PSR: Proactive Source Routing in Mobile Ad Hoc Networks

Zehua Wang, Cheng Li, and Yuanzhu Chen
Memorial University of Newfoundland
St. John's, NL, Canada
$\{$*zehua.wang, licheng, yzchen*$\}$*@mun.ca*

***Abstract*—Innovative routing in mobile ad hoc networks is crucial for unleashing the full potential of such networks. In this paper, we propose a new Proactive Source Routing (PSR) protocol that has a very small communication overhead but provides nodes with more network structure information than distance-vector based protocols. The value of the source routing protocol includes: 1) better control of path selection by the source nodes for congestion avoidance, load and energy consumption balancing, and bypassing untrusted areas, 2) alleviation of IP forwarding at intermediate nodes, and 3) support for opportunistic data forwarding. PSR complements DSR as a proactive counterpart to provide responsive data transportation services in heavily loaded networks. Our simulation results show that PSR achieves performance similar to OLSR and DSDV, but with only a small fraction of the communication overhead.**

***Index Terms*—Mobile ad hoc network, proactive, source routing**

I. Introduction

A mobile ad hoc network (MANET) is a wireless communication network where nodes that are not within direct transmission range establish their communication via the help of other nodes to forward data. It can operate without a fixed infrastructure, support user mobility, and falls under the general scope of multi-hop wireless networking. Such a networking paradigm originated from the needs in battlefield communications, emergency operations, search and rescue, and disaster relief operations. Later, it found civilian applications such as community networks.

A great deal of research results on MANETs have been published since its early days in the 1980's [1], and the network layer has received the most attention. Two most important operations at the network layer are routing and forwarding. Data forwarding regulates how packets are taken from one link and put on another. Routing determines which path a data packet should follow from the source node to the destination. With different network types, topologies and performance objectives, abundant routing protocols with differing features and for various specific needs have been proposed [2].

Many routing protocols in wireless network are fundamentally derived from two algorithms adopted in the Internet — Link State (LS) routing and Distance Vector (DV) routing. In LS routing, every node provide the cost to it neighbors to all other nodes in the network, so every node has the knowledge of the topology of the entire network and it can always select the best route to the destination. In DV routing, a node provide its neighbors the knowledge of the estimated cost to reach a particular destination, so every node can choose the most efficient neighbor as the next hop to reach a destination node. Because LS provides more information about the network structure than DV does, the LS usually has a larger overhead than DV. Meanwhile, routing protocols in MANETs are usually categorized as *proactive*, a.k.a. *table driven*, and *reactive*, a.k.a. *on demand* according to their timing strategy. Proactive routing means that nodes in the network should maintain valid routes to all destinations at all time. Instead, reactive routing means the nodes in the network do not always maintain routing information. When a node receives data from the upper layer for a given destination, it must first find out about how to reach the destination. In general, most proposed routing protocols in MANETs can be categorized according to their different fundamental algorithms and timing strategies. For example, Destination-Sequenced Distance Vector (DSDV) [3] is a proactive protocol based on DV, Ad-hoc On Demand Distance Vector (AODV) [4] is a reactive one based on DV, and Optimized Link State Routing (OLSR) [5] is a proactive routing protocol using LS.

Another important type of routing protocol is source routing, which is neither LS nor DV, where the entire route is included in the data packet. Therefore, source routing not only provides routing information, but also controls data forwarding when it is handled by intermediate nodes, which is quite different from IP forwarding used by LS or DV. Such a feature in source routing has its exclusive advantages:

- More control by source node — As the entire route can be selected by the source node, it has complete control of how the packet should be forwarded. This allows it to avoid congested areas, balance the load and energy consumption among nodes, and bypass untrusted segments of the network.
- Less requirement on intermediate nodes — Without relying on traditional IP forwarding, source routing does not require intermediate nodes have switching/routing capabilities. The intermediate nodes only need to examine the path embedded in the data packets to forward the data to the next hop neighbor without looking up in its

This work was partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Discovery Grants 293264-07 and 327667-2010).

forwarding table.

