

# MP-OLSR: Multi-Path OLSR for Mobile Ad hoc Networks

Eddy Cizeron

Institut de Recherche en Communications  
 et en Cybernétique de Nantes  
 Rue Christian Pauc BP 50609  
 44306 Nantes cedex 3 France  
 Email: Eddy.Cizeron@univ-nantes.fr

Salima Hamma

Institut de Recherche en Communications  
 et en Cybernétique de Nantes  
 Rue Christian Pauc BP 50609  
 44306 Nantes cedex 3 France  
 Email: Salima.Hamma@ircsyn.ec-nantes.fr

**Abstract**—One of the main challenges for mobile ad hoc networks is to design routing protocols that cope efficiently with dynamic network topologies. In this paper, we propose a new routing protocol called Multi-Path Optimized Link State Routing Protocol (MP-OLSR). The main idea is to use the sizeable amount of information collected by every node in OLSR in order to select not one but several reliable routes used simultaneously. The information is not merely parcelled out between these routes, but coded using a Multiple Description method, which reduces the dependency on topology changes. Each route among the  $N$  selected ones is used to transmit a specific description. Any subset of  $M$  routes among these  $N$  routes is sufficient to rebuild the initial data information. Furthermore, thanks to path diversity, the local bitrate is reduced. This feature may be interesting in the case of multimedia information exchanges (such as video) over mobile ad hoc networks. Performance analysis of this new algorithm shows that the paths reliability is improved. In particular, load balancing for high bitrates and dense networks enables to reach interesting performances.

## I. INTRODUCTION

A Mobile Ad hoc NETWORK (MANET) is a collection of mobile wireless devices. The main characteristic of such a network is the perpetual change of topology due to mobility, appearance and disappearance of the nodes. For each pair of nodes in the network that have to communicate, the routing protocol must ensure the construction and the maintenance of at least one multi-hop path. In this context, certifying the stability of the transfer (by maintaining a constant delay and limiting retransmission) is a highly challenging issue.

Among the large amount of already existing protocols, a usual categorization discriminates reactive and proactive ones. In reactive protocols, routes are built on demand while in proactive protocols, recurring updates ensure that a path to every destination is determined a priori. In this paper, we will focus on proactive protocols and particularly on *Optimized Link State Routing* protocol (OLSR) [2]. We propose an improvement of OLSR route selection process and a new organisation of data transmissions.

The remainder of this paper is organized as follows: in section 2, we present other multi-path routing protocols proposed in literature. In section 3, we briefly introduce the principle of the multiple description coding and its practical application using multiple routes discovered by MP-OLSR algorithm. The

path selection algorithm is then described in section 4. The main hypotheses and metrics are given in the same section, as well as the description of the functions used as parameters of our algorithm. Section 5 gives the simulation results, focusing on the reliability criterium, and their analysis. Eventually, section 6 concludes our paper and presents outlooks.

## II. MULTI-PATH ROUTING PROTOCOLS

A usual method proposed in literature consists in using only one route among the determined ones. In case of transmission failure, it is replaced almost instantaneously by an alternate route. The *Split Multi-path Routing* protocol (SMR) proposed in [3] focuses on the selection of multiple routes of maximally disjoint paths to prevent certain nodes from being congested but distribute information in only two routes per session.

The *Multi-path Dynamic Source Routing* protocol (MP-DSR) defines a new QoS metric, end-to-end reliability in order to select a subset of end-to-end paths to provide increased stability and reliability of routes [3]. In [5] The *On Demand Multi-path routing* builds multiple routes but uses only the primary route while alternate routes are used only when the primary one fails.

Other solutions are based on an optimal method [8] such as in [6]. The authors build completely disjoint routes (i.e. disjoint by nodes and links) which limited the number of routes. In [1] authors proposed shortest paths (i.e. minimum number of hops) with fewer number of disjoint nodes between two paths. Consequently, such algorithms are not appropriate for multiple description coding purpose.

## III. THE MULTIPLE DESCRIPTION CODING APPROACH

Multiple Description Coding (MDC) refers to a group of data representation and transformation methods which purpose is to improve the transmitted information integrity by transforming it into a set of redundant data packets called descriptions.

### A. MDC principle

Given a piece of information  $I$ , a multiple description coding method generates  $N$  independently communicable packets  $(D_1, D_2, \dots, D_N)$ . Each description  $D_i$  is generally

much smaller than the original information. However, the total size of all descriptions is higher than the size of the original message  $I$ . This set of descriptions is such that there exists an integer  $M$  ( $0 \leq M \leq N$ ) such that every subset of descriptions containing at least  $M$  different descriptions is sufficient to rebuild  $I$ . Thus, the higher is  $M$ , the lower is the redundancy. In particular,  $M = 1$  (respectively  $M = N$ ) corresponds to the case where  $(D_i)_{i \in [1, N]}$  are copies of  $X$  (respectively where  $(D_i)_{i \in [1, N]}$  are different pieces of  $I$ ). For a detailed review, refer to [7].

