A Sowing Routing Protocol for Dense Mobile Ad Hoc Networks

Julien Cartigny[1], Xavier Défago[1,2]

[1]*School of Information Science, Japan Advanced Institute of Science and Technology (JAIST)*
[2]*PRESTO, Japan Science and Technology Agency (JST)*

July 19, 2005

IS-RR-2005-009

Japan Advanced Institute of Science and Technology (JAIST)

School of Information Science
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

http://www.jaist.ac.jp/

ISSN 0918-7553

A Sowing Routing Protocol for Dense Mobile Ad-Hoc Networks

Julien Cartigny* and Xavier Défago*†
*School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-1292, Japan
†PRESTO, Japan Science and Technology Agency (JST)
Email: {cartigny,defago}@jaist.ac.jp

***Abstract*— To reduce the number of control messages in dense ad-hoc networks, some protocols reduce the redundancy (or overlap) between communication radii to limit the routing overhead. In this paper, we propose a novel method to decrease the routing overhead by reducing redundancy of data between neighbors: a node selectively sends routing information that was *not* recently sent by neighbors, and transmits it using the payload of HELLO messages. In addition, the node introduces a bias favoring information pertaining to nearby nodes, as information about farther nodes need to be updated less often. When a route is needed, the route request message is relayed only by nodes that are closer to the destination, thus forming a broadcast limited to a subset of the network. The protocol, called SiRuP (Sowing Routing Protocol) provides a low-overhead routing for dense networks and offers good resistance against route request failures, when nodes are highly mobile.**

I. Introduction

Ad-hoc networks consist of mobile hosts with a wireless radio interface, forming a decentralized network. Because of the radio nature of the communication, a node can directly communicate with its neighbors (a message is received simultaneously by all neighbors within the radio transmission radius). But when a node wants to reach farther nodes, the message must be forwarded by other nodes to the destination. Hence, the network is only supported by the users.

Dense ad-hoc networks are defined as ad-hoc networks with a high number of nodes in a bounded space. Each node has numerous neighbors and competes with them to send a message. To reduce the number of collisions (nodes receiving two messages at the same time cannot understand either) and the routing overhead, messages have to be short and the interval between two packets sent by the same node must be long enough.

Dense ad-hoc networks are a challenging environment for ad-hoc routing protocols. For instance, the proactive routing protocol Destination-Sequence Distance Vector (DSDV) [1] has a high and constant overhead in term of control messages, as each node periodically exchanges its routing table with its neighbors. Ad hoc On-demand Distance Vector (AODV) [2] has a high packet overhead, as route request messages are forwarded by every node in the network. Optimized Link State Routing (OLSR) [3] reduces the control overhead by reducing the number of relays used when a message is broadcasted in the network. A heuristic algorithm selects the minimal set of one-hop neighbors. This set covers every two-hop neighbors, and thus topology control messages are broadcast in the network at a lower cost than with classical blind flooding. However, OLSR requires that every HELLO message includes the list of neighbors, which can become large in a network with high density.

In this paper, we propose a new routing protocol, called Sowing Routing Protocol (SiRuP) that reduces the number of collisions and routing overhead. The main idea of the protocol is to diffused identifiant in control messages with a lower overhead cost. This routing information is not reliable to find an optimal unicast route as other routing protocol. To find the optimal, a broadcast is started every time a route is needed, but this broadcast is limited to a part of the network, because nodes uses the information disseminated by control messages to limit the expanison of the broadcast.

SiRuP is a hybrid routing protocol composed of two parts. First, a table-driven proactive algorithm called Proactive Network Discovery protocol (PND) disseminates identifiers with low redundancy: each node tries to add only information not recently sent by neighbors in the HELLO messages. Then, a reactive algorithm called Reactive Route Request protocol (RRR) uses the information disseminated by PND to limit the size of the route request broadcast. Sowing Routing Protocol is efficient in dense networks, using the reduction of data redundancy in HELLO message and a size-limited broadcast to reduce the cost of a route request. Furthermore, because of the reactive route request broadcast, it is very efficient in case of high mobility in the network. The main contributions of the paper is a new method to reduce routing overhead by minimizing data redundancy between neighbors and limiting the size of a route request broadcast.

Periodically, PND selects a subset of known node identifiers and broadcasts it to its neighbors in HELLO messages. This subset contains the identifiers of the nodes with the oldest timestamp (*i.e.*, identifiers not heard recently from neighbors nodes). Consequently, the routing information is disseminated with a low overhead, as nodes are able to listen to data sent by other nodes and thus avoiding to waste bandwidth by broadcasting the same information. Furthemore, the number

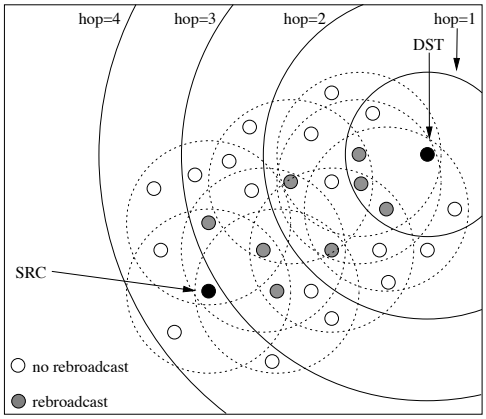

Fig. 1. Example of nodes forwarding the request message from the source (src) to the destination (dst)

of identifiers in the subset is based on a policy indicating the number of node identifiers located within a given number of hop. Nodes are selected within this region (for instance, x identifiers of nodes located at one hop, y identifiers of nodes located at two hops, etc...). We propose in this paper different policies with a goal inspired from the fisheye protocol [4]: select more identifiers of closer nodes than farther nodes, because closes nodes need to be updated more often than farther nodes.

