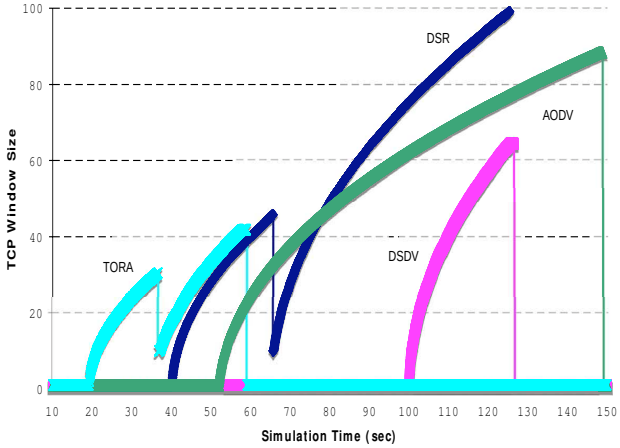

Fig. 2. TCP window size for various routing protocols.

destination, it will attempt to find a new route.

4. SIMULATION MODEL AND RESULTS

For our simulations, we used the *ns*-2 network simulator [11] with the CMU extensions for IEEE 802.11 wireless LAN. We consider two network scenarios: topology **A** which comprises a mobile source and destination nodes with a single stationary relay node and topology **B** with a mobile source node, two stationary relay nodes and a stationary destination node.

4.1. Topology A

The network scenario for topology **A** is shown in Fig. 1. In this network, nodes 0 (source) and 1(destination) move at 3 m/s and 5 m/s respectively towards node 2, which is a stationary relay node. The initial positions, (x,y), of the nodes 0,1 and 2 are (5,5), (425,375) and (275, 200) respectively in the 500X400m topology. For this network, the routing path may be 0-2-1(when nodes 0 and 1 are within radio coverage of node 2, but not of each other) or 0-1(when nodes 0 and 1 are within radio coverage). The effective communication range of each node is 250 meters. We use TCP NewReno and FTP begins transferring packets of size 1000 bytes after 10 second. Each wireless node has a buffer size of 50 packets and its raw radio link capacity is 2 Mbps. Each simulation run lasts 170 seconds.

4.1.1. Performance in Window Size

With the combined slow-start with congestion avoidance TCP algorithm, if W refers to the window size, then W=1 initially, and is increased according to $W = W + \frac{1}{W}$ for each successfully-received packet at the destination. This process is repeated until the connection breaks up, such that for each packet dropped for routing path change, the window size is decreased to half the default threshold value [6]. The evolution of TCP window size with time, t, obtained with each routing protocol is shown in Fig. 2.

According to Fig. 2, we observe that TORA established the TCP connection very quickly and packets are transferred over the interval [19.273, 59.0322]. At t = 37 sec, a change in routing path occurs, resulting in packet drop and causing the TCP congestion window to be reduced to 10 (half the threshold). TORA achieves a maximum window size of 42, which is the lowest amongst all protocols.

DSDV requires the longest connection establishment time, and hence, it is slow, and does not adapt well to an ad hoc network. Although the window size achieved with DSDV is larger than TORA, it is not an actual candidate for ad hoc networks. We observe that DSDV works properly when the mobile nodes are stationary; on the other hand, AODV adapts well to node mobility since it is an on-demand routing protocol. In terms of connection establishment time, its performance lies between that of DSDV and TORA.

According to Fig. 2, we observe that after connection is established with the AODV algorithm, the window size increases monotonously since the same routing path is maintained. On the other hand, DSR dynamically changes its routing path to the shortest one, and therefore works very well in ad hoc networks. As such, it achieves the largest window size and requires a relatively short connection establishment time.