### B. MDC, OLSR and MP-OLSR

The operation of OLSR, as in every proactive link-state routing protocol, consists in two main steps: topology discovery and computation of the shortest path in order to determine the most appropriate next hop for every potential destination. In the first step, each node floods the network with a packet that contains its neighborhood information. This information, collected by all the other nodes, allows them to build a virtual representation of the actual network topology. A shortest path algorithm (generally Dijkstra's algorithm) provides the best path to reach each destination. In fact, for a given destination, only the next node of the path is recorded. No global view of paths is considered: an intermediate node of a given communication only supervises the next hop. In the proposed approach, several routes are determined for a given destination. One major difference with OLSR strategy is that routes are not selected repeatedly after every flooding procedure, but only when a destination has to be reached. Furthermore, the source entirely determines the routes. For each route, a description  $D_i$  is then generated and sent. By propagating through redundant paths, information is less influenced by route failures. Even if a certain number of the  $N$  paths vanishes, if at least  $M$  are valid, the communication can go on.

## IV. ROUTE SELECTION ALGORITHM

The aim of this procedure is to build a set  $\mathcal{K}$  of  $N$  paths, with no loops, joining a source node (noted  $s$ ) and a destination node (noted  $d$ ). This set must comply with the following conditions: the number of elements (nodes and links) shared by two distinct routes of  $\mathcal{K}$  is as small as possible; the cost of each route of  $\mathcal{K}$  is as small as possible. It appears that these two properties cannot be perfectly satisfied at the same time.

### A. Hypotheses

An ad hoc network is represented by a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$  where  $\mathcal{V}$  is the set of vertices,  $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$  the set of arcs and  $c : \mathcal{V} \rightarrow \mathcal{R}^{*+}$  a strictly positive cost function. We assume the graph is initially undirected (i.e.  $(v_1, v_2) \in \mathcal{E} \Rightarrow (v_2, v_1) \in \mathcal{E}$  and  $c(v_1, v_2) = c(v_2, v_1)$ ) and loopless (i.e. no arcs joining a node to itself). We also assume that any pair of vertices cannot be connected by more than one arc. Given an ordered pair of distinct vertices  $(s, d)$  we can define a path between  $s$  and  $d$  as a sequence of vertices  $(v_1, v_2, \dots, v_m)$  such that  $(v_q, v_{q+1}) \in \mathcal{E}$ ,  $v_1 = s$  and  $v_m = d$ .

### B. Path selection metric

The above representation implies that we define precisely what the cost function  $c$  refers to in an ad hoc context. Generally the cost of a link is an quantitative assessment of its quality. The cost is additive and as small as the link is considered good. For example, it can be the delay necessary to deliver information through it, a quantity based on the bit error rate, or on the dependency on interferences.

Considering the period  $T$  between two successive updates for the same link  $e$ , we define  $\pi_e$  as the probability that the link  $e$  is actually able to transmit the data during  $T$ . Of course  $\pi_e$  depends on the physical properties of the environment, but it also reckons with the vertices mobility and the local stability of the neighbourhood relation. One possible way of evaluating this value could be to use statistical properties of the periodic control messages.  $c$  is then defined as  $c(e) = -\log(\pi_e)$ . This information has to be transmitted along with the neighborhood during the flooding procedure as OLSR allows it.

One can notice that the chosen metric, is not only correlated with the physical quality of links (such as mean delay, binary error ratio or more complex metrics as in [9]) but also with its limited lifespan, due to topology changes.