PND minimizes data redundancy between neighbors, but is not as accurate as other protocols for several reasons. First, as SiRuP does not use a topology approach, it does not guarantee that routing information is correctly diffused within the network. Second, as it uses unicast route requests, the length of the discovered route is likely not optimal. Finally, the diffusion of identifiers is slow (compared to protocols like OLSR), as only HELLO messages are used to diffuse control information. Note, howevers that PND has not thegoal to diffuse accurate routing information. The identifiers sent by each node are "sown" in the network to help the reactive route request protocol to find the route and limit the broadcast size.

RRR is the route search part of SiRuP. Each time a node needs a route, it sends a route request message to its neighbors, including the last information it knows about the destination through PND. When a node receives a route request message, it rebroadcasts it only if the distance to the destination is shorter than the distance in the route request packet. Hence, the broadcast is limited to a geographical zone between the source and the destination, and the route request messages become more and more accurate as they get closer to the destination (because the information about destination node is more often updated). Figure. 1 shows an example of route research. The large circles (shown as hop=1, hop=2...) represent the distance in hops from the node dst diffused by HELLO messages. The reactive route research can be seen as a multicast covering nodes between the source and destination, where the route request "harvests" information diffused by PND to direct the research to the destination. Hence, several routes can be found, and the optimal one is used as path for the unicast route reply message.

The paper is organized as follows. Section II presents the OLSR protocol, that we use in our performance comparaisons. Section III is an overview of our Sowing Routing Protocol. Simulation results are presented in Section IV. Section V presents a review of several existing protocols. Finally, Section VI concludes the paper and presents possible improvements of the model.

II. Background: The OLSR Protocol

To show the advantages of our algorithm we compare it to Optimized Link State Routing protocol (OLSR) [3], [5]. OLSR is a proactive link state routing protocol designed to minimize flooding by using only selected nodes. To achieve such broadcast reduction, OLSR has introduced a method called MPR (MultiPoint Relays) [6] to minimize flooding of control packets (called Topology Control packets, or TC) by reducing duplicate retransmissions in the same region.

To accomplish this, a set of neighbors (called multipoint relays or MPRs) is selected by each node in the network to act as relays. Each node periodically broadcasts a HELLO message that includes the list of its neighbors. This way, a node knows the neighboring topology (*i.e.* the identifiers of each one-hop and two-hop neighborhood, the links between one-hop neighborhood, and the links between one-hop neighbors and two-hop neighbors. Hence it can compute, using a heuristic algorithm, the minimal set of one-hop neighbors (the MPR set) which cover all two hop neighbors. Afterwards, multipoint relays of a given node are declared in the subsequent HELLO messages, and each neighbor is able to discover if it is part of the MPR. Topology Control (TC) messages are broadcasted periodically (but less frequently than HELLO message) using MPR (*i.e.* when a node sends the TC message, only its MPR neighbors rebroadcast it). From the TC message data (TC messages contain the identifier of the original sender and the neighbor identifiers which have selected its as MPR) each node is able to compute a route to each node in the network.

OLSR is efficient in reducing the routing overhead, as the percentage of relays for broadcasting of TC messages is decreasing when the density grows up (compared to blind flooding, where every node rebroadcasts the message). But OLSR suffers from several drawbacks. First, OLSR is a uniform protocol: it diffuses the same information periodically to every node in the network using TC messages. But, as presented by the protocol Fisheye (described in Section. V), information about distant node can be less frequent than information about close nodes. Second, a node needs to know information about its one and two-hop neighbors. To be able do gather this information, OLSR periodically sends the list of its one-hop neighbors in each HELLO message. But, if d is the density of the neighborhood, then each node periodically receives $\Theta(d^2)$ identifiers. Hence, the number of identifiers is thus rather high for a dense network, as the probability of collisions at the MAC layer level increases when the density grows. Finally, two neighbors need at least two exchanges to construct their MPR set and, in case of high mobility, the

MPR set of each node can become inconsistent because of the numerous link changes in the neighborhood.

III. Sowing Routing Protocol

The Sowing Routing Protocol (or SiRuP) is a hybrid routing protocol designed for dense networks. It is composed of two parts where one is proactive and the other one is reactive. First, the Proactive Network Discovery protocol (PND) is a proactive identifier dissemination algorithm. A node periodically sends HELLO messages to its neighbors to sown identifiers in the network. The payload of these HELLO messages is a list of identifiers selected according to a given policy and the reception timestamp of known node identifiers. The second part, the Reactive Route Request protocol (RRR), is a reactive route research algorithm. The starting node sends a route request packet to its neighbors. A node forwards the request message only if that node is closer to the destination than the node from which it has received the request (distance are estimated from information disseminated by PND). When the destination receives the route request message, it sends back an unicast route reply message to the source to establish the route between the two nodes.

