Spatial and Traffic-Aware Routing (STAR) for Vehicular Systems

Francesco Giudici, Elena Pagani
Information and Communication Department
Università degli Studi di Milano, Italy
e-mail: {fgiudici, pagani}@dico.unimi.it

***Abstract*— Vehicular ad hoc networks are studied with increasing interest for the many possible applications they have. Equipping vehicles with wireless devices primarily allows to design protocols and mechanisms to improve street safety. But also distributed applications for cooperative work, fleet management, passengers entertainment, could be supported. However, novel network protocols must be designed to make the deployment of vehicular networks possible.**

In this paper we consider the problem of data routing. We propose a novel position-based routing algorithm that is able to exploit both street topology information achieved from geographic information systems and information about spatial distribution of vehicles along street and vehicular traffic, in order to perform accurate routing decisions. The algorithm has been implemented in the NS-2 simulation environment and its performance has been compared with three other algorithms proposed in the literature.

Keywords: wireless systems; vehicular ad hoc networks; position-based routing; positioning and tracking technologies; geographic information systems; performance evaluation.

I. Introduction

Progresses in wireless technologies and decreasing costs of wireless devices are leading toward an increasing, pervasive, availability of these devices. Recently, research took interest in possibilities opened by equipping vehicles with wireless devices, and as a matter of fact cars with this equipment start to be available. Vehicles carrying on-board wireless devices are able to connect with one another in an ad hoc mobile network, and can also connect to Internet if access points to a fixed network are available within their communication range. These systems can be useful for several distributed applications, ranging from vehicular safety, to cooperative workgroup applications and fleet management, to service retrieval (e.g., availability of parking lots or petrol pumps in the neighborhoods), to entertainment for passengers (e.g., Internet browsing or distributed gaming).

Several problems are still to be solved in order to deploy vehicular communication networks. A main issue is to design effective and efficient routing algorithms appropriate for the characteristics of these systems, such as variable network topology due to node mobility. A promising approach seems to be the use of *position-based* routing, according to which a node is characterized by its position rather than a network address, and routing is performed basing on the current position of both the data source and destination. This approach is also of interest for particular applications in which the destination of a communication is determined by its position with respect to the source. This is for instance the case of a call addressed to the ambulance or the police squad that is *currently* nearest to the position of the caller. Some works have already been proposed in the literature. They deal with high node mobility in one of two ways: either no spatial consideration is taken into account and data routing is performed trying to exploit neighbors that are approximately in a good position to approach the destination of data, or spatial information is used in the attempt of forwarding data along streets – where vehicles can be and move – toward the destination. Both approaches can fail: with the former approach a neighbor can be chosen, which forces subsequent forwarding through zones where vehicles *cannot* be; the latter approach does not consider where vehicles *actually* are.

In this paper, the novel *Spatial and Traffic-Aware Routing* (STAR) algorithm is proposed, that overcomes the drawbacks mentioned above. It both exploits information about the road map in node surroundings, and discover further information about the local network status and vehicle distribution in space, through the exchange of *beacons* among neighbors.

The paper is organized as follows: in sec.II, we

This work has been partially supported by the Italian Ministry of Education, University and Research in the framework of the FIRB "Web-Minds" project.

describe the system model we adopt throughout the paper. Related works existing in the literature are briefly overviewed in sec.III, with particular emphasis on the algorithms we consider for performance comparison. In sec.IV, STAR algorithm is described in detail. In sec.V, the analysis is performed of the simulation results obtained with STAR and three algorithms existing in the literature. Sec.VI concludes the paper and highlights the future works we are carrying on.

II. System Model

In this work, we consider *Vehicular Ad Hoc Networks* (VANETs). As in Mobile Ad Hoc Networks (MANETs) [1], devices – in this case, vehicles – are equipped with wireless network interface cards that allow them to connect in a network infrastructure. Nodes are required to have unique identifiers; MAC addresses could be used to this purpose. The network topology dynamically changes as a consequence of vehicle movements, possibly at high speed. Each vehicle is responsible for forwarding data traffic generated or addressed to other vehicles, thus behaving as a router. The network is completely decentralized: vehicles have no information on either the network size – in terms of number of nodes involved – or topology. Two vehicles are said to be *neighbors* if they are in communication range. Differently from MANETs, in VANETs power saving is not of concern.

