Performance Analysis of XOR-based Flat Routing Protocols in High Mobility Vehicular *ad hoc* Networks

Rodolfo Oliveira*†, André Garrido*, Miguel Luís†, Rafael Pasquini§, Luis Bernardo*†, Rui Dinis*‡
Paulo Pinto*†,
*Departamento de Engenharia Electrotécnica, Faculdade de Ciências e Tecnologia, FCT,
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal †UNINOVA, 2829-516 Caparica, Portugal ‡
IT, Instituto de Telecomunicações, Portugal
§ State University of Campinas 13083-970 - Campinas - SP - Brazil

Abstract

The routing strategy is an important issue in Vehicular *ad hoc* Networks since the shared nature of the wireless medium, the time-varying capacity of the links and the highly dynamic network topology due to vehicle's mobility severely restrict the choices available for the creation of the paths. These issues influence the availability of the path and its long-term duration.

In this report we present a performance analysis of XOR[1]-based flat routing protocols in high mobility conditions, considering a vehicular *ad hoc* network (VANET) formed in a highway scenario. First, we propose an XOR-based protocol that incorporates several adaptations of the existing XOR-based routing algorithms, in order to cope with the network mobility. Then we propose an improved version of it, XORi, which modifies the protocol's information gathering process to accommodate the specific dynamic nature of VANETs topology. Finally, we evaluate the performance of XOR-based protocols with other topology-based routing protocols. Simulation results allow us to characterize the performance of this class of protocols through the comparison of the packet delivery ratio, end-to-end path delay and average number of path hops[2]. When a moderate density of nodes is considered, simulations show that XOR-based algorithms achieve almost the same packet delivery rate as link state algorithms, such as OLSR, while for high density of nodes XOR-based algorithms scale better in terms of delay when compared to source routing algorithms, such as DSR.

Keywords: XOR-based Routing Protocols, Flat Routing Protocols, Vehicular *ad hoc* Networks.

[1] Exclusive-or logical operator.

[2]The source code of the simulated XOR-based protocols was written for the network simulator ns-2.34 and is available to download at `http://tele1.dee.fct.unl.pt/people/rado/html/downloads .html`, allowing the community to evaluate their own scenarios and compare it with other protocols.

I. INTRODUCTION

Vehicular *ad hoc* Networks (VANETs) are rapidly becoming a reality since several organizations are supporting standardization activities that will enable a variety of applications such as safety, traffic efficiency and infotainment. VANETs are self-organized networks where the level of node's mobility is generally higher and the mobility is constrained by the roads. Due to the fast change of the topology, VANETs demand for routing protocols able to provide high packet delivery rate and low end-to-end packet delivery delay.

Generally the routing protocols that have been proposed for VANETs can be classified into three basic groups [1]: unicast position-based, unicast topology-based or group-based multicast and broadcast.

In unicast position-based routing protocols [2], the nodes do not needed to store any route or routing table to the destination. Instead, the nodes use the location of their neighbors and the location of the destination node to determine the neighbor that forwards the packet. Therefore, these routing schemes require information about the position of the nodes, which is a drawback because of the cost of disseminating this information across all VANET nodes.

In topology-based protocols the nodes need to store routing tables or routes that depend on the topology. This class of protocols include the well known Ad hoc On-Demand Distance Vector Routing (AODV) [3], Optimized Link State Routing (OLSR) [4] and others (DSDV [5], DSR [6], TORA [7], FSR [8]). These protocols pose great challenges in VANETs, since the mobility of the nodes causes frequent topology changes: usually they continuously maintain up-to-date routes for valid destinations and require periodic updates to reflect network topology changes. This requirement can lead to high bandwidth consumption, which can be alleviated by some optimization processes, such as the MultiPoint Relay (MPR) scheme used in [4].

Group-based multicast and broadcast protocols are designed to route packets from one node to multiple destinations. Since this work considers unicast routing (packets destined to a given node), we do not introduce this class of protocols in this report.