- Support of opportunistic data forwarding — The broadcast nature of wireless links differentiates multi-hop wireless networks from the Internet [6]. Opportunistic data forwarding explicitly utilizes such a feature in mesh networks [7]. To achieve this, a redundant set of nodes along the path are allowed to help forwarding data, which are included as an ordered list in the data packets.

In source routing, the source node must include the entire path in data packets for a given destination, DV cannot provide sufficient routing information in this case. On the other hand, LS gives each node the knowledge of the complete topology of the entire network, it could be utilized by source routing as in [7]. However, proactive LS protocols are rather expensive in terms of communication overhead. In this paper, we propose the Proactive Source Routing (PSR) in MANETs to complement the DSR [8] to fulfill the needs of rapid data transportation with a small end-to-end delay. When designing PSR, our primary goal is to minimize the communication overhead without sacrificing the network performance. This is evidenced by our performance evaluation and comparison to OLSR and DSDV using extensive simulation experiments in ns-2.

The rest of this article is organized as follows. Section II reviews related work on source routing and summarizes the difference between PSR and others. Section III describes the design and implementation details of PSR. We use computer simulation to evaluate the performance of PSR and the settings and results of these experiments are presented in Section IV. Section V concludes the article with outlook to future research.

II. Related work

DSR is a pioneering work in source routing in MANETs. In DSR, when a node has data for a destination node, a route request is flooded in the network. An intermediate node who has received the request adds itself to the request and rebroadcasts it. When the destination node receives the request, it transmits a route reply packet back to the source node following the route used by the request. As a reactive protocol, DSR is suitable for delay-insensitive applications and situations where data requests are infrequent.

DSR has since been extended to meet different objectives. Hu and Johnson [9] propose to use *flow ID* to identify a route in the network. The intermediate nodes use the flow ID to look up in a local data structure so that data packets no longer need to carry the full path. To better utilize the cached routes in [10], explicitly signals are used to inform the upstream nodes when detecting a broken link instead of using a timer as in the original DSR. In [11], Bai and Singhal proposed a Way Point Routing (WPR), which combines DSR and AODV together in a hierarchical network. Specifically, they break the end-to-end route into segments with way points, and use DSR for "intersegment" routing and AODV for "intrasegment" routing, respectively. Thus, when a route is broken within a segment, it only needs to find another segment to connect the two way points of the broken segment. Sivakumar and Ramkumar enhanced the security for DSR in [12] by preventing illegal operations on the route information in a data packet. Hu and Gharavi [13] used directional antennas to support DSR for better performance by use of the reduced interference.

Garcia-Luna-Aceves and Spohn proposed STAR [14] as a proactive routing protocol in MANETs. They first put forward two astute observations about route update strategies, *i.e.*, Optimum Routing Approach (ORA) and Least-Overhead Routing Approach (LORA), to interpret the difference between proactive routing and reactive routing from a different perspective. In STAR, every node maintains a tree structure for the network, and adopts a tree update strategy that is neither proactive nor reactive. Instead, it uses a "lazy" approach, where update message, will only be transmitted when the local tree structure is considered sufficiently inferior to the original optimum. Here, all the tree updates are performed differentially and the link states are timestamped.

In this article, we present the working and performance evaluation of a new proactive source routing algorithm called PSR. PSR is built on the similar idea of spanning tree as in STAR. It has the following distinct features. First, we do not timestamp links. Instead, only the topological information is used for tree updates in order to reduce the communication overhead. Second, PSR always maintains a breadth-first spanning tree at each node, to provide responsive data transportation services. Third, we utilize both full dump and differential updates to strike the balance between efficient and robust network operations. Last, we use source routing to forward data rather than IP forwarding as in STAR, which can potentially support opportunistic data transfer in MANETs.

III. Design of PSR

In this section, we highlight some of the challenges of PSR and present the general operations of PSR formulated as graphs. Then we discuss some of the crucial aspects in implementing the protocol. Essentially, PSR builds a Breadth-First Spanning Tree (BFST) in every node of the network. To do that, we must address the following challenges:

- Overhead reduction — As a proactive routing protocol, we must reduce the overhead of PSR. Ideally, we need to provide each node with abundant routing information using a communication overhead similar to or smaller than that of a proactive DV protocol.
- High data transportation performance — Reducing the communication overhead should not penalize the network's capability in data communication.
- Loop prevention — PSR should allow intermediate nodes to modify the paths carried by data packets according to their updated network structure information. However, this needs to be handled properly so as to avoid possible loops.