To route data packets, vehicles use information about the system. In particular, each vehicle can exploit a *Global Positioning System* (GPS) [2] to determine its own position. Moreover, vehicles are equipped with a GPS navigation system, which allows to obtain information about the local road map and the vehicle's direction of movement toward its destination. This assumption is in line with the current trend for default car equipment. For the use of STAR, digital road maps are translated into graphs, so that crossroads are represented by vertexes while streets are the edges of the graph.

The IEEE 802.11 technology [3] seems particularly well suitable for VANETs, as it provides a communication range large enough to guarantee good network connectivity also when vehicles move at high speed and maintain a relevant safety distance. Anyway, in this paper no assumption is done about the use of a specific wireless technology. According to all currently proposed wireless routing algorithms, vehicles periodically exchange network-layer *beacon* messages among neighbors, allowing each node to discover the identities and the positions of its own neighbors.

In VANETs, nodes are addressed through their position rather than through their network address. When a vehicle has data to send to another vehicle, it can discover the current position of the receiver by exploiting a *location service*. A location service is a distributed service that allows to maintain and discover the current position of vehicles moving around. Several location services have been proposed in the literature [4], [5], [6], [7], [8]. In this paper, we do not assume the existence of any particular location service, although the performance of the location service could affect STAR performance.

In some cases, a VANET may connect to a fixed network infrastructure via access points available along roadsides; yet, in this paper a *pure* wireless ad hoc network is considered. Different mobility scenarios are possible. Often, in the literature, a random-waypoint model is assumed. This model is not able to capture the peculiarities of vehicular movements. In real scenarios, vehicle movements are constrained by roads; "swarms" of vehicles can move in the same direction at a comparable speed, thus being motionless with respect to one another. On the other hand, a vehicle can abruptly change its direction in crossroads. In this article, a city or highway mobility model is assumed; in the former case, a Manhattan street topology is considered.

III. Related works

Routing algorithms for MANETs should be loop-free, to avoid wasting the scarce wireless bandwidth; they should operate on-demand to save node resources in terms of memory, computation and communication, and should take into account the possibility of nodes entering doze mode. The algorithms proposed in the literature can be divided into two classes: topology-based algorithms and *position-based algorithms*. In the former case, routing decisions are taken basing on the topology of wireless links existing among nodes. The algorithms can be proactive such as DSDV [9], reactive such as AODV [10] and DSR [11], or hybrid such as ZRP [12].

In this paper we focus on position-based algorithms. These algorithms do not require nodes to maintain routing tables. Basic assumptions of these algorithms are that a node knows its own position and a location service is available. These algorithms are thus particularly scalable and suitable for VANETs.

The GREEDY approach assumes that packets carry the receiver location, discovered by the sender by means of a location service. Each node routes a packet to the neighbor in its cell that is the nearest to the receiver. Several strategies can be adopted to select the next hop [13], [14], [15], [16]. In fig.1*(a)*, an example of GREEDY forwarding is shown, taken from [17]. Node x has a packet addressed to destination D. It discovers from beacons the positions of its neighbors; among them, y is the nearest to D and

is chosen as the next hop for the packet. The procedure is recursively applied by y and subsequent nodes till D is reached. The GREEDY approach does not perform well in case of sparse networks, where it is more likely that a packet is received by a node that has no neighbors nearer to the receiver than the node itself, i.e., the packet has reached a *local maximum*. A *recovery procedure* must be executed in this case, such as Face-2 [18] or GPSR [17]. As soon as it is possible to apply the greedy strategy again, the recovery procedure is abandoned. *Greedy Perimeter Stateless Routing* (GPSR) consists in building a *planar graph* using the links connecting a node to its neighbors; this operation can be executed locally by each node. A packet in a local maximum is forwarded on the face of the planar graph that is closer to the destination, according to a right-hand rule: the packet is sent along the link counterclockwise from the link on which it arrived. If a link should be used that intersects the straight line source/destination, and this intersection is closer to the destination than any other intersection previously encountered, then this link is abandoned and the next counterclockwise link is used instead. The algorithm guarantees that the packet reaches the destination if at least a path exists in the non-planar graph. Anyway, the recovery strategy requires that additional information is recorded in the packet header, thus being expensive, and can decrease performance if it is used often. Moreover, in some cases the GPSR strategy could be not applicable: in real settings the communication range among nodes also depends on surrounding environment, for instance on the presence of buildings that are obstacles to radio wave propagation. As a consequence, a node could be in a "shadow" zone where no neighbors are available.