### C. Proposed algorithm

Let  $f_p : \mathcal{R}^{*+} \rightarrow \mathcal{R}^{*+}$  and  $f_e : \mathcal{R}^{*+} \rightarrow \mathcal{R}^{*+}$  be two functions such that  $id < f_p$  and  $id \leq f_e < f_p$  (where  $id$  is the identity function). The proposed algorithm (see 1) is applied to a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ , two vertices  $(s, d) \in \mathcal{E}^2$  and a strictly positive integer  $N$ . It provides a  $N$ -uple  $(P_1, P_2, \dots, P_N)$  of  $(s, d)$ -paths extracted from  $\mathcal{G}$ .

```

MultiPathDijkstra( $s, d, \mathcal{G}, N$ )
 $c_1 \leftarrow c$ 
 $\mathcal{G}_1 \leftarrow \mathcal{G}$ 
for  $i \leftarrow 1$  to  $N$  do
   $SourceTree_i \leftarrow Dijkstra(\mathcal{G}_i, s)$ 
   $P_i \leftarrow GetPath(SourceTree_i, d)$ 
  for all arcs  $e$  in  $\mathcal{E}$  do
    if  $e$  is in  $P_i$  OR  $Reverse(e)$  is in  $P_i$  then
       $c_{i+1}(e) \leftarrow f_p(c_i(e))$ 
    else if the vertex  $Head(e)$  is in  $P_i$  then
       $c_{i+1}(e) \leftarrow f_e(c_i(e))$ 
    else
       $c_{i+1}(e) \leftarrow c_i(e)$ 
    end if
  end for
   $\mathcal{G}_{i+1} \leftarrow (\mathcal{V}, \mathcal{E}, c_{i+1})$ 
end for
return  $(P_1, P_2, \dots, P_N)$ 

```

Algorithm 1. Calculate  $N$  routes in  $\mathcal{G}$  from  $s$  to  $d$

where  $Dijkstra(\mathcal{G}, n)$  is the standard Dijkstra's algorithm which provides the source tree of shortest paths from vertex  $n$  in graph  $\mathcal{G}$ ;  $GetPath(SourceTree, n)$  is the function that extracts the shortest-path to  $n$  from the source tree  $SourceTree$ ;

$Reverse(e)$  gives the opposite edge of  $e$ ;  $Head(e)$  provides the vertex edge  $e$  points to.

#### D. Incrementation Functions

The cost incrementation functions  $f_p$  and  $f_e$  are used at each step in order to prevent Dijkstra's algorithm from generating a new path between  $s$  and  $d$  that would be too similar to one already found. This way, they ensure a certain diversity in the final  $N$ -tuple, but contrary to what provide most of multi-path computation algorithms, generated paths need not be completely disjoint. This choice is due to several considerations explained below.

First, the number of disjoint paths is limited to the  $(s, d)$  minimal cut (defined as the size of smallest subset of edges one cannot avoid in order to connect  $s$  and  $d$ ). This minimal cut is often determined by the source and destination neighborhoods. For example, if  $s$  only has 3 distinct neighbors, one cannot generate more than 3 disjoint paths from  $s$  to  $d$ . As a consequence, this limitation of diversity may be local, the rest of the network being wide enough to provide far more than 3 disjoint paths. Another drawback of completely disjoint paths algorithms is that it may generate very long paths as every local "cutoff" can only be used once.

Secondly, we don't focus in this paper on the optimal solution. Generally, optimal algorithm (as in [8]) provides a set of disjoint paths whose global cost (sum of all path costs) is minimal. In our case, adding the cost  $c(e_1) + c(e_2)$  of successive arcs  $e_1$  and  $e_2$  is equivalent to multiplying their success probabilities and thus to consider the success probability of their concatenation. In the same manner, adding the costs of two parallel paths is equivalent to computing the probability that they both succeed. However, we do not expect the result to contain simultaneously valid paths, but enough valid paths among all paths generated. As a consequence, the interesting value is the probability that the valid number of routes during period  $T$  (between two updates) is at least a given threshold  $M$ , which is highly more difficult to maximise. Nevertheless, given that the real topology may be quite unstable, we expect a practical algorithm to be sufficient to quickly obtain interesting paths.

$f_p$  is used to increase costs of the arcs belonging to the previously path  $P_i$  (or which opposite arcs belong to it). This encourages future paths to use different arcs but not different vertices.  $f_e$  is used to increase costs of the arcs who lead to vertices of the previous path  $P_i$ . As a consequence 3 possible behaviors can be distinguished:

- if  $id = f_e < f_p$ , paths tend to be arc-disjoint;
- if  $id < f_e = f_p$ , paths tend to be vertex-disjoint (which is stronger than the previous case);
- if  $id < f_e < f_p$ , paths also tend to be vertex-disjoint, but when not possible they tend to be arc-disjoint.