XOR-based flat routing algorithms, such as the ones described in [9], [10] and [11] have been proposed for wired networks. These protocols do not rely neither on the network topology nor on the location of the nodes. Instead, they use a metric based on the logical XOR operation between the identifiers of the network nodes, which is solely used to route the packets. This class of routing protocols is completely decoupled from the actual topology and state of the network: the routing mechanism is "blinded" in the sense that it only uses the information related with the identifiers of the nodes, independently of any other metric, as is the case with topology-based or position-based protocols. XOR-based protocols are proposed to solve the scalability problems usually faced in both topology or position-based protocols, such as:

- most of these protocols store information about active routes or about every node addressable in the network, limiting the scalability of the protocols when the number of network nodes increase;
- most of these protocols may have to broadcast queries through most of the network before the desired route is found, which limits their scalability and introduces large amounts of routing traffic overhead.

In this work we evaluate the performance of XOR-based routing protocols considering a high mobility application scenario such as highways, to provide the deployment of comfort applications (e.g. onboard games and video/music file sharing). First, we propose an XOR-based protocol that incorporates several adaptations of the existing protocols in order to handle the node's mobility. Then we propose a second improvement, which modifies the protocol's information gathering process to accommodate the dynamic nature of VANET's topology. Finally, we evaluate the performance of XOR-based protocols with other topology-based routing protocols, characterizing their performance through the comparison of the packet delivery ratio, end-to-end path delay and average number of path hops. As far as we know this is the first work that analyzes the performance of XOR-based routing protocols in a VANET scenario.

The rest of the report is organized as follows. Section II describes the traditional XOR-based routing architecture and some of the adaptations need to handle node's mobility. In Section III we propose an improvement of the traditional XOR-based protocols, which modifies the protocol's information gathering process. Section IV presents and discusses the experimental results. Finally, some concluding remarks are given in section V.

II. XOR-based Routing: system description

XOR-based routing algorithms use $n-$bit identifiers of the network nodes, which can represent their addresses at the network layer, to build the routing tables and to route packets through the network. Its routing principle uses the distance between two identifiers a and b as their bitwise exclusive or (XOR), which is represented by $d(a,b) = a \oplus b$, being $d(a,a) = 0$ and $d(a,b) > 0, \forall a,b$. Given a packet originated by the node owning the identifier x and destined to the node with the identifier z, and denoting $\mathbb{Y}$ as the set of identifiers contained on x's routing table, the XOR routing algorithm applied at node x selects the node $y \in \mathbb{Y}$ that minimizes the distance to z, which is expressed by the following routing policy

$$\mathcal{R} = \underset{y \in \mathbb{Y}}{\operatorname{argmin}} \left\{ d(y,z) \right\}. \tag{1}$$

The routing policy $\mathcal{R}$ routes each packet to a node y that guarantees the minimum distance to the destination z. In the rest of the report we refer to node x as the node which owns the identifier x.

Each node maintains a table, previously referred as routing table, where the available knowledge about its neighbor nodes is stored. The table, exemplified in Table I, is organized in n columns denominated buckets and represented by $\beta_\alpha, 0 \leq \alpha \leq n-1$. Each time a node a knows a novel neighbor b it stores that information in the bucket β_{n-1-i} given by the highest i that satisfies the following condition[3]

$$d(a,b) \text{ div } 2^i = 1, a \neq b, 0 \leq i \leq n-1. \tag{2}$$

If we use 4-bit identifiers ($n = 4$) and admitting $a = 1001$ and $b = 1010$, the distance $d(a,b) = 0011$ and the highest i that satisfies the condition (2) is $i = 1$, concluding that the identifier $b = 1010$ should be stored in the bucket $\beta_{n-1-i} = \beta_2$. In other words, condition (2) denotes that node a stores the identifier of node b in the bucket

[3]div denotes the integer division operation on integers.

β_{n-1-i}, where $n-1-i$ is the length of the longest common prefix between the identifier of node a and node b. This can be observed in Table I, where the buckets $\beta_0, \beta_1, ..., \beta_3$ store the identifiers having a common prefix of length $0, 1, ..., 3$ with node 0001. The way how the information about the identifiers is stored in the buckets is one of the advantages of XOR-based algorithms. This is because a node only has to know about n of the possible 2^n nodes available in the network to successfully route a packet.

