Optimized Link State Routing Protocol for Ad Hoc Networks

P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot*

Hipercom Project, INRIA Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France

Abstract: In this paper we propose and discuss an optimized link state routing protocol, named OLSR, for mobile wireless networks. The protocol is based on the link state algorithm and it is proactive (or table-driven) in nature. It employs periodic exchange of messages to maintain topology information of the network at each node. OLSR is an optimization over a pure link state protocol as it compacts the size of information sent in the messages, and furthermore, reduces the number of retransmissions to flood these messages in entire network. For this purpose, the protocol uses multipoint relaying technique to efficiently and economically flood its control messages. It provides optimal routes in terms of number of hops, which are immediately available when needed. The proposed protocol is best suitable for large and dense ad hoc networks.

keywords: routing protocol, link state protocol, proactive or table-driven protocol, mobile wireless networks, ad hoc networks

1 Introduction

With the advent of new technologies and the demand for flexibility and ease in working environment, the use of mobile wireless computing is growing fast. Besides their use, mobile wireless networks are assumed to grow in size too. They can function in independent groups, containing some tens of nodes up to several hundreds of nodes. As the network size increases, it becomes common for the nodes to be dispersed in a larger area than the radio range of individual nodes. Under such conditions, one has to employ routing techniques such that the out of range nodes may communicate with each other via intermediate nodes. This problem of routing in mobile ad hoc networks is our focus of discussion in this paper, and a protocol is proposed as a solution.

Design issues for developing a routing protocol for wireless environment with mobility are very different and more complex than those for wired networks with static nodes. Major problems in mobile ad hoc networks are (a) limited bandwidth and (b) high rate of topological changes. Thus the goal for a routing protocol is to minimize its control traffic overhead while at the same time, it should be capable of rapidly adapting to link failures and additions caused by node movements. It implies, therefore, that the routing protocol should work in a distributed manner and it should be self starting and self organizing. The possibility of ad hoc networks to grow in size to have large diameters brings with it the scaling up problem, possibility of loops in the routes, and inconsistency of information in different parts of the network. Moreover, the existence of uni-directional links is a real challenge for routing protocols.

2 Reactive versus proactive routing approach

Different routing protocols try to solve the problem of routing in mobile ad hoc networks in one way or the other. In reactive routing approach, a routing protocol does not take the initiative for finding a route to a destination, until it is required. The protocol attempts to discover routes only "on-demand" by flooding its query in the network. This type of protocols reduces control traffic overhead at the cost of increased latency in finding the route to a destination. The examples of this kind of protocols are AODV [9], DSR [5] and TORA [7]. On the other hand, proactive protocols are based on periodic exchange of control messages. Some messages are sent locally to enable a node to know its local neighborhood, and some messages are sent in entire network which permit to exchange the knowledge of topology among all the nodes of the network. The proactive protocols immediately provide the required routes when needed, at the cost of band-

*[Philippe.Jacquet, Paul.Muhlethaler, Thomas.Clausen, Anis.Laouiti, Amir.Qayyum, Laurent.Viennot]@inria.fr

width used in sending frequent periodic updates of topology. The examples of this kind of protocols are DSDV [8], STAR [2] and TBRPF [6]. Some protocols use a mixture of the two techniques, i.e., they keep routes available for some destinations all the time, but discover routes for other destinations only when required. [3] analyses some routing protocols for ad hoc networks.

3 OLSR (Optimized Link State Routing) protocol

3.1 Overview

We propose a proactive routing protocol for mobile ad hoc networks, which we call as Optimized Link State Routing (OLSR). The protocol inherits the stability of the link state algorithm. Due to its proactive nature, it has an advantage of having the routes immediately available when needed. In a pure link state protocol, all the links with neighbor nodes are declared and are flooded in the entire network. OLSR protocol is an optimization of a pure link state protocol for mobile ad hoc networks. First, it reduces the size of control packets: instead of all links, it declares only a subset of links with its neighbors who are its *multipoint relay selectors* (see Section 3.2). Secondly, it minimizes flooding of this control traffic by using only the selected nodes, called *multipoint relays*, to diffuse its messages in the network. Only the multipoint relays of a node retransmit its broadcast messages. This technique significantly reduces the number of retransmissions in a flooding or broadcast procedure [10, 12].