Spatially-Aware Routing (SAR) [19] tries to overcome GPSR's problems by exploiting information on the road scenario in the surroundings, obtained from a *Geographic Information System* (GIS) [20]. The idea is that, as in real environments vehicles move along streets, paths along which packets are forwarded – vehicle by vehicle – must overlap with streets. GIS digital maps are translated into graphs; they are used by a node to choose the neighbor to which a packet should be forwarded, according to the positions of both the destination and the neighbors on the roads. In fig.1*(b)*, an example is shown, taken from [19]. Source S has a packet addressed to destination D. If a pure greedy strategy were applied, the packet should be forwarded to node A that is S's neighbor nearest to D – as shown in the left side of the figure. However, by considering the road map in the right side of the figure, one can notice that A is moving away from D, while B is moving along the same road as D

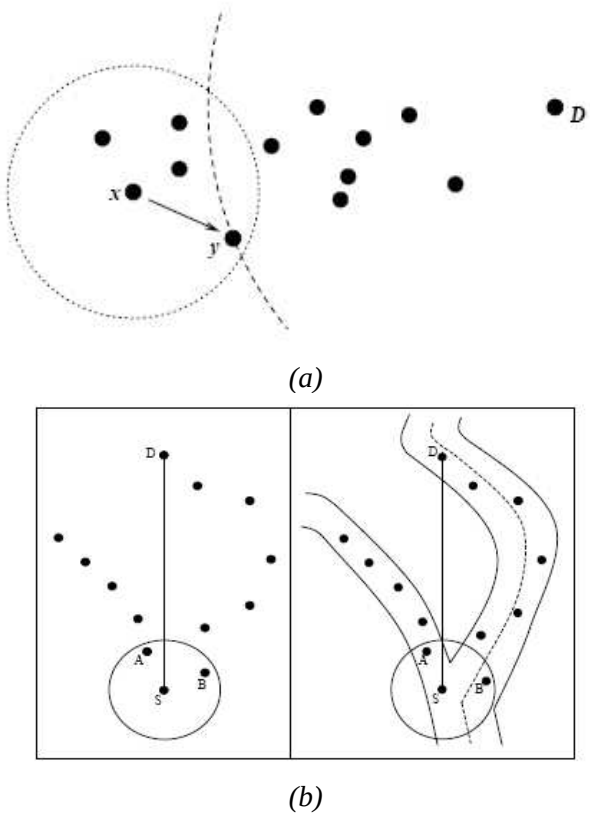


Fig. 1. *(a)* Example of packet forwarding with GREEDY from [17] *(b)* Example of packet forwarding with SAR from [19]

and is thus approaching the destination. B is then chosen as the next hop by SAR, rather than A. When a packet must be sent, SAR computes the shortest path between source and destination on the graph extracted from the map, for instance exploiting Dijkstra's algorithm. Some reference points (*Anchor Points*, or APs for short) on the map traversed by the shortest path are taken to form a *Geographic Source Routing* (GSR), that is a list of coordinates. The GSR is inserted into the packet header. Each node forwards the packet to the neighbor in its cell that is the nearest to the next AP in the GSR. Once an AP is reached, it is removed from the GSR and routing continues toward the successive AP listed. Also in this case, a node could have no neighbors appropriate to carry on packet forwarding, and could thus resort to a greedy approach, or recalculate the GSR, or temporarily maintain the packet in a *suspension buffer* trying to continue the packet routing as soon as it has a suitable neighbor.

Other solutions have been proposed, such as *restricted directional flooding* or *hybrid/ hierarchical routing*. With the former approach, adopted for instance by *Distance Routing Effect Algorithm for Mobility* (DREAM) [5] and *Location Aided Routing* (LAR) [21], the vehicle

source of a packet sends it to all its neighbors that are included in a region computed basing on the source and the destination positions. If no neighbors are available inside such a region, the packet is flooded to all neighbors. These algorithms clearly do not scale well. The hybrid/hierarchical approach is for instance adopted by TERMINODES [22] and GRID [23]. Packets are routed using a greedy position-based algorithm until they reach the neighborhoods of the destination; at this point a proactive distance-vector algorithm is used.

