A New Loop-Free Proactive Source Routing Scheme for Opportunistic Data Forwarding in Wireless Networks

Zehua Wang, Yuanzhu Chen, and Cheng Li

***Abstract*—Opportunistic data forwarding (ODF) has drawn much attention in wireless networking research in recent years. The effectiveness of ODF in wireless networks is heavily depended on the choice of proper routing protocols which can provide effective source routing services. In this paper, we propose a new routing protocol named PSR for ODF in mobile ad-hoc networks. PSR is featured by proactive source routing, loop-free, and extremely small routing overhead. Compared to existing routing protocols, there is no need to timestamp routing updates in PSR and the update messages are harmoniously integrated into the tree structure, so that the overhead can be significantly reduced.**

***Index Terms*—Mobile ad hoc networks (MANETs), proactive source routing (PSR), opportunistic data forwarding (ODF).**

I. Introduction

How to better utilize the broadcast nature in wireless communication networks has drawn much attention in the research community in recent years. Opportunistic data forwarding (ODF) represents one of the most promising solutions to this initiative. One of the initial work on ODF is the selective diversity forwarding (SDF) proposed by Larsson [1]. In their work, a transmitter picks the best forwarder from the multiple receivers which successfully received its data, and explicitly request the selected node to forward data. However, its overhead must be significantly reduced before it can be implemented in any practical networks. ExOR provides a solution to this problem and it is the first work which effectively used the concept of ODF [2]. In ExOR, nodes are enabled to overhear all packets on the air, so a multitude of nodes can potentially forward a packet as long as they are included in the *forwarder list* carried by the packet. Furthermore, ExOR fuses the MAC and network layers so that the MAC layer can determine the actual next-hop forwarder. In this way, the forwarder in the list which is *closer* to the destination will access the medium more aggressively so as to better utilize the long-haul transmissions. The idea of ExOR inspired a number of interesting extensions. MORE [3] enhanced ExOR to further increase the spatial reuse via intra-flow network coding. To support ODF in mobile ad-hoc networks (MANETs), Yang *et al.* proposed to use position information for the routing module [4]. Apparently, such a requirement inevitably limits the scope of its use.

On the other hand, although many routing protocols have been proposed for MANETs, *e.g.*, DSDV [5] and OLSR [6], they are not suitable for ODF because either they cannot support source routing or they incur too much overhead. Reactive source routing protocols, *e.g.*, DSR [7] and SASR [8], are not suitable for ODF because longer delay will be experienced and the route reply messages may be lost. WRP [9] and STAR [10] could provide source routing with less overhead. However, they are not optimal choices because they need additional information in routing update to avoid loops which must be prevented in ODF. Moreover, because only differential update is considered in both schemes, the topology may become inaccurate or even unusable over time. Hence, more appropriate routing protocols are required to support ODF in MANETs.

In this paper, we propose a new loop-free proactive routing scheme (PSR) for ODF in MANETs. PSR utilizes the hop count information as a metric to better explore the broadcast nature of the wireless medium, and enhance the efficiency and spatial use in ODF [2]. Network topology information is efficiently maintained and exchanged by using the tree structure, hence the overhead get greatly reduced.

II. Design of PSR

PSR provides every node with a breadth-first spanning tree (BFST) of the entire network rooted at itself. To do that, nodes periodically broadcast the tree structure to its best knowledge. Based on the information collected from neighbors during the most recent iteration, a node can update its knowledge about the network topology by constructing a deeper and more recent BFST. This knowledge will be distributed to its neighbors in the next round of operation. Meanwhile, when a neighbor is deemed lost, a procedure is triggered to remove its relevant information from the topology repository maintained by the detecting node. Intuitively, PSR has the same communication overhead as distance-vector-based protocols. We further reduce the communication overhead incurred by PSR's routing agent. Details about such overhead reduction will be discussed later in this section.

Before describing the details of PSR, we first review some graph theoretic terms used here. Let us model the network with an undirected graph $G = (V, E)$, where V is the set of nodes (or vertices) and E is the set of wireless links (or edges). 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 given reliability. Given a node v, we use $N(v)$ to denote its open neighborhood, *i.e.*, $\{u \in V | (u, v) \in E\}$, and use $N[v]$ to denote its closed neighborhood, *i.e.*, $N(v) \cup \{v\}$.

A. Route update

The update operation of PSR is iterative and distributed among all nodes in the network. Initially, a node v is only

Manuscript received June 6, 2011. The associate editor coordinating the review of this letter and approving it for publication was H.-H. Chen.

The authors are with Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada (e-mail: licheng@mun.ca).

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Discovery Grants 293264-07 and 327667-2010, and Strategic Project Grant STPGP 397491-10).

Digital Object Identifier 10.1109/LCOMM.2011.092011.111200

aware of the existence of itself, so there is only a single node in its BFST, which is root node v. By exchanging the BFST with the neighbors, it is able to construct a BFST within $N[v]$, *i.e.*, the star graph centered at v, denoted by S_v.