A. General operations in PSR

Before we describe the operation of PSR, let us first review some graph theoretic terms used here. Let us denote the network as undirected graph $G = (V, E)$, where V is the set of vertices (or nodes) in the network and E is the set of edges (or links). Two nodes u and v are connected by an edge $e = (u, v) \in E$ if they are close to each other and can communicate directly with a given reliability. Given a node v, we use $N(v)$ to denote its open neighborhood, *i.e.*, $\{u \in V | (u, v) \in E\}$. Similarly, we use $N[v]$ to denote its closed neighborhood, *i.e.*, $N(v) \cup \{v\}$. In general, for a node v, we use $N_l(v)$ $(l \geq 1)$ to denote the distance-l open neighborhood of v, *i.e.*, the set of nodes that are exactly l hops away from v. Similarly, $N_l[v]$ $(l \geq 1)$ denotes the distance-l closed neighborhood of v, *i.e.*, the set of nodes that are within l hops of v. As special cases, $N_1[v] = N[v]$, $N_1(v) = N(v)$, $N_0[v]$ is v itself, and $N_0(v) = \emptyset$. Also as a convention in graph theory, for any $S \subseteq V$, we use $\langle S \rangle$ to denote the subgraph induced by S, *i.e.*, $\langle S \rangle = (S, E')$, where E' is the set of edges where each element has both endpoints in S. The readers are suggested to refer to the monograph of West [15] for other graph theoretic notions and other details.

Generally speaking, PSR is a source routing algorithm in that every node has a BFST of the entire network rooted at itself after convergence. To do that, nodes periodically broadcast network structure information to the best of its knowledge. Based on what has been collected from its neighbors in the current iteration, a node can expand the scope of its knowledge about the network structure. This knowledge is exchanged among all neighboring nodes in the next iteration. PSR achieves this with the same communication overhead as proactive distance vector algorithms, such as DSDV, which is significantly smaller than that of the link state routing algorithms like OLSR. The operation of PSR is iterative and

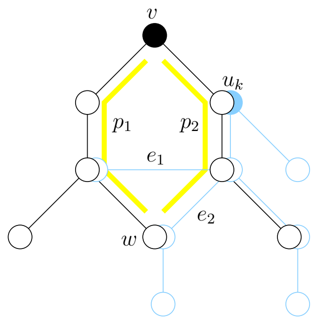

Fig. 1. Add edge

distributed among all nodes in the network. At the beginning, a node v is only aware of the existence of its neighbors by listening to their beacons. Therefore, it is able to construct a BFST rooted at itself within $N_1(v)$, *i.e.*, the star graph centered at v. In each subsequent iteration, nodes exchange their spanning trees with their neighbors. Specifically, in the i^{th} iteration $(i = 2, 3, 4, \ldots)$, node v does the following:

1) It receives a broadcast message from each of its neighbors. The message from neighbor u_k $(k = 1, 2, \ldots, |N(v)|)$ contains the BFST of $N_{i-1}[u_k]$ rooted at u_k. We denote this tree with $T_{u_k}^{i-1}$.
2) Node v calculates its own BFST, denoted T_v^i, based the above spanning trees, *i.e.*, $T_{u_k}^{i-1}$ $(k = 1, 2, \ldots, |N(v)|)$ and T_v^{i-1}. This is essentially done by calculating a BFST of the union graph