A. Proactive Network Discovery (PND)

Each node $Node_i$ maintains a table called $sowing_i$. Each table entry contains the identifier of some other node, together with its distance in a number of hops. More precisely, an entry $sowing_i[j]$ contains the following information about $Node_j$:

- *id:* the identifier of $Node_j$;
- *sn:* the last sequence number of $Node_j$ seen in HELLO messages;
- *hop:* the estimated number of hops between $Node_i$ and $Node_j$;
- *stamp:* the timestamp of the creation or last update of the entry $sowing_i[j]$ (*i.e.*, the time when the last message was received with information about $Node_j$). The timestamp is based on the local clock of each node. Hence, no time synchronization is needed between nodes in the network.

For the sake of the explanation, let us decompose the table $sowing_i$ in several subsets $sowing_i^k$, consisting of the entries with a hop distance of k (*i.e.* $\forall entry \in sowing_i^k : entry.hop = k$). Let also define $sowing_i^{\geq k}$ which consists of the entries with a hop distance of at least k (*i.e.* $entry \in sowing_i^{\geq k} : entry.hop \geq k$).

$Node_i$ periodically sends a HELLO message with interval $hello\_interval$. The payload of the HELLO message contains information about $Node_i$ and $selected$, a subset of $sowing_i$.

The process of selecting the entries to be added in $selected$ is based on the following:

- a policy of entry selection in function of the distance;
- the timestamp of the entries in $sowing_i$.

We next describe how the policy is specified. The policy customizes the number of entries added in $selected$ in function of the hop distance, in order of giving more frequent information about close nodes. It reflects the *distance effect* (described in Section. V): a node needs more frequent information about close nodes than distant nodes.

The policy defines the number of entries to be added in $selected$ among entries with each possible hop distance, *i.e.*, it specifies the number of entries about nodes at one hop, the number of entries about nodes at two hops,... The policy has a parameter $size\_policy$: entries about nodes at at least $size\_policy$ are grouped together, and the policy specifies the number of entries to be added from this group. Hence, we consider every entry where the hop-distance is superior or egal to $size\_policy$ as one scope, due to the distance effect. Thus the policy is defined by $size\_policy$ and $size\_policy$ integers:

$$policy = \{scope(1), scope(2), \ldots, scope(\geq size\_policy)\}$$

$Node_i$ chooses the $sowing_i$ oldest entries. Because $Node_i$ listens to neighborhood communications (and updates the local timestamp of $sowing_i$ entries accordingly), it will select a set of entries different from those that have been recently sent by neighbors. Hence, nodes exchange different entry sets, thus reducing the overhead due to control message (when compared with other solutions where neighbors may send identical control messages, even when they are near each other).

Hence, following the policy, we can define the sets $selected^k$ contain the oldest entries (*i.e.*, those with oldest timestamp) of $sowing_i^k$ with $|selected^k| = max(scope(k), |sowing_i^k|)$. Likewise, the set $selected^{\geq k}$ contains the oldest entries (*i.e.*, those with oldest timestamp) of $sowing_i^{\geq k}$ with $|selected^{\geq k}| = max(scope(\geq k), |sowing_i^{\geq k}|)$. They represents the subset of $sowing_i$ to be included in the HELLO message.

The payload of HELLO messages also contains information about the sender $Node_i$. The field $info_i$ contains the same information as a $sowing$ entry (except for the $stamp$ field, as it is based on the local clock) with $info_i.id$ containing the sender identifier and $info.hop$ is set to zero. For $info_i.sn$, it contains the current sequence number of $Node_i$ called sn_hello_i. The role of the sequence number is to help other nodes find the most recent information about $Node_i$. It is initialized to zero at startup and incremented for each HELLO message sent.

Then, the payload of the HELLO message sent by $Node_i$ contains the union of all theses sets and $info_i$:

$$\left(\cup_{k=1}^{size\_policy-1} selected^k\right) \cup selected^{\geq size\_policy}) \cup info_i$$

When a $Node_i$ receives a HELLO message, PND checks every entry in the payload of the HELLO message, updates $sowing_i$ if necessary and discards the message. Because each node wants to keep fresh information and shortest distance, an update in $sowing_i$ is applied if only the sequence number is more recent or if the distance to the destination is shorter than the equivalent entry stored in the table $sowing_i$. Hence, for each entry $elem$ in the payload of a HELLO message:

- If $\forall x : sowing_i[x].id \neq elem.id$, a new entry $sowing_i[j]$ is created with a copy of the field $elem$.

Then $sowing_i[j].hop$ is incremented (as the distance has increase by one hop) and $sowing_i[j].stamp$ is set to the current time of the internal clock.