In each subsequent iteration, nodes exchange their spanning trees with their neighbors. From the perspective of node v, towards the end of each operation interval, it has received a number of routing messages from its neighbors packaging the BFSTs. Note that, in fact, more nodes may be within the transmission range of v, but their periodic updates were not received by v due to collisions or bad channel conditions. Node v incorporates the most recent information from each neighbor to update its own BFST. It then broadcasts this tree to its neighbors at the end of the period. Formally, v has received the BFSTs from some of its neighbors. Including those from whom v has received updates in recent previous iterations, node v has a BFST, denoted T_u, cached for each neighbor $u \in N(v)$. Node v constructs a union graph

$$G_v = S_v \cup \bigcup_{u \in N(v)} (T_u - v).$$

Here, we use $T - x$ to denote the operation of removing the subtree of T rooted at node x. Then, node v calculates a BFST of G_v, denoted T_v, and places T_v in a routing packet to broadcast to its neighbors.

In our implementation, the above update of the BFST happens multiple times within a single update interval so that a node can quickly incorporate new route information to its knowledge base. To the extreme, T_v is modified every time when a new tree is received from a neighbor. Apparently, this is a trade-off between the routing agent's agility to network changes and computational cost. Here, we choose routing adaptivity as a higher priority assuming that the nodes are becoming increasingly powerful in packet processing. Note that this does not increase the communication overhead at all because one routing message is always sent per update interval.

Assume that the network diameter is D hops. After D iterations, each node in the network has constructed a BFST of the entire network rooted at itself. This information can be used for any generic 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. Neighborhood trimming

The periodically broadcast routing messages in PSR also double as "Hello" messages for a node to identify which other nodes are within its proximity. When a neighbor is deemed lost, its contribution to the network connectivity should be removed, called "neighbor trimming". Consider node v. The neighbor trimming procedure is triggered at v about neighbor u when

- no routing update or data packet has been received from this neighbor for some time, or
- a data transmission to node u has failed as reported by the link layer.

Node v responds as follows.

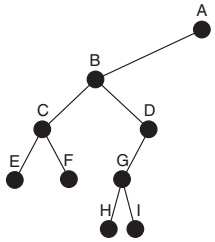

Fig. 1. Binary tree.

1) Update $N(v)$ with $N(v) - \{u\}$.
2) Construct the union graph with the information of u removed:

$$G_v = S_v \cup \bigcup_{w \in N(v)} (T_w - v).$$

3) Compute the BFST T_v.

Notice that T_v thus calculated is not broadcast immediately to avoid excessive messaging. With this updated BFST at v, it is able to avoid sending data packets via lost neighbors.

C. Streamlined differential update

In addition to using route updates as hello messages, we interleave "full dump" routing messages with "differential updates". The basic idea is to send the full update messages less frequently than shorter messages containing the difference between the current and previous states of a node's routing module. Both the benefit of such an approach and how to balance between these two types of messages have been studied extensively in earlier proactive routing protocols. Here, we further streamline the routing update in two new aspects. First, we using a compact tree representation in full-dump and differential update messages to halve the size of these messages. Second, every node attempts to maintain a "stable" BFST as the network changes so that the differential update messages are even shorter.

- Compact tree representation — For the full dump messages, 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 generic rooted tree into a binary tree of the same size, say s nodes. Then we serialize the binary tree using a bit sequence of $34 \times s$ 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 the left and/or right child. For example, the binary tree in Figure 1 is represented as `A10B11C11D10E00F00G11H00I00`. As such, the size of the update message is a bit over half compared to the traditional approach, where the message contains a discrete set of edges.
  The difference between two BFSTs can be represented by the set of nodes who have changed parents, which are essentially a set of edges connecting to their new parents. We observe that these edges are often clustered in groups. That is, many of them form a sizeable tree subgraph of the network. Similar to the case of full dump, rather than

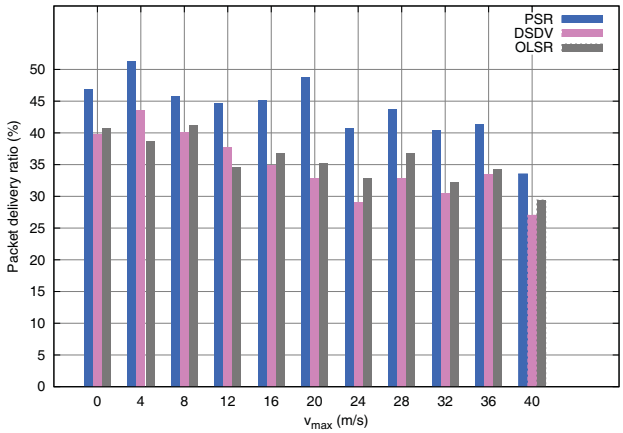

Fig. 2. Packet delivery ratio vs. node velocity.