In [24], a comparison is performed among some of the cited algorithms, showing that GREEDY has a lower communication complexity than restricted directional flooding, while is comparable with hybrid/hierarchical solutions. Moreover, GREEDY has a lower implementation complexity than hybrid/hierarchical solutions. On the other hand, SAR is the unique algorithm expressly designed for VANETs and is an extremely promising approach as it is designed to work according to the behavior of vehicles in terms of movement pattern. In sec.V, we compare the performance of STAR with GREEDY as the basic technique, GPSR that is currently considered standard for MANETs, and SAR.

IV. Spatial and Traffic-Aware Routing (STAR)

STAR approach to vehicular routing problem is quite different from other position-based routing algorithms. As discussed in sec.III, the strength of SAR lies in the knowledge of the topology that constrains vehicle movements: a node can exist only along a road. This means that to a certain extent SAR can *see* where to forward packets: the only way to route a packet is along streets. If there are no streets from source to destination, then a packet cannot be routed at all. Although knowing street topology is a big advantage, this approach can fail in case the algorithm tries to forward a packet along streets where no vehicles are moving. Such streets should be considered as "broken links" in the topology. This problem can be overcome by knowing the *real* topology, that is, by trying to use for packet forwarding only streets where vehicular traffic exist. This can be obtained through cooperation among vehicles in order to detect anomalous situations in vehicle distribution on streets, and to collect and spread information about the real topology and state of the network in real time. Status knowledge should not concern the whole network: collecting and exchanging information about topology can be expensive, and on the other hand this information is highly volatile due to node mobility. To avoid wasting resources in useless updates of status information and achieve high scalability, it is thus preferable that each

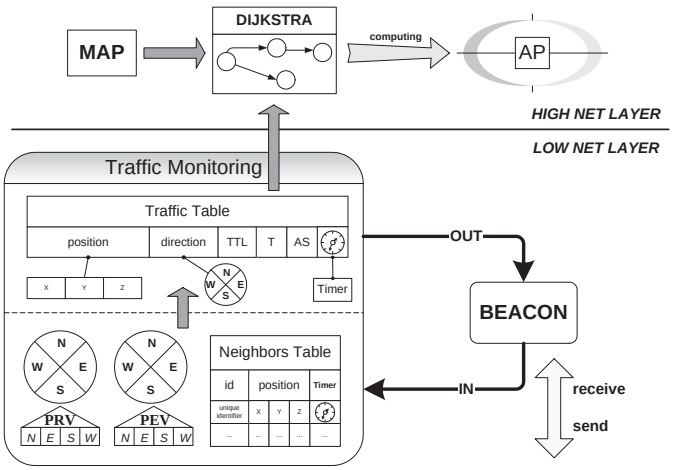


Fig. 2. Functional architecture for the STAR algorithm

node gathers partial knowledge of the network concerning a restricted area around its position. Similarly, it is convenient to compute only a *partial* path to approach the destination position by determining only a subset of Anchor Points. When a packet arrives to the last AP computed for it, the node responsible for forwarding takes in charge the characterization of the next APs. Partial successive computation of the path has a threefold advantage. First of all, independently of the path length, the size of packet header is fixed because the number of APs' positions it must contain is upper bounded. Moreover, the computation of subsequent APs needed to carry on packet forwarding is done exploiting more updated and accurate information about traffic distribution, thus decreasing the probability of a packet reaching a local maximum. Finally, in case querying the location service is not expensive in terms of latency, subsequent APs can be computed exploiting updated information about the current position of the destination.[1]

To reach this objective the STAR algorithm is organized in two layers (fig.2): a lower layer that manage the gathering and exchange of information about network status and a higher layer for the computation of paths. In the following subsections we explain the functionalities deployed at each layer. In subsection IV-C, we discuss in detail STAR operations and dimensioning of parameters.

[1]Let us notice that, however, the "speed" at which packets are forwarded is far higher than vehicle speed. As a consequence, the destination position should change negligibly throughout packet transmission, and the lack of an efficient location service should not impact severely on STAR performance.

A. Traffic Monitoring

The lower layer of STAR is in charge of finding and diffusing information about the network status. As we previously noticed, a relevant information for packet forwarding is vehicles' distribution along streets. In particular, we are interested in detecting two extreme situations: the presence of queues of vehicles or the total absence of vehicles. Streets where queues of vehicles form should be preferred, as they provide several alternatives for packet forwarding, thus minimizing the risk of a packet reaching a local maximum. By contrast, the routing algorithm *must* avoid streets where vehicles are not present, because packets cannot for sure be routed over them as long as their status does not change.