#### E. Algorithmic complexity and improvements

Dijkstra's algorithm performs generally in  $O(|\mathcal{V}|^2 + |\mathcal{E}|)$  (although complexity may be reduced in the case of sparse graphs). Thus global complexity is  $O(k(|\mathcal{V}|^2 + |\mathcal{E}|))$ . However,

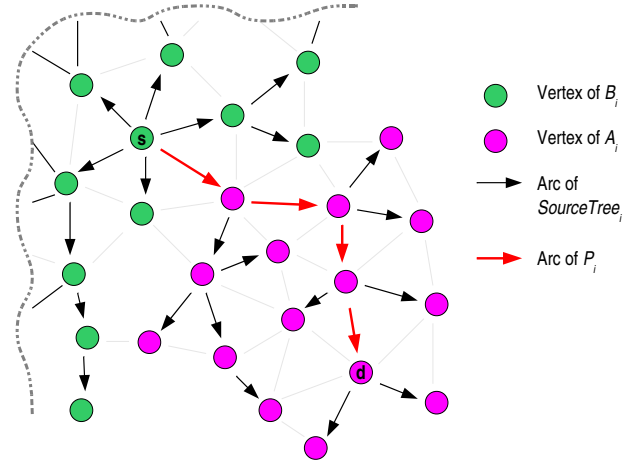


Fig. 1. Set of vertices to which Dijkstra must be re-applied at step  $i + 1$

at each step  $i$  ( $1 \leq i \leq k$ ) of our algorithm, Dijkstra's algorithm has not to be applied again to all vertices. Given path  $P_i$  let us consider  $\mathcal{A}_i$ , the set of vertices that belongs to the  $SourceTree_i$  branch which contains  $P_i$ . Otherwise, node  $v$  is in  $\mathcal{A}_i$  if the shortest path from  $s$  to  $v$  in  $\mathcal{G}_i$  shares at least one arc with  $P_i$ . Let  $\mathcal{B}_i = \mathcal{V} \setminus \mathcal{A}_i$  (see figure 1). We can easily prove that for every vertex  $x$  in  $\mathcal{B}_i$ , the shortest path from  $s$  to  $x$  deduced from  $SourceTree_i$  can still be used at step  $i + 1$ .

*Proof* : For  $v \in \mathcal{V}$  let us call  $P_i^{s \rightarrow v}$  the shortest path from  $s$  to  $v$  in  $\mathcal{G}_i$ .  $P_i^{s \rightarrow v}$  is thus the shortest path from  $s$  to  $v$  provided by  $SourceTree_i$ . We can notice that if  $v \in \mathcal{B}_i$  then all vertices of  $P_i^{s \rightarrow v}$  belong to  $\mathcal{B}_i$ . Let us suppose that for some  $x \in \mathcal{B}_i$ ,  $P_i^{s \rightarrow x} \neq P_{i+1}^{s \rightarrow x}$ . Given an arc  $e$ , its cost increases between step  $i$  and step  $i + 1$  if and only if  $head(e)$  is in  $P_i$ . Thus, all the more, only if  $head(e)$  is in  $\mathcal{A}_i$ . As every vertex of  $P_i^{s \rightarrow x}$  is in  $\mathcal{B}_i$ , the costs of arcs of  $P_i^{s \rightarrow x}$  remain the same in  $\mathcal{G}_{i+1}$ :  $c_{i+1}(P_i^{s \rightarrow x}) = c_i(P_i^{s \rightarrow x})$ . The fact path  $P_{i+1}^{s \rightarrow x}$  has been selected at step  $i + 1$  implies that its new cost is either smaller or equal to the constant cost of  $P_i^{s \rightarrow x}$ :  $c_{i+1}(P_{i+1}^{s \rightarrow x}) \leq c_{i+1}(P_i^{s \rightarrow x})$  and thus smaller or equal to its own previous cost:  $c_{i+1}(P_{i+1}^{s \rightarrow x}) \leq c_i(P_i^{s \rightarrow x})$ . As those transformations never reduce the costs of arcs, the cost of a path cannot decrease. As a consequence we can ensure that:  $c_i(P_{i+1}^{s \rightarrow x}) \leq c_{i+1}(P_{i+1}^{s \rightarrow x})$ . That proves that the cost of  $P_{i+1}^{s \rightarrow x}$  is also constant and equal to the one of  $R_i^{s \rightarrow x}$ . This cost is still minimal at step  $i + 1$  and, as a consequence, we need not compute any new path  $P_{i+1}^{s \rightarrow x}$ .

Moreover, as neighborhood information is received periodically, we may also adapt selected routes. Let us suppose that a last graph  $\mathcal{G}_{k+1}$  has been generated after the path  $P_k$  computation. Receiving neighborhood information, we can deduce if one or more previously chosen paths have disappeared or have become unreliable. In this case, we can modify  $\mathcal{G}_{k+1}$  by giving back to all concerned arcs (those who points to these arcs) their previous costs and execute as many steps of the algorithm as we have paths to replace.