- If $\exists j : sowing_i[j].id = elem.id$, then the entry is updated if:
  - the sequence number of the HELLO message entry is more recent than the one in $sowing_i[j]$ (*i.e.* $sowing_i[j].sn < elem.sn$);
  - or if the sequence number of the HELLO message entry is equal to the one in $sowing_i[j]$ (*i.e.* $sowing_i[j].sn = elem.sn$) AND the number of hops in the HELLO message entry is lower than the number of hops in $sowing_i[j]$ (*i.e.* $sowing_i[j].hop > elem.hop$).

The update operation consists of copying the sequence number and the hop distance from the field to the entry: $sowing_i[j].sn$ is set to $elem.sn$ and $sowing_i[j].hop$ is set to $elem.hop + 1$ (as the distance has increase by one hop). $sowing_i[j].stamp$ is updated to the actual time of the internal clock. Hence, using sequence number, nodes can find the most recent information with the minimal distance to the destination.

B. Reactive Route Request (RRR)

A route request message is sent when a node needs a route. The message is only forwarded by nodes that are closer to the destination, according to the information disseminated by PND. Hence, the propagation of route request message is limited to a "corridor" with a length equal to the distance between source and destination, and a width approximately equals to the communication radius. This size-limited broadcast can be seen as a implicit multi-path, offering several route to the destination without incurring a high overhead for explictly managing multipath structures.

More precisely, when a source $Node_{src}$ needs a route to a destination $Node_{dst}$, it sends a route request message (called $request$) to its neighboring. The request message includes a copy of the entry $sowing_{src}[dst]$ in the field $request.field$ ($sowing_{src}[dst].stamp$ is not included in the message). The message includes a last-hop node field $request.last$ with the identifier of the last node which has rebroadcasted the route request message (to be able to send back the unicast route reply message). The message also contains the distance from the source with the field (called $request.distance$) and a sequence number (called $request.sn$), different from sn_hello_i. This sequence number is here to help other nodes to discern different route requests issued by the same source. So, each $Node_i$ keeps a $sn\_request_i$ counter set to zero and incremented for every route request generated by the node.

When a $Node_i$ receives the request message, it rebroadcasts it if an entry with the same identifier exists in its $sowing$ table ($\exists dst : sowing_i[dst].id = request.field.id$) and if the number of hops of the entry is inferior to the number of the message ($sowing_i[dst].hop < request.hop$). The message is rebroadcasted as it, except the hop distance is updated to the hop distance to the entry from $sowing$ ($request.field.hop = sowing_i[dst].hop$) and the distance from the source is incremented ($request.distance = request.distance + 1$).

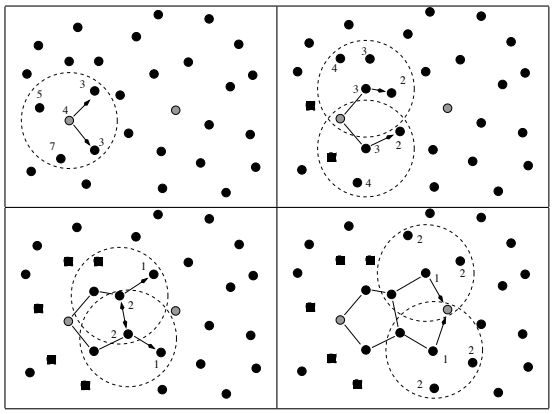

Fig. 2. Example of route request

When a destination $Node_{dst}$ receives a route request message, it waits for a predeterminated amount of time to receive all other messages for the same route request. Then $Node_{dst}$ chooses the best route to $Node_{src}$ (*i.e.*, the route request message with the shortest hop distance) and sends a unicast route reply to the last-hop relay node with the shortest route to $Node_{src}$. Then each node forwards the message until it arrives to $Node_{src}$ and therefore, $Node_{src}$ has now a route to $Node_{dst}$. To be able to forward the route reply message, each node maintains a table to keep track of each route in construction or recently established, associated with a timeout (to remove old routes) and the last-hop node identifier of each route request (to be able to send back the message). Furthermore, nodes continue to listen to other messages of the same route request broadcast. Therefore, if they receives a route request message with the same sequence number and a better hop distance $request.distance$ than any previous route request message, they update the last-hop identifier to help finding the shortest route (but does not rebroadcast the route request message).

The Figure. 2 presents how a route request is forwarded from source to destination. The numbers in the figures indicate the hop-distance to the destination acquired using PND. Each node receiving the route request message forwards it if the distance to the destination is shorter than the previous relay. Several paths are found by the protocol, and the destination can select the best way to reach the source node. Furthermore, as seen in Fig. 1, not every node in the zone forwards the route request message. This is because nodes rebroadcast the message if and only if the distance to $Node_{dst}$ is inferior to the previous relay. Thus a node $Node_j$ receiving a route request message carrying the same hop distance as in its $sowing_j[dst]$ does not rebroadcast the message. This behavior helps reducing the breadth of the route request broadcast.