Apart from normal periodic control messages, the protocol does not generate extra control traffic in response to link failures and additions. The protocol keeps the routes for all the destinations in the network, hence it is beneficial for the traffic patterns where a large subset of nodes are communicating with each other, and the [source, destination] pairs are also changing with time. The protocol is particularly suitable for large and dense networks, as the optimization done using the multipoint relays works well in this context. More dense and large a network is, more optimization is achieved as compared to the normal link state algorithm.

The protocol is designed to work in a completely distributed manner and thus does not depend upon any central entity. The protocol does not require a reliable transmission for its control messages: each node sends its control messages periodically, and can therefore sustain a loss of some packets from time to time, which happens very often in radio networks due to collisions or other transmission problems. The protocol also does not need an in-order delivery of its messages: each control message contains a sequence number of most recent information, therefore the re-ordering at the receiving end can not make the old information interpreted as the recent one.

OLSR protocol performs hop by hop routing, i.e. each node uses its most recent information to route a packet. Therefore, when a node is moving, its packets can be successfully delivered to it, if its speed is such that its movement could be followed in its neighborhood, at least. The protocol thus supports a nodal mobility that can be traced through its local control messages, which depends upon the frequency of these messages.

3.2 Multipoint relays

The idea of multipoint relays [1] is to minimize the flooding of broadcast packets in the network by reducing duplicate retransmissions in the same region. Each node in the network selects a set of nodes in its neighborhood, which retransmits its packets. This set of selected neighbor nodes is called the multipoint relays (*MPRs*) of that node. The neighbors of any node N which are not in its *MPR* set, read and process the packet but do not retransmit the broadcast packet received from node N. For this purpose, each node maintains a set of its neighbors which are called the *MPR Selectors* of the node. Every broadcast message coming from these *MPR Selectors* of a node is assumed to be retransmitted by that node. This set can change over time, which is indicated by the selector nodes in their *HELLO* messages (see Section 4.1).

Each node selects its multipoint relay set among its one hop neighbors in such a manner that the set covers (in terms of radio range) all the nodes that are two hops away. The multipoint relay set of node N, called $MPR(N)$, is an arbitrary subset of the neighborhood of N which satisfies the following condition: every node in the two hop neighborhood of N must have a bidirectional link toward $MPR(N)$. The smaller is the multipoint relay set, the more optimal is the routing protocol. Figure 1 shows the multipoint relay selection around node N.

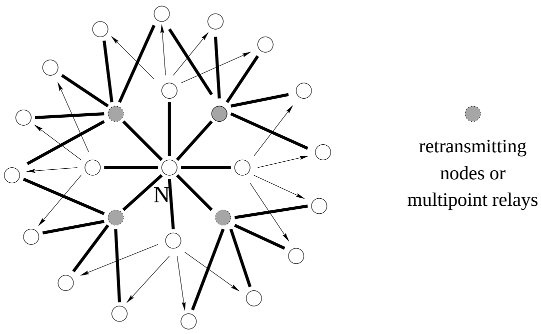

Figure 1: Multipoint relays

OLSR protocol relies on the selection of multipoint relays, and calculates its routes to all known destinations through these nodes, i.e. *MPR* nodes are selected as intermediate nodes in the path. To implement this scheme, each node in the network periodically broadcast the information about its one-hop neighbors which have selected it as a multipoint relay. Upon receipt of this *MPR Selectors'* information, each node calculates and updates its routes to each known destination. Therefore, the route is a sequence of hops through the multipoint relays from source to destination.

Multipoint relays are selected among the one hop neighbors with a bi-directional link. Therefore, selecting the route through multipoint relays automatically avoids the problems associated with data packet transfer on uni-directional links. Such problems may consist of getting an acknowledgment for data packets at each hop which cannot be received if there is a uni-directional link in the selected route.