$$T_v^{i-1} \cup \bigcup_{k=1}^{|N(v)|} T_{u_k}^{i-1}$$

of all these trees. This is achieved by incrementally incorporating each tree $T_{u_k}^{i-1}$ $(k = 1, 2, \ldots, |N(v)|)$ to T_v^{i-1}. Consider the example of node v and its neighbor u_k in Figure 1, T_v^3 is depicted in black and $T_{u_k}^3$ is in light blue. We use $\mathcal{T}$ to denote a temporary tree which is initially set to T_v^{i-1}. We add each edge of $\bigcup_{k=1}^{|N(v)|} T_{u_k}^{i-1}$ to $\mathcal{T}$ one at a time, denoted by $\mathcal{T}'$. If $\mathcal{T}'$ contains a cycle, we use the following procedure to break it.
   a) If the cycle is odd (*e.g.*, adding edge e_1 in the figure), in which case the newly added edge must connect two vertices of the same distance from v. Remove this new edge.
   b) Otherwise, on the even cycle (*e.g.*, case of adding e_2), we locate the vertex which is the farthest from v, denoted w. Node w has two paths, p_1 and p_2, to the vertex that is on the opposite side of the cycle. In this case, we need to break the tie by removing one of the two edges during the cycle that are incident on w. In PSR, we only use the hop count in the BFST to evaluate the costs on different routes, so to avoid too much unnecessary operations. In that case, we can just keep the original edge in path p_1 and remove the new edge.

   After incorporating all edges of $\bigcup_{k=1}^{|N(v)|} T_{u_k}^{i-1}$, we have constructed T_v^i, *i.e.*, the BFST of $N_i(v)$ rooted at v.
3) At the end of the period, node v broadcasts T_v^i to all of its neighbors.

Assume that the network diameter, *i.e.*, the maximum pairwise distance, is D hops. After D iterations of operation, each node in the network has constructed a BFST of the entire network rooted at itself. This information can be used for any source routing protocol. The amount of information that each node broadcasts in an iteration is bounded by $O(|V|)$ and the algorithm converges in at most D iterations.

B. Implementation

The implementation of PSR has to address a fairly large number of issues. Due to the limited space, we highlight those related to overhead reduction, notification of unreachable nodes, and incorporation of neighbors' trees.

1) Overhead reduction: Two of the primary approaches that we took to reduce communication overhead include 1) using compact tree representation in route exchange messages, and 2) interleaving less frequent "full-dump" messages with more frequent "differential" ones.

- Compact tree representation — Here, our goal is to broadcast the BFST information stored at a node to its neighbors in a short packet. To do that, we first convert the general rooted tree into a binary tree of the same size, say s nodes. Then we serialize the binary tree using a bit sequence of $34s$ bits, where the IPv4 is assumed. Specifically, we scan the binary tree layer by layer. When processing a node, we first include its IP address in the sequence. In addition, we append two more bits to indicate if it has a left and/or right child. For example, the binary tree in Figure 2 is represented as A10B11C11D10E00F00G11H00I00.

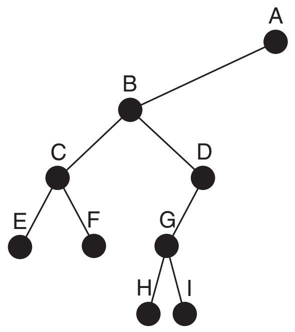

Fig. 2. Binary Tree

- Full dump and differential updates — If the topology of the network is relatively stable, broadcasting full BFSTs frequently would be an overkill. Hence, in PSR, we only allow each node to broadcast the full tree structure in a *full-dump* message once very f cycles. In stead, in every cycle, a node broadcasts a *differential update* message to depict the changes in its locally stored BFST if any. Since the broadcast messages in wireless networks are usually unreliable, there is a tradeoff between overhead reduction and robust dissemination of the trees. In our experiments, we set $f = 3$ to strike this balance.
The difference between two BFSTs over the same set of nodes can be represented by the set of nodes who have changed parents, *i.e.*, which are essentially a set of edges connecting to these new parents. Note that these edges form a forest, we use the above tree representation to package each tree in the forest. After a node receives a differential update message from a neighbor, it extracts the edges and use them to reconstruct the new BFST. This requires each node to cache an old copy of the BFST for each of its neighbors.