IV. Experimental Results

In our simulations, we compare our protocol (SiRuP) with OLSR (see Section. II). We used the discrete event simulator

TABLE I

PARAMETERS FOR SOWING ROUTING PROTOCOL AND OLSR

SiRuP	HELLO interval	2s
	sowing Information Hold Time	5 * HELLO interval
OLSR	HELLO interval	2s
	TC interval	5s
	neighbor Hold Time	3 * HELLO Interval
	Topology Information Hold Time	3 * TC interval

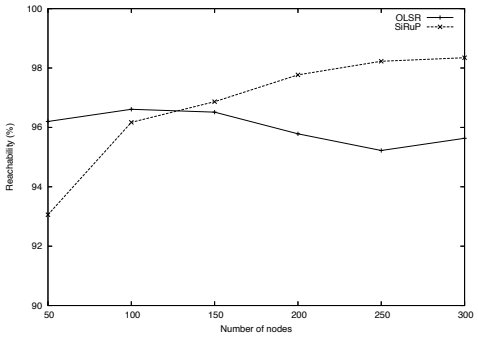

Fig. 3. Reachability as a function of the number of nodes

Omnet++ [7] together with the mobility framework [8], using a perfect MAC layer (no collisions occur). The common parameters for all scenarios are the following: the size of the overall area is 1000m x 1000m with 200 nodes and 1 route request from each node during the simulation that lasted for 90s in simulation time, the transmission radius is 210m and the policy A is used for SiRuP (see Table. II). Each scenario is repeated 30 times. Nodes use the random waypoint model: a node randomly selects a destination in the area and goes to it at constant speed (1.5m/s); Such steps are repeated until the end of the simulation. Table. I summarizes the parameters used for the two protocols (the OLSR parameters are the same as the ones used in RFC 3626 [5]). Each node sends route request messages from 10s until the end of the simulation, at randomly chosen times. We evaluate the protocol using 4 scenarios described bellow.

A. First Scenario: Number of Nodes

In the first scenario, we observe the performance (in terms of reachability, packet overhead, and data overhead) of both protocols while varying the number of nodes from 50 to 300 (this is equivalent to a density from 6.93 to 41.56 nodes per communication radius). Figure. 3 shows the reachability, that is the percentage of route constructions that succeed. OLSR presents a stable result, whatever is the density, but SiRuP increases its performance as the number of nodes grows, outperforming OLSR when the number of nodes grows beyond 150. The advantage of SiRuP when the network is dense is due to two reasons. First, the policy is fixed in every configuration, hence the amount of control message exchanges between neighbors grows as the density increases, and thus each node receives more frequent updates about the topology. Second, due to the size-limited broadcast, the route research

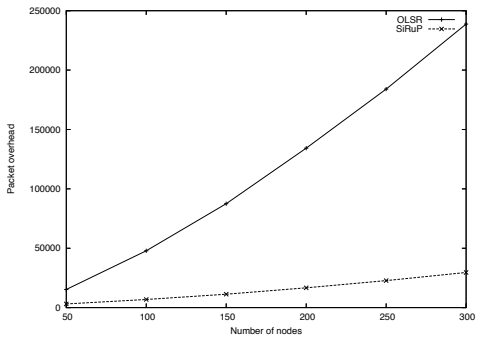

Fig. 4. Packet overhead as a function of the number of nodes

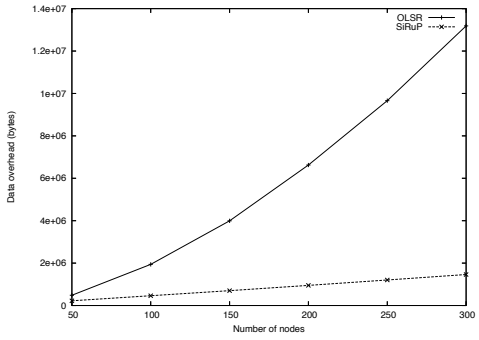

Fig. 5. Data overhead as a function of the number of nodes

is not limited to an unicast message and hence the destination has a higher chance to be reached.

Figure. 4 and Figure. 5 present the routing overhead in terms of the number of packets and the amount of data sent, respectively. The gap between OLSR and SiRuP can be explained by the different approaches used in each protocol. On one hand, OLSR has a high overhead for updating information on the network topology, but has a smaller cost for route construction. On the other hand, SiRuP minimizes the redundancy of data in HELLO packets and offers a lower constant overhead, but the route request broadcast depends on the density. As shown below, the gap is smaller when the number of requests per node increases.

B. Second Scenario: Policies

In the second scenario, we are going to evaluate various policies. Table. II presents the policies and Figure. 6 presents the results in terms of reachability and data overhead (the packet overhead is almost equal for each policy). The difference between each policy is low as for the reachability

TABLE II

POLICIES USED IN THE EXPERIMENTATION

	$size\_hello$	$\|selected_1\|$	$\|selected_2\|$	$\|selected_{\geq 3}\|$
policy_A	20	8	4	8
policy_B	10	4	2	4
policy_C	5	2	1	2