4.1.2. Throughput Performance

We evaluate the throughput obtained with each routing protocol in terms of the total number of packets received at the

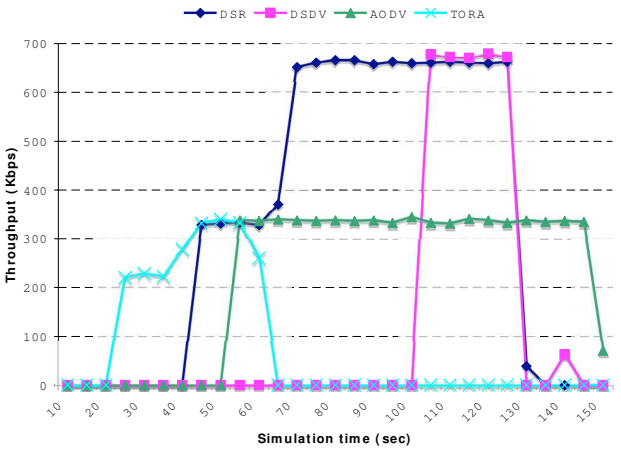

Fig. 3. TCP throughput for various routing protocols.

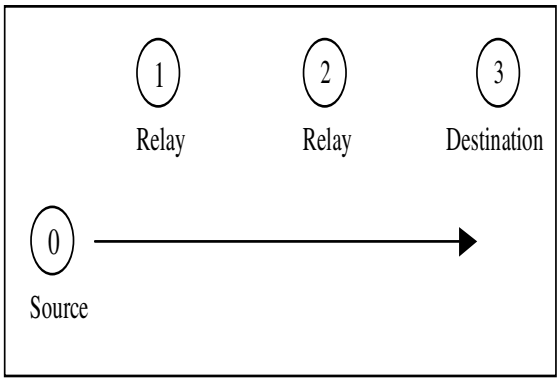

Fig. 4. 1000x800m Network Scenario for Topology **B**.

Speed of node 0 (m/s)	Average Throughput (Kbps)			
	DSR	DSDV	AODV	TORA
0	213.03	160.99	213.27	212.49
5	288.98	342.24	442.15	83.90
10	489.34	379.62	437.91	92.11
15	490.50	253.61	403.39	87.27
20	446.72	344.52	396.16	91.92
25	451.45	271.84	363.70	91.33
30	477.45	139.47	338.71	94.80

Table 1. Throughput vs node mobility obtained with various routing protocols for Topology **B**.

destination node per unit time over intervals of 5 seconds, and the results are plotted in Fig. 3. For each routing protocol, we observe that throughput increases when the number of hops between the source and destination nodes decreases.

According to Fig. 3, TORA achieves a substantial throughput of 220 Kbps at t=20, which is further increased to 330 Kbps for t [40,60] seconds. DSDV maintains a connection only for a very short interval of time [110,130], but achieves a very high throughput of 700 Kbps within this interval (at the maximum window size of 64 packets). AODV achieves a relatively constant throughput (335 Kbps) for t [60,145], which is a longer connected duration than TORA and DSDV. Finally, DSR performs very well for our network topology and maintains throughput at two levels: 330 Kbps for t [40,60] and 660 Kbps for t [70,125] due to the switch from a 2-hop to a single hop path. DSR achieves higher throughput than AODV because the latter does not adapt the routing path sufficiently rapidly.

From Fig. 2 and 3, we observe a strong correlation between TCP window size and throughput, where throughput increases as window size increases and vice versa.

4.2. Topology B

The network scenario for topology **B** is shown in Fig. 4. In this network, node 0 (source) moves towards node 3 (destination) at various speeds (0,5,10,15,20,25,30) m/s while nodes 1 and 2 serve as stationary relay nodes. As the speed increases from 0 to 30 m/s, the routing path changes from 0-1-2-3 to 0-2-3 to 0-3.

4.2.1. Average Throughput Performance

For topology **B**, we average the throughput over the simulation duration and tabulate the average throughput obtained with each routing protocol as a function of node mobility in Table 1.

According to Table 1, when the source node is stationary, all the routing achieves the same throughput of 213 Kbps, except for DSDV. As node mobility increases, the throughput achieved with TORA is constant (90 Kbps) and is lower than that achieved with the other protocols. DSDV performs well at low node mobility, but the throughput performance is significantly degraded at high node mobility. While AODV achieves better throughput than DSDV and TORA, DSR achieves the best throughput performance at all mobility conditions.