2) Notification of unreachable nodes: When a node v is about to transmit its periodical route information, it verifies if it is still connected to all of its neighbors. That is, if it has missed the periodic broadcast message from a neighbor during the previous cycle, the link to this neighbor is considered to be lost. Then, node v will trigger a repair procedure as follows. It first removes all broken links from its BFST, consequently trimming some branches from the tree. Each trimmed branch is rooted at a lost neighbor, rendering the nodes in this branch temporarily unreachable. Node v then incorporates the cached copies of the BFSTs of its neighbors to repair its own tree. Such a repair procedure may or may not give node v BFST to all temporarily unreachable nodes, *i.e.*, some nodes in the network may still be unreachable according to the combined knowledge of $N(v)$. Thus, in the event of differential update, we use a dummy tree rooted at the special IP address of 255.255.255.255 to represent the changes in the "set of unreachable nodes". This being said, full-dump message always include reachable nodes only, which implies that all other nodes are implicitly unreachable. Upon receiving the route message from a neighbor, node v needs to update the cached set of unreachable nodes for the neighbor as well. If these unreachable nodes cannot be reached via other neighbors of v, they become part of v's set of unreachable nodes. The set of unreachable nodes from v, whether detected by v itself or informed by a neighbor, will be used when incorporating a neighbor's tree at a later time.

3) Incorporation of neighbors' trees: The description of PSR's general idea earlier in this section covers the case of growing the spanning trees of a node. After the BFST has been constructed and during the network's subsequent operation, when a node v has received a route update from a neighbor u, it incorporates this new information differently based on the following cases for each destination d.

1) If the neighbor u indicates that node d, which is considered unreachable previously by v, is reachable again, the augmentation of v's spanning tree is similar to growing the tree.
2) If the neighbor u indicates that d has a shorter distance than what node v is aware of, v adopts this neighbor as the next hop to reach this destination node unless there is a loop.
3) If the neighbor u indicates that d has a longer distance instead, and if node v considers u as the next hop to reach d, node v first adopts u's suggestion. Then it query the cached trees for all other neighbors. If any neighbor yields a better route to d, v's BFST is modified, *i.e.*, using that neighbor as next hop for d.

IV. Performance evaluation

We study the performance of PSR by running computer simulation using Network Simulator ns-2 (version 2.34). We compare the results with those from OLSR and DSDV with varying network densities and node mobility rates as they both are widely adopted proactive routing protocols in MANETs and their characteristics are well understood by the research community. We observe that the overhead of PSR is only a fraction of OLSR and DSDV, however, it achieves similar performance in transporting TCP and UDP data flows when compared with the other two protocols.

A. Experiment settings

To model node mobility, we adopt the random waypoint model to generate the simulation scenarios. In this model, each node moves towards a series of target positions. The rate of velocity for each move is uniformly selected from $[0, v_{\max}]$. Once it has reached a target position, it may pause for a specific amount of time before moving towards the next position. In our tests, we have two series of mobile scenarios using this mobility model. The first series of scenarios have a fixed $v_{\max}$, a fixed number of nodes, but with a varying network dimension. The second series have the same network size and dimension, but with a varying $v_{\max}$. A total of 50 nodes are deployed in different network dimensions to achieve varying node densities. The simulation time is 500 seconds for all scenarios. We set the data rate at the link layer to 2 Mbps.

We apply TCP and UDP flows separately to the network scenarios. Specifically, both TCP and UDP segments have a payload of 512 bytes. The UDP flows are generated by CBR sources at the rate of 1 pkt/s, and TCP flows are generated by FTP agents. When comparing these protocols, we record 1) their routing overhead, 2) UDP Packet Delivery Ratio (PDR), and 3) TCP throughput. All of these measurements are compared for varying node densities and mobility velocity rates.

B. Peformance analysis

We measure and compare the total communication overhead (bytes/node/second) of PSR, OLSR, and DSDV versus varying node area density (Figure 3) and varying node rate of velocity (Figure 4). We observe that PSR has only a very small fraction of the overhead of OLSR and DSDV. In addition, its overhead is insensitive to node density and node velocity change. In contrast, both OLSR and DSDV's overhead increase with the node mobility. Note that OLSR's overhead is less when the node density is either fairly low or fairly high because the Multi-Point Relay (MPR) mechanism becomes very effective in dense networks. We then investigate and compare the TCP throughput of these protocols for the two scenarios. We can see that the TCP throughput performance of all three protocols is little affected by the node density which is mostly in the range of 400 to 600 kbps as shown in Figure 5. The similar result is observed for the other scenario where node velocity is varying, but the variance is slightly larger, as shown in Figure 6. Note that, for each simulation scenario, we run PSR, OLSR, and DSDV separately and measure their performance; hence there is a high correlation in TCP throughput among these protocols.