4 Protocol functioning

4.1 Neighbor sensing

Each node must detect the neighbor nodes with which it has a direct and bi-directional link. The uncertainties over radio propagation may make some links uni-directional. Consequently, all links must be checked in both directions in order to be considered *valid*.

To accomplish this, each node periodically broadcasts its *HELLO* messages, containing the information about its neighbors and their link status. These control messages are transmitted in the broadcast mode. These are received by all one-hop neighbors, but they are not *relayed* to further nodes. A *HELLO* message contains:

- the list of addresses of the neighbors to which there exists a valid *bi-directional* link;
- the list of addresses of the neighbors which are heard by this node (a *HELLO* has been received) but the link is not yet validated as *bi-directional*: if a node finds its own address in a *HELLO* message, it considers the link to the sender node as *bi-directional*.

Remark: The list of neighbors in the *HELLO* message can be partial, the rule being that all neighbor nodes are cited at least once within a predefined refreshing period.

These *HELLO* messages permit each node to learn the knowledge of its neighbors up to two hops. On the basis of this information, each node performs the selection of its multipoint relays. These selected multipoint relays are indicated in the *HELLO* messages with the link status *MPR*. On the reception of *HELLO* messages, each node can construct its *MPR Selector* table with the nodes who have selected it as a multipoint relay.

In the neighbor table, each node records the information about its one hop neighbors, the status of the link with these neighbors, and a list of two hop neighbors that these one hop neighbors give access to. The link status can be uni-directional, bi-directional or *MPR*. The link status as *MPR* implies that the link with the neighbor node is bi-directional AND that node is also selected as a multipoint relay by this local node. Each entry in the neighbor table has an associated holding time, upon expiry of which it is no longer valid and hence removed.

The neighbor table also contains a sequence number value which specifies the most recent *MPR* set that the local node keeping this neighbor table has selected. Every time a node selects or updates its *MPR* set, this sequence number is incremented to a higher value.

4.2 Multipoint relay selection

Each node of the network selects independently its own set of multipoint relays. The *MPR* set is calculated in a manner to contain a subset of one hop neighbors which covers all the two hop neighbors, i.e., the union of the neighbor sets of all *MPRs* contains the entire two hop neighbor set. In order to build the list of the two hop nodes from a given node, it suffices to track the list of bi-directional link nodes found in the *HELLO* messages received by this node (this two-hop neighbor information is stored in the neighbor table). The *MPR* set need not be optimal, however it should be small enough to achieve the benefits of multipoint relays. By default, the multipoint relay set can coincide with the whole neighbor set. This will be the case at network initialization. One possible algorithm for selecting these *MPRs* is presented in [11], which is analysed in [13] and [10], and improved in [12].

Multipoint relays of a given node are declared in the subsequent *HELLOs* transmitted by this node, so that the information reaches the multipoint relays themselves. The multipoint relay set is re-calculated when:

- a change in the neighborhood is detected when either a bi-directional link with a neighbor is failed, or a new neighbor with a bi-directional link is added; or
- a change in the two-hop neighbor set with bi-directional link is detected.

With information obtained from the *HELLO* messages, each node also construct its *MPR Selector* table, in which it puts the addresses of its one hop neighbor nodes which has selected it as a multipoint relay along with the corresponding *MPR* sequence number of that neighbor node. A sequence number is also associated to the *MPR Selector* table which specifies that the *MPR Selector* table is most recently modified with that sequence number. A node updates its *MPR Selector* set according to the information it receives in the *HELLO* messages, and increment this sequence number on each modification.

4.3 MPR information declaration

In order to build the *intra-forwarding* database needed for routing packets, each node broadcasts specific control messages called *Topology Control (TC)* messages. *TC* messages are forwarded like usual broadcast messages in the entire network. This technique is similar to the link state technique used in ARPANET, but it takes advantage of *MPRs* which enable a better scalability of *intra-forwarding* [4].

A *TC* message is sent periodically by each node in the network to declare its *MPR Selector* set, i.e., the message contains the list of neighbors who have selected the sender node as a multipoint relay. The sequence number associated to this *MPR Selector* set is also attached to the list. The list of addresses can be partial in each *TC* message, but parsing must be complete within a certain refreshing period. (In [1], the list of addresses is mandatorily exhaustive). The information diffused in the network by these *TC* messages will help each node to build its topology table. A node which has an empty *MPR Selector* set, i.e., nobody has selected it as a multipoint relay, may not generate any *TC* message.