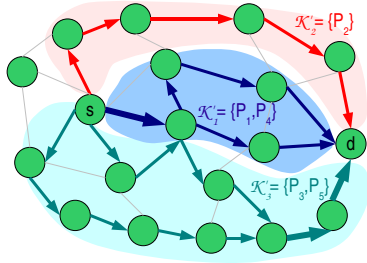


Fig. 2. Example of three subsets  $\mathcal{K}'_1$ ,  $\mathcal{K}'_2$  and  $\mathcal{K}'_3$

### F. OLSR adaptation to multi-path context

The multi-path computation algorithm in mobile ad hoc network requires modification of the data circulation in comparison with the classical OLSR protocol, in particular concerning the use of routes.

In OLSR protocol, each node maintains a routing table that contains the appropriate next node to use in order to reach any given destination. Our approach is a source routing strategy. This means that normally, every description packet must contain the path to follow as it has been defined by the source. Then intermediate nodes should not be allowed to disobey it. As a matter of fact, if they do so, the source may not be able to control the path dispersion anymore.

## V. SIMULATIONS

### A. Performance criteria: Reliability

Let us consider a set of  $N$  paths  $\mathcal{K} = (P_1, P_2, \dots, P_N)$  from  $s$  to  $t$ . For every edge  $e$  we can define the random variable  $X_e$  as being equal to 0 if the edge fails and to 1 if it is valid during the period  $T$  between two consecutive updates for  $e$ . We consider that edges are independent. Similarly we can define a random variable  $Y_i$  for route  $P_i$ . We have  $Y_i = \prod_{e \in P_i} X_e$ . The random variable  $Z$  that provides the number of available routes is equal to  $Z = \sum_i Y_i = \sum_i \prod_{e \in P_i} X_e$ . Supposing that  $N$  descriptions ( $D_1, \dots, D_N$ ) have been generated from  $I$  (the information to send), we use path  $P_i$  to carry  $D_i$ . The ability to reconstruct  $I$  at the destination is equal to the probability of receiving enough descriptions. Of course this depends on the parameter  $M$  used for multiple description coding.

For given values of  $M$  and  $N$ , we can define the reliability  $\rho$  of  $\mathcal{K} = (P_1, P_2, \dots, P_N)$  as the probability that  $Z \geq M$ . The higher is  $\rho$ , the most probably the destination may obtain the information. The calculation of  $\rho$  requires to keep in mind that paths are not necessarily disjoint.

We can define sets  $\mathcal{K}'_1, \mathcal{K}'_2, \dots, \mathcal{K}'_g$  following the definition: each path of  $\mathcal{K}$  must belong to exactly one set  $\mathcal{K}'_i$ ; two paths that share at least one arc must be in the same set  $\mathcal{K}'_i$ ; a set  $\mathcal{K}'_i$  cannot be divided in two subsets with no common arcs. Figure 2 illustrates the distribution of 5 paths among 3 subsets.

We can then define the random variable  $Z_j$  as the number of available routes in  $\mathcal{K}'_j$ . Then, as  $(\mathcal{K}'_j)_j$  are edge-disjoint,  $(Z_j)_j$  are independent:  $Pr(Z \geq M) = \sum_{m_1 + \dots + m_g \geq M} \prod_{j=1}^g Pr(Z_j = m_j)$ . Calculating  $Pr(Z_j =$

	$e \text{ in } P_1$ $\pi_1 = 0.74$	$e \text{ in } P_3$ $\pi_3 = 0.81$	$e \text{ in } P_1 \text{ and } P_2$ $\pi_{12} = 0.73$	$e \text{ in } P_2 \text{ and } P_3$ $\pi_{23} = 0.90$	
$Pr(Y_1=0 \& Y_2=0 \& Y_3=0)$	0	0	0	0.05	0.09
$Pr(Y_1=0 \& Y_2=0 \& Y_3=1)$	0	0	0	0.22	0.20
$Pr(Y_1=0 \& Y_2=1 \& Y_3=0)$	0	0	0.05	0.04	0.04
$Pr(Y_1=0 \& Y_2=1 \& Y_3=1)$	0	0.26	0.21	0.15	0.13
$Pr(Y_1=1 \& Y_2=0 \& Y_3=0)$	0	0	0	0	0.05
$Pr(Y_1=1 \& Y_2=0 \& Y_3=1)$	0	0	0	0	0
$Pr(Y_1=1 \& Y_2=1 \& Y_3=0)$	0	0	0.14	0.10	0.09
$Pr(Y_1=1 \& Y_2=1 \& Y_3=1)$	1	0.74	0.60	0.44	0.40

$\rho \rightarrow \rho'$  represents  $\rho' = \pi_e \cdot \rho$   
 $\rho, \rho_1, \rho_2 \rightarrow \rho'$  represents  $\rho' = \rho + (1 - \pi_e) \cdot (\rho_1 + \dots + \rho_i)$