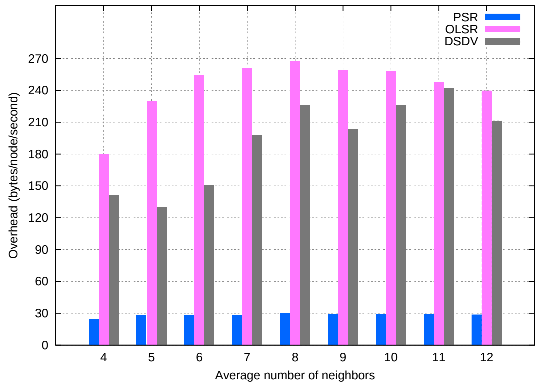

Fig. 3. The overhead vs. network density

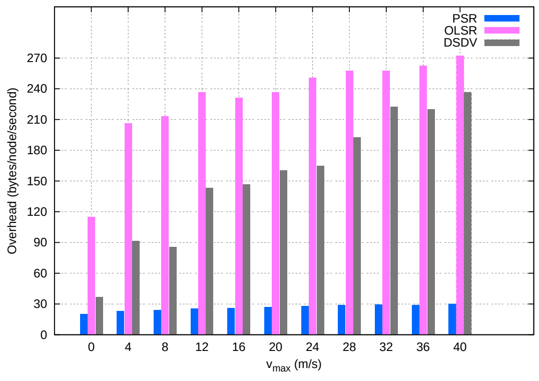

Fig. 4. The overhead vs. node velocity

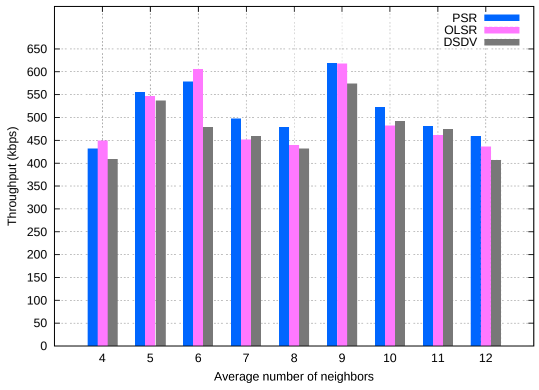

Fig. 5. TCP throughput vs. network density

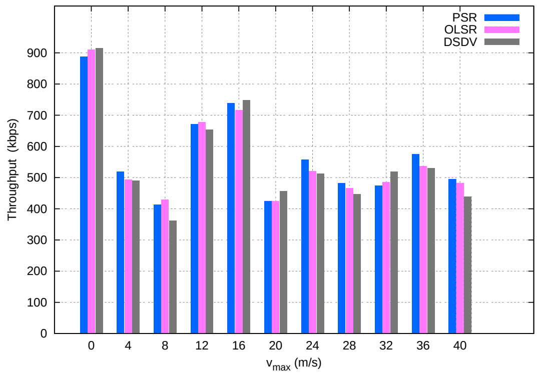

Fig. 6. TCP throughput vs. node velocity

We also study the UDP packet delivery ratio of PSR, OLSR, and DSDV under these two scenarios. Again, their relative performance is very similar. As shown in Figure 7 and 8, the PDR increases with the network density. This is because of two reasons: first, the number of available paths increases with the average number of neighbors of a node; second, the average length of paths decreases with the density. In contrast, the UDP PDR drops gradually as the node velocity increases, rendering all these protocols not as effective in capturing the network topology changes.