TABLE I

HYPOTHETIC ROUTING TABLE OF THE NODE 0001 USING 4-BIT IDENTIFIERS.

β_0	β_1	β_2	β_3
1000	0100	0010	0000
1010	0101		
1100			

A node a can know b from two distinct ways: a **discovering process** in which the nodes actively search for neighbors to fill the buckets; a **learning process** where a node uses the packets received to add more information to the buckets in a costless passive fashion.

"XOR" algorithm will be used throughout this report to designate the algorithm that incorporates the concepts presented in this section.

A. Discovering Process

In the discovering process a node always knows its **physical neighbors**, since a `HELLO` message is broadcasted by each node at frequency f_H Hz. Each physical neighbor node discovered is then stored in the bucket having the greatest i that solves the condition (2).

While a node stills having one of the buckets β_α empty, it actively searches for neighbors that can fill the bucket. Since such neighbors are not physically connected, they are denominated **virtual neighbors**. A node starts to send a `QUERY` message describing the empty buckets, which is sent to its neighbors already stored in the buckets (in the initial step the buckets only contain physical neighbors). The neighbor nodes that receive the `QUERY` and have at least a neighbor that fills in one of the requested buckets, answers with a `RESPONSE` message. After receiving the `RESPONSE` a node stores the discovered virtual neighbors in its buckets. New queries are sent to the discovered neighbors if at least one of the buckets is still empty.

The queries are limited in range to K_h hops far away from the query's originating node. The neighbors stored in the buckets are deleted after T_β seconds due to the high degree of mobility of the nodes. When the queries sent to all neighbors do not originate any response, a node repeats the `QUERY` sending process after T_Q seconds.

B. Routing and Forwarding

Since a node can route a packet to a neighbor node that can be $k-$hops away, we distinguish the notion of routing and forwarding.

Each time a node receives a packet not destined to him, it tries to route it by applying the routing policy $\mathcal{R}$. The routing process starts by identifying the bucket that should be used. Considering that node a receives a packet destined to b, it defines the bucket β_α to use by applying the condition (2). Then node a routes the packet to the node that is closest to node b. In other words, the packet is routed to node n_R, which is selected by node a computing the following solution

$$n_R = \underset{id \in \beta_\alpha}{\operatorname{argmin}} \{d(id, b)\}, \tag{3}$$

where id represents each one of the identifiers contained in the bucket β_α.

Since n_R is the identifier of a node that can be a virtual neighbor $k-$hops away ($k > 1$), node a must know how to forward the packet to the node n_R. This example is illustrated in Figure 1. Since node a must route the packet to node n_R, it forwards the packet to its physical neighbor g. Node a knows that it must forwards the packet to node g because when it receives the `RESPONSE` from node n_R, node a stores the node n_R associated with the physical neighbor from which it was received (node g). Thus each identifier contained in a bucket has an associated physical neighbor (node g), which is used to forward the packet to the node chosen by the routing algorithm (node n_R).

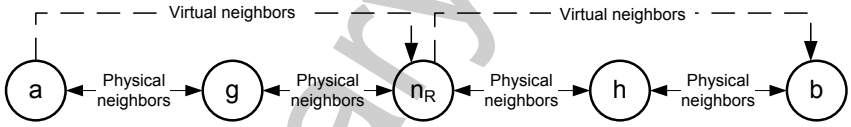

Fig. 1. Routing/forwarding example.

When the packet arrives at g, it performs an XOR operation to define the bucket in which the next hop must be taken and, as a consequence of the routing tables building process, node g is able to find node n_R in the defined bucket. In this case, the flat routing decision is identical to the physical forwarding action (to send the packet to node n_R). Once node n_R receives the packet, it is the node pointed by the previous routing decision as progress in the flat identity space, i.e., the node in which the packet can walk, at least, one bit towards the destination. In this case, node n_R defines the bucket and finds the destination node b in its routing table towards the interface connected to node h. Consequently, all the possible progress in the flat identity space towards the destination node is done and the remaining action is to forward the packet to the physical neighbor node h that, in its turn, will deliver the packet to node b.