Fig. 3. Example of the beginning of the computation of  $Pr(Y_1 = y_1 \& Y_2 = y_2 \& Y_3 = y_3)$  for a subset  $\mathcal{K}' = \{P_1, P_2, P_3\}$

$m)$  requires to construct paths by considering successively every edge  $e$  of  $\mathcal{K}'_j = \{P_{j1}, \dots, P_{jn_j}\}$  and to consider the values of  $Pr(Y_{j1} = y_1 \& Y_{j2} = y_2 \& \dots \& Y_{jn_j} = y_{n_j})$  with  $y \in \{0, 1\}$ . At each step the probabilities of the different cases are modified. If  $\mathcal{K}'_j$  contains  $n_j$  paths, there are  $2^{n_j}$  cases to process at each step (corresponding to all possible values of  $(y_1, \dots, y_{n_j})$ ).

Figure 3 shows the 4 first steps of the procedure used for a subset  $\mathcal{K}' = \{P_1, P_2, P_3\}$ .

In our approach, vertices are considered entirely trustworthy. This hypothesis might be deemed as unrealistic but we consider that every vertex failure can be taken into account as the failure of all its edges.

### B. Model description

Our simulations require to define ad hoc network topologies. This is done by spreading a given number of nodes in a square area. All nodes have the same range. A source and a destination are designated. Each link  $e$  is given a success probability  $\pi_e$  by randomly selecting a value in a given subinterval of  $[0, 1]$ . Furthermore, we take into consideration the limited memory available at each intermediate node for messages. This implies a dependency between nodes and their received bitrates. As our calculation is based on links reliabilities, we multiply the success probability of every link  $e$  by the coefficient  $1 - \exp(-\Lambda/\lambda_v)$  where  $\Lambda$  is a constant and  $\lambda_v$  the bitrate received at node  $v$ , the head of  $e$ . As packets are transformed before being sent, the actual bitrate on a given path is smaller than the bitrate of transfer  $\lambda$ . We consider optimal multiple description coding, thus on every path  $P$  the bitrate is  $\lambda_P = \lambda/M$ . If  $U_v$  is the number of paths using node  $v$ , the bitrate on  $v$  is  $\lambda_v = \lambda U_v/M$ . Table I provides the different parameters considered in the simulated scenarios.

### C. Results analysis

Figures 4, 5 and 6 represent the values of reliability  $\rho$  function of the parameter  $M$ . Colors refer to  $N$ , the total number of used routes (green for 2 routes, cyan for 3 routes, blue for 4 routes, purple for 5 routes and red for 6 routes). Different curves of a same color correspond to different kinds

Range of nodes	200 m
Area dimension	1000 m × 1000 m
$\Lambda$	60 Mb/s
Number of nodes	30, 60, 120, 180 or 240
Edge success probability	[0.9, 1], [0.7, 1] or [0.5, 1]
Transfer rate $\lambda$	3 Mb/s, 6 Mb/s, 10 Mb/s, 15 Mb/s or 30 Mb/s
$f_p$	$f_p(c) = c + c_0$ $c_0 \in \{0.1, 0.3, 0.7, 1, 2, 5\}$
$f_e$	$id \leq f_e \leq f_p$
Number of routes $N$	2, 3, 4, 5 or 6
Number of routes necessary for reconstruction $M$	$1 \leq M \leq N$

TABLE I  
PARAMETERS USED IN THE SIMULATED SCENARIOS

of incremental functions. The dotted horizontal line indicates the reliability of the best path used alone, which corresponds to the case where  $N = M = 1$ .

A general observation is that the choice of the incremental function is not so much relevant in comparison with the choice of other parameters. We remind that the cases where  $M = 1$  and  $N > 1$  correspond to the creation of  $N$  copies of the original data and the cases where  $M = N$  correspond to the division of the original data in  $N$  non-redundant parts. This strategy does not seem relevant because it requires all routes to be valid together. This surmise is confirmed by all the figures: the reliability becomes very low when  $M$  tends to  $N$ .

1) *Influence of rate*: In figure 4, it is noticeable that in case of a low bitrate, we obtain very good values for low values of  $M$ . This can be explained by the fact that even if copies are created, as they are small, the transfer efficiency is not really degraded. As the rate grows, cases with low  $M$  (including the single path method) become less interesting while cases with intermediate values of  $M$  still provide high reliability. Moreover, the more paths are used, the better is  $\rho$ .

2) *Influence of network density*: As might be expected, the denser is the network, the higher is  $\rho$ . However, depending on the rate, the global behaviour is not the same when the density changes. With small rates the density does not imply noteworthy changes in the curves relative positions. On the contrary, high rates (see figure 5) imply that using a single route is the best strategy in sparse networks and that using the maximum number of routes is preferable in dense networks. In fact, when the rate is high, the dispatching of data among several routes avoid local congestions. In a sparse network this may not be possible given that few disjoint paths are available. Consequently, focusing on the best path of a sparse network is a better strategy than using multiple but very similar paths.