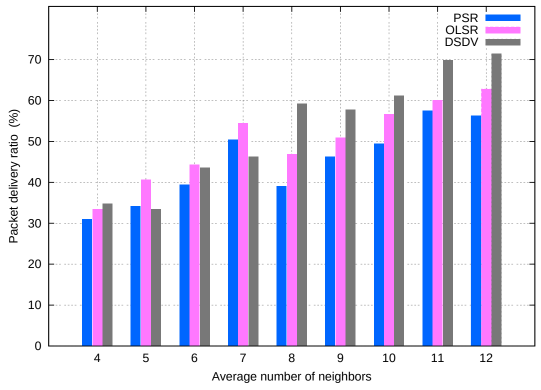

Fig. 7. UDP PDR vs. network density

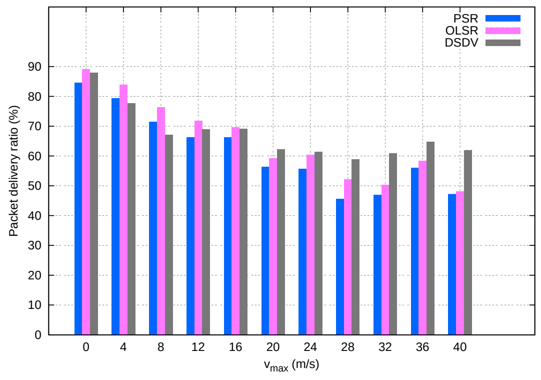

Fig. 8. UDP PDR vs. node velocity

V. Concluding remarks

This work was motivated by the need of providing source routing to support opportunistic data forwarding in MANETs. In order to provide responsive data transfer capability in such networks, a proactive source routing protocol is highly preferred. Despite that the array of optimization techniques employed by OLSR, its overhead remains high in the presence of the constrained communication resources in MANETs. Thus, we set forth to design PSR, which can provide nodes with the cost of network structure information for source routing at a communication overhead similar to or even less than a proactive distance vector routing protocol. In PSR, nodes maintain and exchange BFSTs periodically. The full-dump message containing the entire spanning tree is of the size $O(|N|)$, which is in fact much less frequently broadcast than a compact differential updates. While achieving these objectives, PSR yields the same transportation capability as the more expensive protocols like OLSR and DSDV.

Our future research includes further improving the performance of PSR in the way that a data packet is not dropped immediately after the link layer reports a transmission error. Instead, the intermediate node can choose to recover it by conducting some sort of quick local repair. The goal is that by increasing the persistence in forwarding a packet, both UDP PDR and TCP throughput will be increased.

References

[1] I. Chlamtac, M. Conti, and J.-N. Liu, "Mobile Ad hoc Networking: Imperatives and Challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, July 2003.

[2] R. Rajaraman, "Topology control and routing in ad hoc networks: A survey," *SIGACT News*, vol. 33, pp. 60–73, June 2002.

[3] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Computer Communication Review*, pp. 234–244, October 1994.

[4] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," *RFC 3561*, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[5] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *RFC 3626*, October 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

[6] Y. P. Chen, J. Zhang, and I. Marsic, "Link-Layer-and-Above Diversity in Multi-Hop Wireless Networks," *IEEE Communications Magazine*, vol. 47, no. 2, pp. 118–124, February 2009.

[7] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks," in *Proceedings of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Philadelphia, PA, USA, August 2005, pp. 133–144.

[8] D. B. Johnson, Y.-C. Hu, and D. A. Maltz, "On The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," *RFC 4728*, February 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4728.txt

[9] Y.-C. Hu and D. B. Johnson, "Implicit Source Routes for On-Demand Ad Hoc Network Routing," in *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, Long Beach, CA, USA, October 2001, pp. 1–10.

[10] X. Yu, "Distributed Cache Updating for the Dynamic Source Routing Protocol," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 609–626, June 2006.

[11] R. Bai and M. Singhal, "DOA: DSR over AODV Routing for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 10, pp. 1403–1416, October 2006.

[12] K. A. Sivakumar and M. Ramkumar, "An Efficient Secure Route Discovery Protocol for DSR," in *Proceedings of the 2007 IEEE Conference on Global Telecommunications (GLOBECOM)*, Washington D.C., DC USA, November 2007, pp. 458–463.

[13] B. Hu and H. Gharavi, "DSR-Based Directional Routing Protocol for Ad Hoc Networks," in *Proceedings of the 2007 IEEE Conference on Global Telecommunications (GLOBECOM)*, Washington D.C., DC USA, November 2007, pp. 4936–4940.

[14] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks," in *Proceedings of the 7th Annual International Conference on Network Protocols (ICNP'99)*, Toronto, Canada, October 1999, pp. 273–282.

[15] D. West, *Introduction to Graph Theory (2nd Edition)*. Upper Saddle River, NJ, USA: Prentice Hall, August 2000.