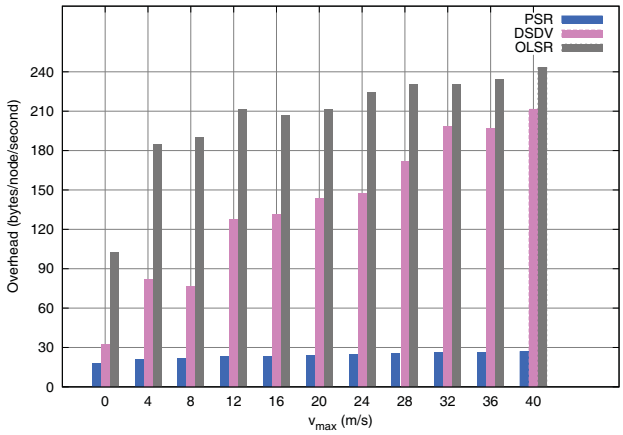

Fig. 3. The overhead vs. node velocity.

using a set of loose edges, we use a tree to package the edges connected to each other. As a result, a differential update message usually contains a few small trees, and its size is noticeably smaller.

- Stable BFST — The size of a differential update is determined by how many edges it includes. Since there can be a large number of BFSTs rooted at a given node of the same graph, we need to alter the BFST maintained by a node as little as possible when there are changes detected. To do that, we modify the computation described earlier in this section such that a small portion of the tree needs to change either when a neighbor is lost or when it reports a new tree.

  Consider node v and its BFST T_v. When it receives an update from neighbor u, denoted T_u, it first removes the subtree of T_v rooted at u. Then it incorporates the edges of T_u for a new BFST. Note that the BFST of $(T_v - u) \cup T_u$ may not contain all necessary edges for v to reach every other node. Therefore, we still need to construct the union graph

$$(T_v - u) \cup \bigcup_{w \in N(v)} (T_w - v)$$

  before calculating its BFST. To minimize the alteration of the tree, we add one edge of $T_w - v$ to $T_v - u$ at a time. During this process, when there is a tie, we always try to add edges that were originally remove from T_v.

When node v thinks that a neighbor u is lost, it deletes the edge (u, v) but still utilizes the network structure information contributed by u earlier. That is, even if it has moved away from v, node u may still be within the range of one of v's neighbors. As such, T_v should be updated to a BFST of

$$(T_v - u) \cup (T_u - v) \cup \bigcup_{w \in N(v)} (T_w - v).$$

Note that, since $N(v)$ no longer contains u, we need to explicitly put it back into the equation. Similarly in this case, edges of $(T_u - v) \cup \bigcup_{w \in N(v)} (T_w - v)$ are added to $T_v - u$ one at a time, with those just removed with u taking priority.

III. Performance Evaluation

We implement ODF in ns-2 (v2.34) and use PSR as the routing protocol under the Nakagami fading model. We conduct a set of simulations, where 50 nodes are tested in a square network with the side length of 1100m by varying maximum node velocity from 0m/s to 40m/s. The results are compared to DSDV and OLSR in terms of packet delivery ratio (PDR) and overhead, as shown in Figures 2 and 3. It is obvious that the proposed scheme possesses higher PDR and much lower overhead.

IV. Concluding Remarks

In this paper we have proposed a new source routing protocol for ODF in MANETs. By using compact tree representation and joint full-dump/differential messages in routing update, the overhead in PSR has been greatly reduce. Moreover, we have proposed an improvement scheme to further reduce the overhead. Performance study shows that PSR clearly outperforms other routing protocols, including DSDV and OLSR.

References

[1] P. Larsson, "Selection diversity forwarding in a multihop packet radio network with fading channel and capture," *ACM Mobile Computing and Commun. Rev.*, vol. 5, no. 4, pp. 47–54, Oct. 2001.

[2] S. Biswas and R. Morris, "ExOR: opportunistic multi-hop routing for wireless networks," in *Proc. ACM SIGCOMM*, Aug. 2005, pp. 133–144.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. ACM SIGCOMM*, Aug. 2007, pp. 169–180.

[4] S. Yang, F. Zhong, C. K. Yeo, B. S. Lee, and J. Boleng, "Position based opportunistic routing for robust data delivery in MANETs," in *Proc. IEEE GLOBECOM*, Dec. 2009, pp. 1325–1330.

[5] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *Computer Commun. Rev.*, pp. 234–244, Oct. 1994.

[6] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, Oct. 2003.

[7] 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, Feb. 2007. Available: http://www.ietf.org/rfc/rfc4728.txt

[8] E. Papapetrou and F.-N. Pavlidou, "A novel approach to source routing for multi-hop ad hoc networks," *IEEE Commun. Lett.*, vol. 7, no. 10, pp. 472–474, Oct. 2003.

[9] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *Mobile Networks and Applications*, vol. 1, no. 2, pp. 183–197, 1996.

[10] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," in *Proc. ACM SIGCOMM*, Oct. 1999, pp. 273–282.