3) *Influence of link stability*: Figure 6 shows that in dense networks the impact of the link stability is not very relevant. Indeed, even if some links might be bad, there exist enough different paths to select the best ones. On the contrary, in sparse networks all methods fail when the links stability falls. As a general tendency we can notice that when links become unstable, using copies becomes the best approach.

## VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we have proposed a multi-path routing algorithm for mobile ad hoc networks (MP-OLSR). In particular, we have proposed a new algorithm for multiple route computation based on the principle that all the routes are to be used simultaneously. Each packet is transformed into  $N$  subpackets called descriptions, and distributed on  $N$  selected paths. Only  $M$  of the  $N$  descriptions are necessary to reconstruct the original packet. This kind of transformation, called multiple description coding, ensures at the same time a better load balancing and an increase of reliability of transmissions.

Simulation results show an improvement of performance, in particular when the data rate is high. Also, the denser the wireless network is, the more sensitive the improvement of reliability is. The MP-OLSR can satisfy the needs for QoS of certain applications. Some outlooks for evolutions are considered:

- it may be interesting to refine the choice of incremental functions  $f_e$  and  $f_p$ , used at each step of our route computation algorithm;
- in order to compute the paths reliability, the links probabilities are chosen in an independent way: no interdependence is considered. When a link fails, some others may also become unreliable. Consequently, the success probability model can be improved by considering this dependency;
- we are implementing MP-OLSR on NS2 simulator in order to evaluate the improvement of performances, and in particular the packet loss rate and the end-to-end delay when the topology changes.

## REFERENCES

- [1] J.A.De Azevedo, J.J.Madeira, E.Q.V.Martins, F.M.A.Pires, "A Shortest paths ranking algorithm", proceeding AIRO, 1990.
- [2] T.Clausen, P.Jacquet, "Optimized Link State Routing Protocol", IETF RFC 3626, October 2003.
- [3] S.J.Lee, M.Gerla, "Split Multi-Path Routing with maximally disjoint Paths in ad hoc Networks", International Conference on Communication, Helsinki, June 2001.
- [4] R.Leung, J.Liu, E.Poon, A.Cahan, B.Li, "MP-DSR: AQoS-aware Multi-path Dynamic Source Routing Protocol For wireless ad hoc Networks", Proceedings of the 26 Annual Conference on local Computer Networks, 2001, pp. 132-141.
- [5] A.Nasipuri, S.Das, "On Demand Multi-path Routing for Mobile ad hoc Networks", Proceeding of 8th Annual IEEE international Conference on Computer Communications and Networks (ICCCN)", Boston, MA, October 1999.
- [6] Srinivas, E.Modiano, "Minimum energy disjoint path in wireless ad hoc networks", proceeding of 9th Annual International Conference on Mobile Computing and Networking, pp.122-133, September 2003.
- [7] V. Goyal, "Multiple Description Coding: Compression Meets the Network", IEEE Signal Processing Magazine, vol. 18, pp. 74-93, September 2001.
- [8] J.W.Suurballe, "Disjoint Paths in a Networks", Network 4, pp.125-145, 1974.
- [9] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing", Mobicom 2003: Proceedings of the 9th annual international conference on Mobile computing and networking, pp. 134-146, 2003.

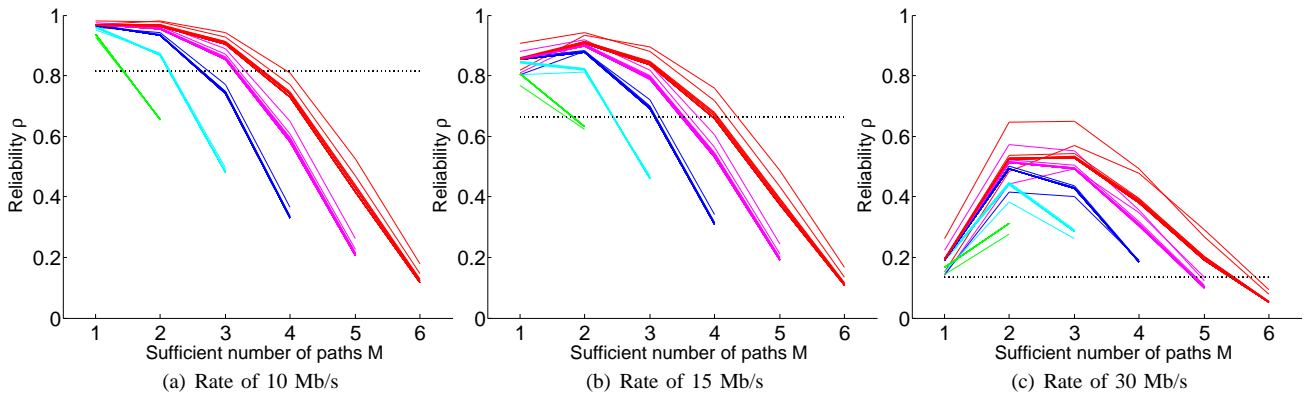
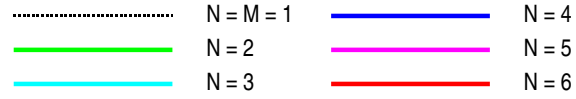