C. Learning Process

Figure 2 motivates the learning process, where node 1 sends a QUERY to node 2, and node 2 sends back a RESPONSE informing node 1 about the existence of node 3. In the same way, node 1 queries the nodes 3, 4 and 5 until filling all its buckets. The learning process takes place when nodes 3, 4 and 5 receive the QUERY from node 1 and from that instant they know node 1 in a passive way, learning about its existence. Following the same rationale, when node 5 sends back a RESPONSE to node 1, both nodes 3, 2, and 1 learn about its existence. Thus the learning process takes place when a node receives a QUERY or a RESPONSE message. The node receiving the message verifies if the identifier of the source node is already stored in its table. If is not the case, the identifier of the source node can be stored in a bucket if the information does not originates the count-to-infinity problem, avoiding loops formation in the packet forwarding stage.

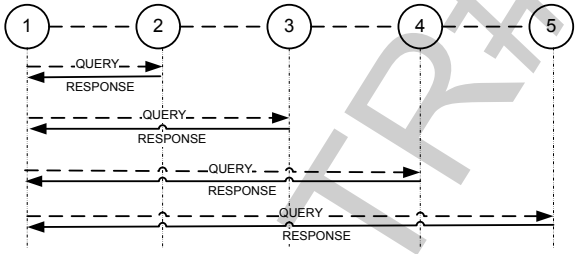

Fig. 2. Motivation example of the learning process.

The use of time-limited neighbor entries in the buckets can originate a count-to-infinity. Suppose that node 3 learns that node 1 exists by receiving its QUERY forwarded by node 2. Node 3 deletes the identifier 1 from its bucket after T_β seconds. If the bucket does not contain more identifiers, the node will send new QUERY messages to all its neighbors, since it needs to fill the empty bucket. Now suppose that node 4 also knows about node 1, since node 1 have previously sent QUERY messages to it and to node 5. When node 3 queries its neighbors, node 4 will answer that it knows node 1, but the path to reach node 1 also includes node 3.

The queries and the responses originated by each nodes are univocally identified by the source node, destination node and a given incremental identifier. Each node updates its cache with the univocal identifier of the QUERY or RESPONSE from which it learned. All the identifiers of received queries or responses are cached as well. The count-to-infinity problem is solved by verifying if the information requested by a node was previously seen by the same node. In the former example node 4 informs node 3 that it knows node 1, but the RESPONSE includes the univocal identifiers of the QUERY from which node 4 learned about node 1. By receiving that information node 3 also verifies that it caches the same QUERY univocal identifiers, so it rejects this information and does not stores node 1 in the bucket, avoiding the count-to-infinity problem.

III. XOR-based Routing: improvements for VANETs

This section presents XORi algorithm, which is an improvement of XOR algorithm presented in the last section. XORi is more suited for high mobile networks, such as VANETs.

A. Main Improvements

As described in the last section, in the beginning of the routing table building process a node starts to query all its neighbors about the required information to fill its buckets. This is not a good practice in wireless networks, since it decreases the network capacity. XORi maintains the same rationale as XOR algorithm, except it proposes the following changes:

- instead of starting to query all neighbors, in XORi a node only interrogates a single node (denominated Broadcast Group Leader - BGL) selected by the node that originates the `QUERY`.
- a node only interrogates all its neighbors when it has not previously selected a BGL node.

While this practice decreases the network load, if BGL nodes are properly selected according to the network motion, they can form a set of nodes that contains more information. This is because BGL nodes are interrogated more frequently than any other nodes and, since the learning process is used, the probability of storing more information in their buckets increase.

As BGL nodes will own more information in its buckets, and since the nodes are mobile, they should be selected BGLs for the longest time in order to increase the performance of its usage. This is explained by the fact that if selected BGL nodes are often changing, they are interrogated less times, and they will have less information. Since the mobility environment considered in this work is a highway, a BGL node should be the neighbor node which maintains the longest link longevity. In the next subsection we describe the algorithm used to select BGL nodes.

B. Broadcast Leader Selection Algorithm

This algorithm uses the `HELLO` messages previously described, and is computed every time a new `HELLO` message is received from a neighbor node. Each node selects a single BGL.