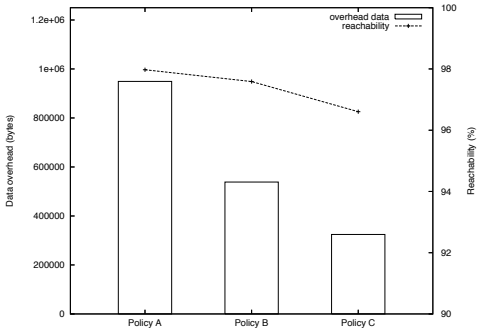

Fig. 6. SiRuP performance with different policies

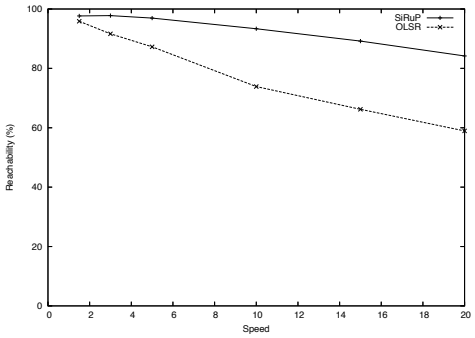

Fig. 7. Reachability as a function of the node speed

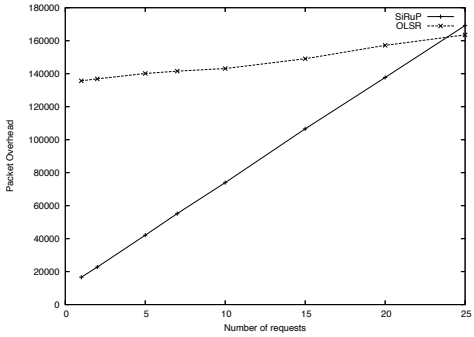

Fig. 8. Packet overhead as a function of the number of requests

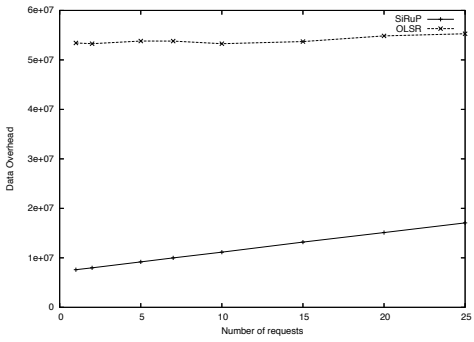

Fig. 9. Data overhead as a function of the number of requests

(around 1.5%), but the data overhead is as much as three times smaller in some cases. Depending on the efficiency constraints, choosing a policy with less information in the HELLO packet results in only a slight performance degradation, and it might be interesting to choose a light policy if the applications can cope well with network failures.

C. Third Scenario: Node Speed

In the third scenario, we compare OLSR and SiRuP once again, this time with a variable node speed ranging from 1.5m/s to 20m/s. Figure. 7 presents the results in term of reachability. Because SiRuP uses a size-limited broadcast for route request, it is less influenced by the node mobility, as the reactive nature of the route requests permits a blind flooding-like research (with good reachability) but without covering the entire network. OLSR offer lowers performance in term of reachability due to frequent link changes, since the multicast relay point (MPR) set needs at least two message exchanges to construct a viable MPR set and because the time between two topology control (TC) messages is important.

D. Fourth Scenario: Number of Requests

The last scenario increases the number of route requests, each node sends from 1 to 25 route requests during the simulation (for a total of 200 to 5000 route requests). As seen in Figure. 8, OLSR becomes more interesting in terms of the packet overhead when each node sends 23 or more requests during the simulation (*i.e.*, a total of 4600 route requests during 90 seconds). In fact, most of the overhead comes from control messages (HELLO or TC messages), as route request and reply messages are forwarded like simple data packet when routes already exist. Thus, OLSR can increase the number of route requests at a lower cost, instead of the higher cost for each route request (with SiRuP). In terms of data overhead, as seen in Figure. 9, OLSR also becomes more interesting than SiRuP for a higher number of route requests. Because the data overhead for each of the two protocols is a linear function, we can extrapolate that OLSR becomes better than SiRuP when the number of route requests by node grows beyond 116 (23200 route requests in the network during 90 seconds, *i.e.*, 290 route requests by second on average). This value is very high for an ad-hoc network and OLSR should be used only for dense networks with heavy route requests.

V. Related Work

Ad-hoc routing protocols can be classified into three categories: distance vector, link state, and on-demand. The first two categories are also known as proactive (or table-driven) routing protocols: they maintain global routes for all known destinations in the background. By exchanging route information, proactive protocols minimize delay for route construction. In contrast, on-demand routing protocols, also known as reactive protocols, discover the route when it is needed. Several surveys cover these protocols [9]–[11].

Distance vector protocols belong to the class of destination based protocols. Each node maintains a distance (in number of hops) and a vector (next-hop identifier) to all the destinations

in the network. Distributed Bellman-Ford (DBF) is a classical distance-vector algorithm: each node maintains the distance for each destination in the network and periodically broadcasts to each of its neighbors the current estimate of the shortest distance to every node. It is well adjusted for wired networks, but suffers of loop formation in ad-hoc networks due to the fluctuating nature of this environment. Destination-Sequence Distance Vector (DSDV) by Perkins *et al.* [1] is an ad-hoc routing protocol based on DBF, using a tag for each route entry with a sequence number to help nodes to select up to date information and thus avoid formation of routing loops. But DSDV suffers from a constant high overhead, because each node periodically exchanges its routing table with their neighbors. When the density and/or the size of the network grows, nodes suffer due to transfering too much information and too many collisions at the MAC layer. SiRuP uses less information by avoiding taht neighbor send redundant data.