Monitoring and propagation of vehicular traffic conditions are performed through the exchange of network-level *beacons* (fig.2), carrying observations of node neighborhoods. The observations are maintained in data structures managed by the *traffic monitoring* module.

A node maintains the `neighbors-table`, that is, a table with the position of its neighbors, as triples $\langle latitude,\ longitude,\ altitude \rangle$. Node neighborhood is discovered via the beacons. In the `presence vector (PRV)` a node maintains four node counters, which represent the number of neighbors it has toward cardinal points (north, east, west, south) computed dividing the node cell into sectors as shown in the figure. Each counter is incremented when a neighbor is discovered in the corresponding direction and decremented when a node leaves the `neighbors-table`, that is, when neighborhood information is not refreshed for a certain time.

When a `PRV` counter exceeds a parameter `highPR` an abnormal traffic condition is detected: a high concentration of vehicles exists in the corresponding direction. By contrast, an elements of `PRV` below parameter `lowPR` indicates scarce vehicular traffic in a street lying in the corresponding direction of the node. When one of these situations occurs, the modification of a related element in the `persistence vector (PEV)` is triggered. `PEV` has four elements as `PRV`. Each element can be in one of three different conditions: it could be in a *reset state* (a value equal to 0), or in a *growing state* (value > 0), or in a *shrinking state* (value < 0). In the event of a `PRV` counter exceeding `highPR`, if the corresponding `PEV` element is in either *reset* or *growing* state, then it is incremented; otherwise the `PEV` element is set to *reset* state (0 value). On the other hand, in the event of a `PRV` element below `lowPR`, if the corresponding `PEV` element is in either *reset* or *shrinking* state then the element is decremented; otherwise it is set to *reset* state. `PEV` is used to register critical situations only when they last for a long time. When the value of an element of `PEV` goes out from a range [`lowPE`, `highPE`], then the information about traffic represented by the element is recorded in the `traffic-table` and the value of the `PEV` element is reset. The use of `PEV` is necessary to guarantee that a temporary abnormal condition is not registered, but if it lasts then it is registered in the `traffic-table`.

Each node has a `traffic-table`. Each entry in this table has five fields to keep track of traffic conditions, namely: `position`, `direction`, `traffic bit (Tbit)`, `already-sent bit (ASbit)` and `Time-To-Live` (`TTL`). The `position` indicates the coordinates where a traffic situation has been registered. The `direction` is the direction in which the traffic condition is taking place with respect to `position`. `Tbit` specifies the type of traffic (high or low). `ASbit` records whether a traffic entry has been already propagated to neighbors. `TTL` is the number of hops the information has to travel and is decremented every time the entry is forwarded. Each entry has an associated `traffic-timer`. When the timer expires, the entry is removed from `traffic-table` in order to forget obsolete information about traffic anomalies that do not exist anymore.

Beacon exchange. Each node periodically broadcasts to its neighbors a network-layer beacon that contains sender identifier, sender coordinates and the vehicular traffic conditions it has in its `traffic- table`. Broadcasting period is determined by a `beacon-timer`.

When a node receives a beacon, first of all it registers the information about the sender in its `neighbors-table`; `PRV` and `PEV` are possibly updated as explained before. If one of the elements of `PEV` becomes either lower than `lowPE` or higher than `highPE` then a new entry is added in the `traffic-table`. The new entry has as `position` the coordinates of the node, `direction` equal to the corresponding element of `PEV` ('N', 'E', 'S' or 'W'), `ASbit` set to zero and `TTL` set to a value `maxTTL`, which determines how far traffic information will be spread. `Tbit` is set to the appropriate value 'H' or 'L' according to the vehicular traffic condition detected. Then each traffic entry is copied from the beacon into `traffic-table`, with `TTL` value decremented by one. In updating the `traffic-table`, existing entries must be compared with the information carried by the beacon. In case two matching entries exist in the beacon and the table, and they have the same `direction`, while `position`s dif-

fer less than a parameter `traffic information distance (TID)`, then the two entries refer to the same traffic condition. Only the one of them having the highest `TTL` is kept. This allows to suppress duplicate advertisements, still guaranteeing that abnormal conditions are notified. If `TTL`s are equal, the `traffic-timer` determining entry expiration time is set to the initial value, and the `ASbit` is set to 0. This occurs when a traffic anomaly persists for a certain time, and is thus advertised more than once (TTLs are equal in the two advertisements); the receiving nodes must refresh the corresponding `traffic-table` entry and re-propagate this information.