In the time instant $t_i(n_y)$, when the node n_a firstly receives an `HELLO` packet from it's neighbor node n_y, an unidirectional logical link is created between the nodes. The link is maintained since n_y periodically transmits beacons with period $T_B = 1/F_H$. The duration of the logical link can be quantified by its **stability** value: the stability $\eta(n_y)$ measures the duration of the logical link between the nodes n_a and n_y in periods of beacons. $\eta(n_y)$ is computed by the node n_a at instant t by applying the following expression

$$\eta(n_y) = 1 + (t - t_i(n_y)) \text{ div } T_B. \tag{4}$$

A logical link is said to be stable if it last longer than a given k_{est} value: $\eta(n_y) \geq k_{est}$. The neighbors with which a node maintains stable links (stable neighbors) are more suitable to be BGL nodes, because these nodes sense less link breaks. We denote by $\xi(n_a)$ the BGL node selected by the node n_a.

Representing the physical neighbors of n_a by N_a and admitting that n_a knows the BGL nodes selected by its physical neighbors ($\xi(n_y), \forall n_y \in N_a$ - which is contained in each Hello packet received from each $n_y \in N_a$ and stored in the beacon table jointly with $\eta(n_y)$), n_a selects its own BGL by applying the Algorithm 1.

```
Input  : N_a, η(n_y) (∀n_y ∈ N_a), ξ(n_y) (∀n_y ∈ N_a), t_i(n_y) (∀n_y ∈ N_a)
Output : ξ(n_a)
1  η_max ⇐ return_max_η_from_beacon_table()
2  address ⇐ MAX_INT
3  ξ_aux ⇐ -1
4  transient_threshold ⇐ 1
5  if stable_node(n_a) then                                   /* R1 - if this node is stable */
6      for each neighbor n_y ∈ N_a do
7          insert_sorted(ξ(n_y), list_BGL)                   /* lower addresses in the head of the list */
8      if (n_a is BGL) then
9          insert_sorted(n_a, list_BGL)
10     for each ξ_y ∈ list_BGL do                            /* ξ_y is removed from the head of the list */
11         for each neighbor n_y ∈ N_a do
12             if (n_y = ξ_y) and stable_node(n_y) then      /* R4 - select a neighbor that already is BGL */
13                 ξ_aux ⇐ n_y
14
15         if (ξ_aux ≠ −1) then                              /* BGL already selected */
16             break
17         if (n_a = ξ_y) then                               /* R3 - auto-selection */
18             ξ_aux ⇐ n_a
19             break
20
21     if (ξ_aux = -1) then                                  /* R2 - its neighbor becomes a new BGL */
22         for each neighbor n_y ∈ N_a do
23             if (η_max − η(n_y) − transient_threshold ≤ 0) and (n_y < address) then
24                 address ⇐ n_y
25                 ξ_aux ⇐ n_y
26
27
28
29 ξ(n_a) = ξ_aux
```

Algorithm 1: Algorithm used by the generic node n_a to select its Broadcast Group Leader $\xi(n_a)$.

The algorithm rules R1-R4 are summarized as follows:

R1 - when n_a is unstable (meaning that n_a does not have stable neighbors) it does not select any BGL;

R2 - when none of n_a's neighbors ($n_y \in N_a$) had previously selected a BGL, n_a selects the neighbor having the smaller address from the set of the neighbors with which n_a has the biggest stability value;

R3 - n_a selects itself as a BGL when n_a is already a BGL node (previously selected by a neighbor) and the neighbors's BGL have higher addresses than n_a;

R4 - when n_a is not selected BGL by its neighbors and there exists at least one neighbor n_y that is already a BGL, n_a selects the node n_y as its own BGL; ties are broken by choosing the smaller address neighbor;

The first BGL node selected in the network is justified by the application of the rule R2. The rule R4 is defined

to merge several BGLs selected by different nodes at 1-hop radius. The rule R3 is also used to merge several BGL in the special case when n_a must selects itself as a BGL.

Since the mobility scenario considered in this work is the same as considered in [12], we adopted the same parameterizations in this work ($k_{est} = 50$, $T_B = 1s$).