Fig. 4. Scenarios with 240 nodes, edge success probability in  $[0.7, 1]$  and variable rates

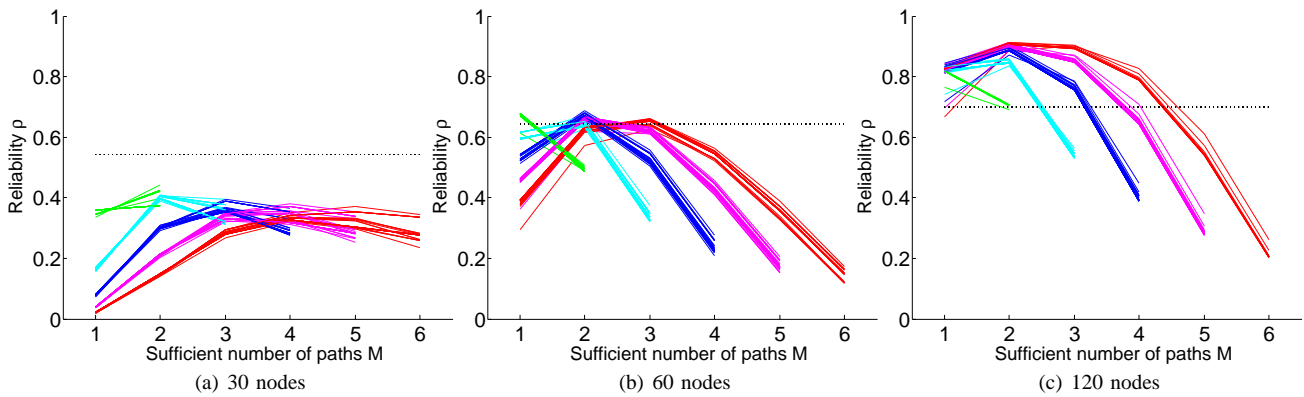


Fig. 5. Scenarios with variable node number, edge success probability in  $[0.7, 1]$  and a rate of 15 Mb/s

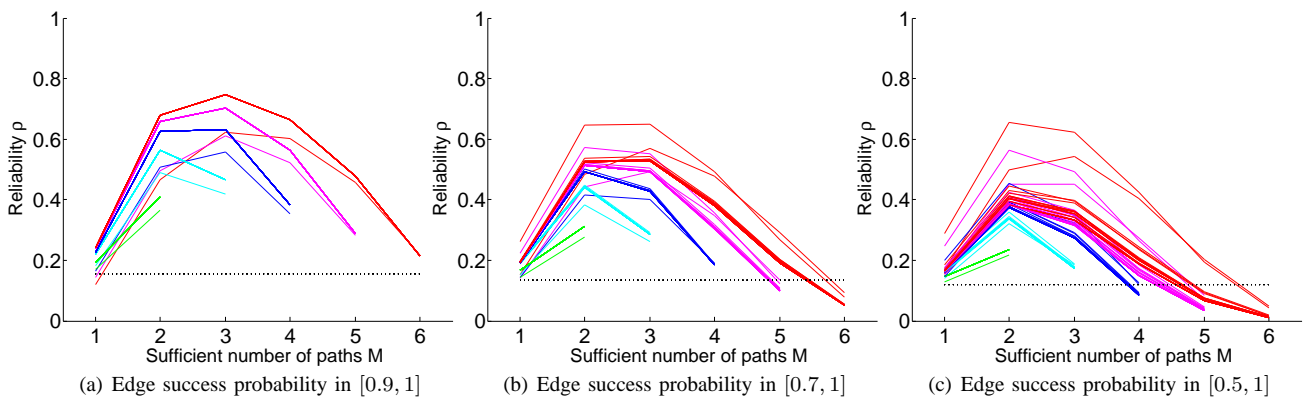


Fig. 6. Scenarios with 240 nodes, various type of edge success probability and a rate of 30 Mb/s