When a node's `beacon-timer` expires, the node creates a new beacon carrying its own identifier and position. Each traffic entry from the `traffic-table` is copied into the beacon if and only if `TTL` > 0 and `ASbit` = 0. Then the `ASbit` of the entry is set to 1. This prevents diffusing multiple times a certain information. Finally the beacon is sent.

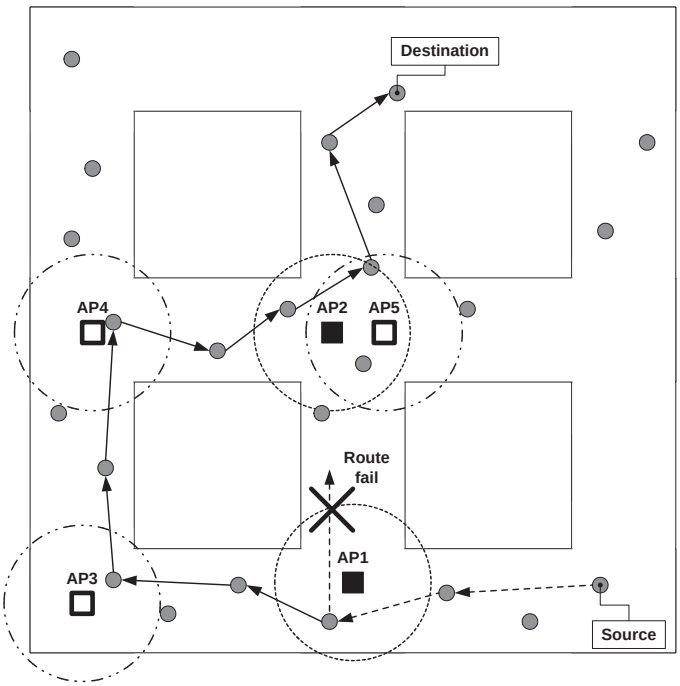


Fig. 3. Example of routing with STAR

B. Routing and packet forwarding

At the higher STAR layer (fig.2), routes are computed *on-demand* exploiting owned information about traffic and neighbors. When a source S has a packet to send to a destination D, first of all S builds a *weighted graph* using street map and traffic information. Edges corresponding to streets without traffic have associated a high weight, so as to discourage their usage on behalf of the algorithm. By contrast, when in a street there is high vehicular traffic, then the weight of the associated edge must be decreased so as to privilege the choice of this street although it could characterize longer paths. As a consequence of vehicles' mobility, weights of the edges are dynamically adjusted. Initial edge weights and the mechanism of weight adaptation must be such that they guarantee that weights never become negative or null. In sec.IV-C, we describe the mechanism we adopted in simulations.

Dijkstra's algorithm is applied to the obtained graph in order to find the shortest route along streets. APs are computed along the streets belonging to the route, like in SAR, to force the packet to travel along them. The packet header includes the destination identifier, the destination position and a limited number of APs, equal to `maxAP`. The value of `maxAP` must be selected accurately and should be in relation with `maxTTL`, as discussed in sec.IV-C. Then the packet is forwarded with geographic greedy routing algorithm toward the first AP. When it is reached, the packet will be forwarded towards the next AP. When the last AP initially computed has been reached, the node who has to forward the packet will compute others APs toward the destination, until it is reached.

In fig.3, an example is shown of packet routing with STAR. A source node – on the bottom right of the figure – has a packet addressed to a destination in the top middle of the figure. By exploiting current knowledge about vehicular traffic and Dijkstra's algorithm, the source computes APs 1 and 2 (shown as full black squares) and sends the packet to its neighbor that is nearest to AP 1 (dashed arrows). At a certain point a node, passed AP 1, takes in charge the packet forwarding toward AP 2; but it fails because it has no neighbors in the appropriate direction, that is, the packet is in a local maximum. Exchanging traffic information allows to take more accurate decisions about routing, but does not make routing failure impossible. The *recovery procedure* adopted by STAR consists in computing new APs (empty gray squares) from the current node, exploiting updated traffic information. The packet is forwarded along the new route (gray arrows). In case of extremely peculiar vehicle distributions, a packet could be routed by recovery procedure more than once, and then be dropped because of TTL expiration before reaching its destination; in this case STAR fails.