IV. Simulations and Performance Results

A. Mobility Model

The VANETs simulated in our evaluation scenarios were obtained using the Trans tool [13], which integrates the SUMO traffic simulator [14]. We have simulated a segment of a straight line highway with 3 lanes in each direction. The simulations start with the vehicles moving in both sides of the highway. During the simulation we launch more vehicles to maintain a constant density of nodes in the network. The highway segment is 10 kms long, which limits the minimum number of the network hops to cover the full highway segment to 10, as all vehicles are assumed to have a radio range of 1000 meters. We defined three different classes of vehicles, which are described in the table II. 60% of the vehicles belong to the class$_1$, which represents medium size cars. The vehicles of class$_2$ represents 25% of the highway traffic. Finally we define 15% of vehicles of class$_3$, which represents long sized vehicles such as trucks. Regarding the vehicle's density, denoted by ρ, we defined 4 different scenarios, described in the table III.

TABLE II
CLASSES OF TRAFFIC CONSIDERED IN THE SIMULATIONS.

	vehicle's length (m)	V_{MAX} (m/s)	acceleration (m/s^2)	deceleration (m/s^2)	%
class$_1$	4	27.8	3.6	3.6	60
class$_2$	5	26.0	2.5	3.0	25
class$_3$	8	20	1.5	2.0	15

TABLE III
VEHICLE'S DENSITIES CONSIDERED IN THE SIMULATIONS.

	number of vehicles	average number of neighbors (ρ)	simulation time (s)
Scen$_4$	80	4.0	747
Scen$_6$	120	6.0	727
Scen$_8$	160	8.0	772
Scen$_{10}$	200	10.0	805

B. Simulation Description

The simulations compare the performance of XOR algorithm (described in Section II) and XORi (presented in Section III) with OLSR, AODV and DSR routing protocols. We used the simulator ns-2 [15] configured with the standard IEEE 802.11[4].

The vehicles moving in the same left-to-right direction generate packets, which are randomly destined to one of the active mobile nodes. The vehicles moving from right-to-left do not generate packets but are able to forward them. The number of packets generated on each density scenario was maintained constant at approximately 3000 packets. The parameterizations used in XOR and XORi algorithms were: $f_H = 1$Hz; $K_h = 5$hops; $T_\beta = 5$s; $T_Q = 5$s; $T_B = 1$s. All simulation results presented were obtained within 95% of confidence interval.

C. Performance Results

Table IV shows the packet delivery ratio for all simulated protocols and for different node densities (ρ). As can be seen XOR and XORi exhibit approximately the same packet delivery ratio, which is very similar to the OLSR protocol. This allow us to conclude that the changes performed in XORi does not have a significant impact on the packet delivery ratio. AODV and DSR protocols present higher packet delivery ratios due to the on-demand nature of AODV and the pure flooding used in DSR.

TABLE IV
PACKET DELIVERY RATIO [%].

ρ	XOR	XORi	OLSR	AODV	DSR
4	43.0	42.0	44.5	65.8	97.7
6	45.3	48.6	46.8	67.9	97.0
8	55.6	52.4	47.5	69.0	97.4
10	49.2	50.3	45.6	71.0	96.9

Other important metric is the average end-to-end delay between the source and destination node. Table V presents end-to-end delay for the same simulations reported in Table IV. While DSR presents the highest delivery ratio, it is the worst protocol in terms of end-to-end delay. This is mainly because it floods the entire network to set up the path. Note that the highest delay value is achieved for the case when a nodes has in average 4 physical neighbors (low density), and this is due to overloaded links that are the unique choices available to route packets between partitions of nodes (lack of path diversity). DSR does not scale, because for higher node densities (e.g. 20 nodes, which is not presented in the tables) the delay increases rapidly while the packet delivery ratio is the smallest one for all simulated protocols. This fact explains why we did not present simulations for density values higher than 10 neighbors, since DSR becomes unstable. AODV performs better than DSR, and OLSR presents the lowest delay

[4]11 Mbps and 2 Mbps were used to transmit unicast and broadcast traffic, respectively.

from all simulated protocols. Note that XOR suffers from a similar problem found in DSR: when the number of physical neighbors increase the delay grows very fast. As can be observed, the use of BGLs in XORi decreases the network load, which in turn decreases the end-to-end delay due to the smallest number of generated queries.