The interval between the transmission of two *TC* messages depends upon whether the *MPR Selector* set is changed or not, since the last *TC* message transmitted. When a change occurs in the *MPR Selector* set, the next *TC* message may be sent earlier that the scheduled time, but after some pre-specified minimum interval, starting from the time the last *TC* message was sent. If this much time has already elapsed, the next *TC* message may be transmitted immediately. All subsequent *TC* messages are sent with the normal default interval for sending *TC* messages, until the *MPR Selector* set is changed again.

Each node of the network maintains a topology table, in which it records the information about the topology of the network obtained from the *TC* messages. A node records information about the multipoint relays of other nodes in this table. Based on this information, the routing table is calculated. An entry in the topology table consists of an address of a (potential) destination (an *MPR Selector* in the received *TC* message), address of a last-hop node to that destination (originator of the *TC* message) and the corresponding *MPR Selector* set sequence number (of the sender node). It implies that the destination node can be reached in the last hop through this last-hop node. Each topology entry has an associated holding time, upon expiry of which it is no longer valid and hence removed.

Upon receipt of a *TC* message, the following proposed procedure may be executed to record the information in the topology table:

1. If there exist some entry in the topology table whose last-hop address corresponds to the originator address of the *TC* message and the *MPR Selector* sequence number in that entry is greater than the sequence number in the received message, then no further processing of this *TC* message is done and it is silently discarded (case: packet received out of order).

2. If there exist some entry in the topology table whose last-hop address corresponds to the originator address of the *TC* message and the *MPR Selector* sequence number in that entry is smaller than the sequence number in the received message, then that topology entry is removed.

3. For each of the *MPR Selector* address received in the *TC* message:

   - If there exist some entry in the topology table whose destination address corresponds to the *MPR Selector* address and the last-hop address of that entry corresponds to the originator address of the *TC* message, then the holding time of that entry is refreshed.
   - Otherwise, a new topology entry is recorded in the topology table.

4.4 Routing table calculation

Each node maintains a routing table which allows it to route the packets for other destinations in the network. The nodes which receive a *TC* message parse and store some of the *connected pairs* of form [last-hop, node] where "nodes" are the addresses found in the *TC* message list. The routing table is built from this database by tracking the connected pairs in a descending order. To find a path from a given origin to a remote node R, one has to find a connected pair (X,R), then a connected pair (Y,X), and so forth until one finds a node Y in the neighbor set of the origin. Figure 2 explains this procedure of searching the [last-hop,destination] pairs in the topology table to get a complete, connected route from source to destination. In order to restrict to optimal paths, the forwarding nodes will select only the connected pairs on the minimal path. This selection can be done dynamically and with minimal storage facilities. The sequence numbers are used to detect connected pairs which have been invalidated by further topology changes. The information contained in the *intra-forwarding* database (topology table), which has not been refreshed is discarded. See section 5 for more details.

The route entries in the routing table consist of destination address, next-hop address, and estimated distance to destination. The entries are recorded in the table for each destination in the network for which the route is known. All the destinations for which the route is broken or partially known are not entered in the table.

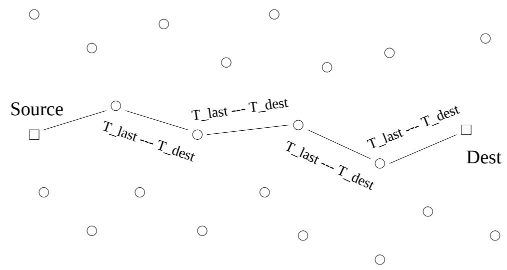

Figure 2: Building a route from topology table

The routing table is based on the information contained in the neighbor table and the topology table. Therefore, if any of these tables is changed, the routing table is re-calculated to update the route information about each known destination in the network. The table is re-calculated when a change in the neighborhood is detected concerning a bi-directional link or when a route to any destination is expired (because the corresponding topology entry is expired). The re-calculation of this routing table does not generate or trigger any packets to be transmitted, neither in the entire network, nor in the one-hop neighborhood.