The Link state (LS) approach is preferred for wired networks. Protocols regularly flood the network with link state updates to refresh the routing tables in all routers. By applying a shortest-path algorithm, each router is able to choose its next hop for each destination. For instance, GSR (Global State Routing protocol) [12] reduces the number of accesses to the MAC layer by periodically broadcasting changes in its connectivity instead of immediately flooding the network when a link to a neighbor change. FSR (Fisheye State Routing) [13] extends GSR, by using different exchange periods for different entries in the routing table: entries corresponding to close nodes are propagated with higher frequency. This idea comes from the *distance effect* described by Basagni *et al.* [14]: *"the greater the distance separating two nodes, the slower they appear to be moving in respect of each other"*. Another protocol, called HSLS (Hazy Sighted Link State) [15], uses a similar idea: link changes are collected and diffused with a time to live (TTL) that depends on the current time index. More precisely, a node floods recent neighborhood changes every t_e (the time interval between two control messages) with TTL set to 2, every $2 * t_e$ with TTL set to 4, every $4 * t_e$ with TTL set to 8, etc... Hence, a distant node receives fewer updates than a close node. The protocol presented in this paper uses a similar concept for the diffusion of routing information, but enhances the method by avoiding data redundancy in the payload of HELLO messages. The idea of decreasing the amount of control information about distant nodes is used in SiRuP to lower the control message overhead.

On-demand routing protocols (also known as reactive) discover the route between two nodes when needed. The source node broadcasts a route request message. Intermediate nodes relay the message if needed and discard duplicates. When the destination receives the request message, it sends a route reply to the source, and this message instantiates routing information of intermediate nodes until it reaches the source. AODV (Ad hoc On-demand Distance Vector) [2] uses an equivalent method but adds sequence numbers to ensure the freshness of each route. DSR (Dynamic Source Routing) [16] emphasizes aggressive caching and deduction of topology information from routing packets, by adding in each route request and route reply message the list of all intermediate nodes. Hence, each node listening to these messages can extract route information to all downstream nodes. However, because of the high number of broadcast messages for route construction, neither AODV nor DSR are efficient for dense networks. SiRuP uses a reactive approach when a route is needed, but avoids broadcasting the information in the whole network, thus reducing the impact of the route research to a limited subset of the network. TORA (Temporally-Ordered Routing Algorithm) [17] tries to limit the routing overhead by broadcasting any topological changes to a very small set of nodes near the occurrence of a topological change. To accomplish this, the protocol maintains for each destination a directed acyclic graph (DAG) in a distributed fashion. Each node has the downstream or upstream links to each of its neighbors. When a link change happens in the neighborhood, the change is spread only to the nodes where the link status has to change. TORA succeeds in reducing the routing overhead but suffers of instability. Nevertheless, the idea of local adaptation is interesting to limit the impact of a change to a small portion of the network, hence SiRuP uses an equivalent approach to reduce the route request broadcast cost inside the network.

Hybrid protocols are designed to combine the advantages of proactive and reactive routing strategies. For instance, ZRP (Zone Routing Protocol) [18] defines zones as the set of zone-radius hop neighbors (*i.e.*, the hop distance from the node to each of its zone-included nodes has to be smaller than a fixed value). For intra-zone routing protocol, ZRP uses a distance vector algorithm. But when a source has no route to a destination, it invokes an inter-zone routing protocol to reach other zones. The combination of both approaches (proactive and reactive) offers more flexible protocols. In fact, SiRuP is a hybrid protocol using a low-cost proactive strategy to reduce the overhead of reactive route requests but, differently from ZRP, uses the combination of both approaches instead of using different strategies, depending on the distance to the destination.

Broadcast (diffusion of a message from a source node to all nodes in the network) is a common operation in ad-hoc networks, and it is used by several routing protocols. Flooding (also called blind broadcast) is the simplest broadcast protocol: each node rebroadcasts the message once and discards duplicates. AODV, SLS, GSR, DSR and HSLS use flooding with various improvements (usually by changing the TTL value of the broadcast packet to limit propagation in the network). The flooding approach is reliable but has a high overhead for the routing protocol (in term of number of packets and MAC layer access) and the number of collisions dramatically increases in the case of dense networks.

The problem, called *"broadcast storm"* by Ni *et al.* [19] has been addressed by several papers, for instance with MPR, used in the protocol OLSR. Wu and Li [20] proposed a distributed deterministic algorithm to compute a dominating set. If G be the graph of a given wireless network, a set is dominating if

all nodes in G are either in the set, or neighbors of nodes belonging to the set. Hence, only nodes from the dominating set rebroadcast the message. Stojmenovic *et al.* [21] proposed some improvements to this model by using node ids as primary key and neighbor elimination scheme to reduce failure due to node mobility.