TABLE V
PATH END-TO-END DELAY $[ms]$.

ρ	XOR	XORi	OLSR	AODV	DSR
4	32.6	5.0	2.7	5.8	180.8
6	20.7	5.0	3.2	8.3	103.8
8	108.4	9.3	3.7	9.3	80.0
10	215.4	18.9	4.3	10.8	107.5

Table VI indicates the average length of the paths measured in number of hops. This metric indirectly indicates the path duration, since the probability of path break increases with the number of path hops. While DSR and AODV are more susceptible to path breaks due to longer paths in terms of hops, XOR and XORi are less susceptible and its average path length is between DSR/AODV and OLSR.

TABLE VI
AVERAGE PATH LENGTH [HOPS].

ρ	XOR	XORi	OLSR	AODV	DSR
4	2.79	2.73	2.33	3.79	4.88
6	2.72	2.99	2.46	3.92	5.02
8	3.42	3.48	2.43	4.04	5.31
10	3.02	3.32	2.41	4.06	5.38

In a nutshell, the results presented in this section show that XOR and XORi have approximately the same performance as OLSR in terms of packet delivery ratio. The improvements presented in XORi marginally decrease its packet delivery ratio face to the XOR algorithm, namely for low density of nodes. But in terms of end-to-end delay the gain evidenced by XORi compensates the marginal loss in terms of packet delivery ratio.

V. CONCLUSIONS

This report presents a performance analysis of XOR-based flat routing protocols in high mobility conditions, considering a vehicular *ad hoc* network formed in a highway scenario.

Simulation results show that traditional XOR-based algorithms can have approximately the same performance as OLSR in terms of packet delivery. In terms of end-to-end delay, XOR-based routing exhibit better end-to-end delay for low density scenarios when compared to source routing algorithms and the improvements proposed in XORi algorithm can significatively decrease the end-to-end delay.

Further work will explore new features capable of decreasing the end-to-end delay and increasing the packet delivery ratio, such as the use of Bloom filters (used in [11]) and novel schemes to fill and maintain the information in the buckets.

REFERENCES

[1] J. Chennikara-Varghese, W. Chen, O. Altintas, and S. Cai. Survey of routing protocols for inter-vehicle communications. In *Mobile and Ubiquitous Systems: Networking and Services, 2006 Third Annual International Conference on*, pages 1–5, July 2006.

[2] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, Nov/Dec 2001.

[3] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90–100, 25-26 Feb 1999.

[4] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol. *IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-10.txt*, 2003.

[5] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[6] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[7] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1405–1413 vol.3, Apr 1997.

[8] G. P. Mario, M. Gerla, and T. Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *in Proceedings of ICC 2000*, pages 70–74, 2000.

[9] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. *Peer-To-Peer Systems: First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002*, 2002.

[10] B. Ford. Scalable internet routing on topology-independent node identities. Technical report, MIT, 2003.

[11] R. Pasquini, F. L. Verdi, M. F. Magalhes, and A. Welin. Bloom filters in a Landmark-based Flat Routing. *IEEE International Conference on Communications - ICC 2010. Cape Town, South Africa.*, 2010.

[12] R. Oliveira, L. Bernardo, M. Luis, and P. Pinto. Improving routing performance in high mobility and high density ad hoc vehicular networks. Technical Report TR#07_09, Universidade Nova de Lisboa, available at http://tele1.dee.fct.unl.pt/people/rado/downloads/draft_TR07_09.pdf, 2009.

[13] TraNS. - open source tool for realistic simulations of VANET applications . Software Package retrieved from http://trans.epfl.ch/, 2009.

[14] Centre for Applied Informatics (ZAIK) and Institute of Transport Research at the German Aerospace Centre. SUMO - Simulation of Urban Mobility. Software Package retrieved from http://sumo.sourceforge.net, 2009.

[15] Information Sciences Institute. NS-2 network simulator (version 2.31). Software Package retrieved from http://www.isi.edu/nsnam/ns/, 2007.