The following proposed procedure may be executed to calculate (or re-calculate) the routing table :

1. All the entries of routing table are removed.

2. The new entries are recorded in the table starting with one hop neighbors ($h = 1$) as destination nodes. For each neighbor entry in the neighbor table, whose link status is not uni-directional, a new route entry is recorded in the routing table where destination and next-hop addresses are both set to address of the neighbor and distance is set to 1.

3. Then the new route entries for destination nodes $h + 1$ hops away are recorded in the routing table. The following procedure is executed for each value of h, starting with $h = 1$ and incrementing it by 1 each time. The execution will stop if no new entry is recorded in an iteration.

   - For each topology entry in topology table, if its destination address does not corresponds to destination address of any route entry in the routing table AND its last-hop address corresponds to destination address of a route entry with distance equal to h, then a new route entry is recorded in the routing table where :

– destination is set to destination adress in topology table;
– next-hop is set to next-hop of the route entry whose destination is equal to above-mentioned last-hop address; and
– distance is set to $h + 1$.

4. After calculating the routing table, the topology table entries which are not used in calculating the routes may be removed, if there is a need to save memory space. Otherwise, these entries may provide multiple routes.

5 Performance analysis

5.1 Route optimality

The main problem is to show that the introduction of a multipoint relay set as a subset of the neighbor set does not destroy the connectivity properties of the network.

Let us take the following model: we consider a network made up of a set of nodes and a set of valid links. This network can be seen as an interconnection graph. We define the usual distance $d(X, Y)$ which gives the minimal number of hops between node X and node Y. We also define $d_F(X, Y)$ as the minimal number of hops, providing that the intermediate relay nodes are forwarders. We can notice that d_F does not define a distance when some nodes are non-forwarders because triangle inequality could be not satisfied every time. In the following, we consider a safe network: *i.e.* $d_F(X, Y) < \infty$ for all pairs of nodes X and Y.

By definition of link validity and since the *HELLO* messages are periodically retransmitted, the neighbor sensing and the election of multipoint relay nodes can be performed without any particular problem, except if mobile nodes move faster than *HELLO* interval. In the following we suppose that every node has a multipoint relay set that covers its two hop neighbor set. We do not need to assume optimality of the multipoint relay set.

The last operation is topology information broadcast. If for any node its multipoint relay set coincides with its neighbor set, then the *TC* message broadcast will reach any node in a straightforward way. In this case the minimal path to a remote node received via *TC* messages would be an optimal path and the routing tables will contain the appropriate information. Our point is that this property remains valid when multipoint relay sets are strict subsets of neighbor sets.

We define for any given node the set of multipoint relays of rank 0 as the node itself and the set of multipoint relays of rank 1 as the multipoint relay set itself. Let us define the set of multipoint relays of rank $k + 1$, for k integer, as the union of multipoint relay set of all nodes element of the multipoint relay set of rank k. In other words, each element M_k of the multipoint relay set of rank k of a node X can be reached via a path $XM_1 \dots M_k$ where M_1 is multipoint relay of X, and M_{i+1} is multipoint relay of M_i.

Theorem 1 *If for two nodes X and Y, $d_F(X, Y) = k + 1$, for k integer, then Y is at distance 1 from the multipoint relay set of rank k of X.*

Proof : By recursion.

The proposition is valid for $k = 0$ and $k = 1$, by definition of the multipoint relay set. We suppose the proposition valid for k, and let us assume a node Y such that $d_F(X, Y) = k + 2$. Therefore there exists an optimal valid path $XF_1 \dots F_k F_{k+1} Y$ where the F_is are all forwarder nodes. We have $d_F(X, F_{k+1}) = k + 1$, therefore F_{k+1} is at distance 1 from the multipoint relay set of rank k. Let M_k be the element of the multipoint relay set of rank k of X such that $d_F(M_k, F_{k+1}) = 1$; therefore $d_F(M_k, Y) = 2$ and Y belongs to the two hop neighbor set of M_k. Let M_{k+1} be the multipoint relay of M_k which covers Y: $d(M_{k+1}, Y) = 1$. Since M_{k+1} belongs to the multipoint relay set of rank $k + 1$ the theorem is proved.