But the dominating set concept and MPR suffer in the case of dense networks because they need to know the two-hop neighbors toplogy (*i.e.*, the identifiers of one-hop and two-hop neighbors, the links between one-hop neighbors and the links between one-hop neighbors and two-hop neighbors). Therefore, each node has to send periodically a list of one-hop neighbors. If d is the average density of the network, the each node receives data in order of $\Theta(d^2)$. Instead of reducing the overlap of broadcast messages, SiRuP reduces the broadcast cost by avoiding redundancy data between neighbors: if a node sends routing data, then neighbors try to avoid to rebroadcast the same data.

VI. Conclusion

In this paper, we have proposed a new routing algorithm for a mobile ad-hoc networks, with a reduced overhead in case of dense networks. It is particulary efficient in case of a high density of nodes with high mobility and a limited number of requests. The reachability is comparable with OLSR, though a greater reduction in terms of packets and data. The idea of avoiding redundancy of data between neighbors is efficient because of the nature of the wireless medium: every node is able to overhear any message sent by its neighbors. The weaker diffusion of identifiers (when compared to OLSR for instance) enables the use of a size-limited broadcast and avoid the contamination of request messages in the network.

As for future works, we plan to adapt SiRuP to different environments. Currently, SiRuP is efficient for dense networks with a limited number of route requests. This is due to the high cost of the route request, because the route request uses a limited blind flooding. Even if the broadcast is not propagated in the network, this still induces a "broadcast tempest" in the local zone between source and destination, especially when the density increases. This shortcomingt comes from the fact that the number of packets for a route request depends on both route length and the local density between source and destination. We plan to work on the variable size of the HELLO packet (adjusted in function of the density) and broadcast reduction algorithm to reduce this drawback.

Acknowledgments

We would like to express our gratitude to Péter Urbán and Matthias Wiesmann for their helpful comments, and to Francisco J. Ros for his valuable advices about the implementation of OLSR.

References

[1] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance vector (DSDV) for mobile computers," *ACM SIGCOMM '94 Computer Communications Review*, vol. 24, no. 4, pp. 234–244, Oct. 1994.

[2] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999, pp. 90–100.

[3] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," in *Proc. IEEE International Multi-Topic Conference (INMIC 2001)*, Dec. 2001.

[4] G. Pei, M. Gerla, and T. Chen, "Fisheye state routing: A routing scheme for ad hoc wireless networks," in *IEEE International Conference on Communications (ICC 2000)*, vol. 1, June 2000, pp. 70–74.

[5] T. Clausen, P. Jacquet, A. L. P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot, "Optimized link state routing protocol (olsr)," IETF MANET Working Group," RFC, Oct. 2003.

[6] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proc. 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, Jan. 2002.

[7] A. Varga, "The Omnet++ discrete event simulation system," in *Proc. of the European Simulation Multiconference (ESM'2001)*, June 2001.

[8] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl, "A mobility framework for omnet++," in *3rd International OMNeT++ Workshop*, 2003 Jan.

[9] S. Ramanathan and M. Steenstrup, "A survey of routing techniques for mobile communications networks," *ACM/Baltzer Mobile Networks and Applications*, vol. 1, no. 2, pp. 89–104, 1996.

[10] L. Feeney, "A taxonomy for routing protocols in mobile ad hoc networks," SICS (Swedish Institute of Computer Science), Tech. Rep. T99/07, Oct. 1999.

[11] E. Royer and C.-K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications Magazine*, pp. 46–55, Apr. 1999.

[12] T. Chen and M. Gerla, "Global state routing: A new routing scheme for ad-hoc wireless networks," in *IEEE International Communications Conference (ICC98)*. IEEE, June 1998, pp. 171–175.

[13] G. Pei, M. Gerla, and T. Chen, "Fisheye state routing in mobile ad hoc networks," in *Proc. of ICDCS Workshop on Wireless Networks and Mobile Computing*, Apr. 2000, pp. D71–D78.

[14] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (dream)," in *Fourth ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, Dallas, Texas, Oct. 1998, pp. 76–84.

[15] C. Santivanez, S. Ramanathan, and I. Stravrakakis, "Making link state routing scale for ad hoc networks," in *Proc. of the 2nd ACM international symposium on Mobile ad hoc networking & computing (MobiHOC'2001)*, Oct. 2001, pp. 22–32.

[16] D. Johnson, "Routing in ad hoc networks of mobile hosts," in *Workshop on Mobile Computing Systems and Applications*, 1994.

[17] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM '97*, Apr. 1997.

[18] Z. Haas, "The routing algorithm for the reconfigurable wireless networks," in *ICUPC'97*, San Diego, CA, Oct. 1997.

[19] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. MobiCom'99*, Aug. 1999, pp. 151–162.

[20] J. Wu and H. Li, "A dominating-set-based routing scheme in ad hoc wireless networks," in *Proc. 3rd Int'l Workshop Discrete Algorithms and Methos for Mobile Computing and Comm (DIALM'99)*, Aug. 1999, pp. 7–14.

[21] I. Stojmenović, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, Jan. 2002.