4.2.2. Average Delay Performance

For topology **B**, we average the delay over the connection establishment duration and tabulate the average delay obtained with each routing protocol as a function of node mobility in Table 2.

According to Table 2, TORA achieves the highest average delay and lowest throughput. Although DSDV achieves low delay performance, its throughput is lower than that of AODV or DSR since DSDV requires a long time to establish connection. AODV and DSR achieves similar delay perfor-

Speed of node 0	Average Delay (sec)			
(m/s)	DSR	DSDV	AODV	TORA
0	0.38	0.38	0.39	0.38
5	0.27	0.12	0.17	0.37
10	0.14	0.10	0.16	0.34
15	0.18	0.12	0.16	0.37
20	0.17	0.16	0.16	0.33
25	0.18	0.13	0.20	0.33
30	0.15	0.14	0.22	0.38

Table 2. Delay vs node mobility obtained with various routing protocols for Topology **B**.

mance under low node mobility, while DSR achieves lower delay under high node mobility.

4.2.3. Impact of Node Mobility

In this Section, we considered the impact of node mobility on TCP performance as the source node moves *towards* the destination node. We note that routing protocols that adapt sufficiently quickly (e.g. DSR) are able to switch to a routing path with fewer hops, giving rise to an increase (decrease) in throughput (delay) as the nodal speed increases.

In general, node mobility during TCP transmission causes link failures, giving rise to degradation in window size, throughput and delay performance until a new route is formed. One way to circumvent this is to predict the link status, and switch to a new route before link failure occurs. In this case, the choice of the parameter that indicates the link status determines the accuracy of the prediction. Alternatively, while the routing protocols studied here are uni-path protocols, multi-path protocols may be used to enhance TCP performance.

5. CONCLUSIONS AND FUTURE WORK

To date, a variety of routing protocols have been proposed for Mobile Ad-Hoc Networks. However, little attention has been paid to study the performance of TCP traffic over these protocols. We investigate the performance of TCP over multi-hop routing protocols using simulations in *ns*-2 for a range of node mobility with a single traffic source. The performance metrics that we considered include TCP window size, throughput and packet delay. Based on numerical results, we show that TCP performance is tightly coupled with the stability and length of routing paths. We plan to investigate TCP performance of routing protocols with multiple traffic sources in the future.

6. REFERENCES

[1] F. Wang and Y. Zhang, "Improving TCP performance over mobile ad hoc networks with out-of-order detection and response," *Proceedings of ACM Mobihoc*, pp. 217–225, June 2002.

[2] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Proceedings of ACM Mobicom*, pp. 219–230, August 1999.

[3] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *Proceedings of IEEE ICDCS*, pp. 472–479, February 1998.

[4] T. D. Dyer and R. V. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," *Proceedings of ACM Mobihoc*, pp. 56–66, October 2001.

[5] H. Lim, K. Xu, and M. Gerla, "TCP performance over multipath routing in mobile ad hoc networks," *Proceedings of IEEE ICC 2003*, vol. 2, pp. 1064–1068, May 2003.

[6] V. Jacobson, "Congestion avoidance and control," *Proceedings of ACM SIGCOMM*, pp. 314–329, August 1988.

[7] X. Yu, "Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness," *Proceedings of ACM Mobicom*, pp. 231–244, October 2004.

[8] C. Perkins and P. Bhagwat, "Highly dynamic destination - sequenced distance vector routing for mobile computers," *Proceedings of ACM SIGCOMM*, vol. 24, pp. 234–244, 1994.

[9] A. Ahujia, S. Agarwal, J. P. Singh, and R. Shorey, "Performance of TCP over different routing protocols in mobile ad hoc networks," *Proceedings of IEEE VTC*, vol. 3, pp. 2315–2319, May 2000.

[10] V. Park and S. Corson, "Temporally-ordered routing algorithm (tora)," *Internet Draft*, June 2001.

[11] K. Fall and K. Varadhan, "NS notes and documentation," *The VINT Project*, November 1997.