5.2 Broadcast performance

Theorem 2 *For all pairs of nodes X and Y, X generating and transmitting a broadcast packet P, Y receives a copy of P.*

Proof : By reversed recursion.

We suppose that transmissions are error free but are subject to arbitrary finite delays. Let k be the closest distance to Y from which a copy of packet P has been eventually (re)transmitted. We shall prove that $k = 1$.

Let F be the first forwarder at distance k ($k \geq 2$) from Y, which has retransmitted P. There exists a multipoint relay F' of F which is at distance $k-1$ of Y. To be convinced: we imagine a path of length k from F to Y: $F, F_1, F_2 \ldots F_{k-1}, Y$ and we take for F' the multipoint relay of F which covers F_2).

Since F' received a copy of P the first time from F (the prior transmitters are necessarily two-hops away from F'), therefore F' will automatically forward P: packet P will be retransmitted at distance $k-1$ from Y. The theorem is proved.

Note that if the transmissions are prone to errors, then there is no guarantee of correct packet reception by the intended destination. But this is a common problem to all unreliable communications networks which need upper layer recovery procedure.

6 Conclusions

For mobile wireless networks, the performance of a routing protocol is coupled with many factors, like the choice of physical technology, link layer behavior, etc. The overall behavior of a protocol specifies its working domain for which it could be suitable. OLSR protocol is proactive or table driven in nature, hence it favors the networking context where this all-time-kept information is used more and more, and where route requests for new destinations are very frequent. The protocol also goes in favor of the applications which do not allow long delays in transmitting data packets. OLSR protocol is adapted to the network which is dense, and where the communication is assumed to occur frequently between a large number of nodes.

References

[1] ETSI STC-RES10 Committee. Radio Equipment and Systems: High Performance Radio Local Area Network (HIPERLAN) Type 1, Functional specifications, June 1996. ETS 300-652.

[2] J. J. Garcia-Luna-Aceves and M. Spohn. Source-Tree Routing in Wireless Networks. In *Proc. IEEE ICNP'99*, November 1999.

[3] P. Jacquet and A. Laouiti. Analysis of Mobile Ad-hoc Network Routing Protocols in Random Graph Models. Technical Report RR-3835, INRIA, 2000.

[4] Philippe Jacquet, Pascale Minet, Paul Muhlethaler, and Nicolas Rivierre. Increasing reliability in cable-free Radio LANs: Low level forwarding in HIPERLAN. *Wireless Personal Communications*, volume 4, pages 65–80, January 1997.

[5] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, volume 5, pages 153–181, 1996. Kluwer Academic Publishers, edited by T. Imielinski and H. Korth.

[6] Richard G. Ogier. Efficient Routing Protocols for Packet-Radio Networks Based on Tree Sharing. In *Proc. 6th IEEE Intl. Workshop on Mobile Multimedia Communications (MOMUC'99)*, November 1999.

[7] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proc. IEEE Conference on Computer Communications INFOCOM '97*, pages 1405–1413. Kobe, Japan, April 1997.

[8] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers. In *Proceedings of ACM SIGCOMM'94*, pages 234–244, September 1994. London, UK.

[9] Charles E. Perkins and Elizabeth M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, February 1999. New Orleans, LA.

[10] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical Report RR-3898, INRIA, February 2000.

[11] Amir Qayyum. Wireless Networks: Hiperlan. Master's thesis, Université de Paris-Sud, Orsay, France, September 1996.

[12] Amir Qayyum. *Analysis and Evaluation of Channel Access Schemes and Routing Protocols in Wireless LANs.* PhD thesis, Université de Paris-sud, Orsay, France, 2000.

[13] Laurent Viennot. Complexity Results on Election of Multipoint Relays in Wireless Networks. Technical Report Research Report # 3584, INRIA, December 1999.