A pseudo-code summarizing the algorithm is shown in Appendix.

C. Details of STAR algorithm

STAR behavior is controlled through some other parameters. In this section, we explain these parameters and discuss the issues related with their dimensioning.

Parameters for beacon exchange. Two parameters are used to set beacon generation frequency. The `beacon-interval (BI)` is the fixed amount of time that elapses before sending the next beacon. The `beacon-desync (BD)` is the upper layer of a random amount of time added to `BI` to desynchronize beacon broadcast among neighbors, thus reducing collisions.
When a node receives a beacon, timers are initialized in both `neighbors-table` and `traffic- table` at `beacon-expire (BEXP)`. When the timers expire, the position and traffic information that were carried in the beacon are removed from both tables.
The `max-traffic-information (maxTI)` is the maximum number of `traffic-table` entries that can be included in a beacon. It is an upper bound to limit overhead and prevent network congestion. If parameters are well chosen, this bound should be never reached. However, if more traffic entries should be sent than those that can be accommodated in a beacon, only part of them is sent, their `ASbit` is set to 1, and the remaining entries with `ASbit` equal to 0 are sent with the successive beacon.

Parameters related with street topology. If a node is far from an AP less than `AP-range-accept (APRA)`, then the AP is considered reached.
The `CROSS RANGE ACCEPT (CRA)` parameter is introduced to enhance STAR traffic detection: if a vehicle is moving along a straight road it does not need to collect information about traffic in (non-existent) lateral streets. Only in proximity of a crossroad traffic information must be collected in every direction. If a vehicle is far from the nearest crossroad more than `CRA` meters, then it stops detecting traffic orthogonal to the direction in which it is moving.
Neighbors influence PRV update only if their reciprocal distance is higher than `presence-vector-distance (PRVD)`. Actually, this parameter is introduced to cope with our mobility model: vehicles have no dimensions, and many vehicles can be compressed in one point without any other node along the street. Without this parameter, this low traffic condition could not be detected.

Parameters for route computation. If a traffic information is collected less than `Dijkstra -cross-assume (DIJCA)` meters from a crossroad, then that information is tied to the crossroad when building the weighted graph.
When computing APs, if a node is distant less than `Dijkstra-node-to-cross-range (DIJNCR)` meters from a crossroad, it is considered as being on the crossroad.
The `Dijkstra-starting-weight (DIJSW)` is the weight initially assigned to each edge when building the weighted graph, while `Dijkstra-high-weight (DIJHW)` and `Dijkstra-low- weight (DIJLW)` are respectively weight increment and decrement for an edge with associated information of high and low traffic. `DIJHW` is a negative value, to make more eligible a street with high traffic. By contrast, `DIJLW` is a positive value. In a regular Manhattan street topology, to guarantee that empty streets are avoided, `DIJLW` must be three times `DIJSW`.[2] Moreover, to prevent an edge weight to become negative as a consequence of duplicate notifications of the same traffic anomaly, the following equation should be satisfied: `DIJSW` $>$ `DIJHW` $\times street\_length/$`TID`, where $street\_length$ is the length of a street included between two crossroads.
The `max-anchor-point (maxAP)` is the maximum number of APs that are included in a packet. This parameter is related with `maxTTL`: if `maxAP` is large with respect to `maxTTL`, some APs are computed without relying on traffic information. By contrast, a small `maxAP` could waste computation time because a vehicle refrains from using the information it owns to compute a longer path, but leaves this task to other nodes. Computing more APs allows greater accuracy in choosing a path; on the other hand, this implies higher overhead in both packet header and beacon traffic, as a consequence of the higher `maxTTL` needed.

V. Performance Analysis

STAR performance has been analyzed using the NS-2 simulation package [25], under different parameter settings. STAR performance has been compared with results achieved by using the GREEDY approach without any recovery procedure, the GPSR approach, and the SAR algorithm. In the following subsections the simulation conditions are described, and the simulation results are discussed.

A. Simulation parameters

Simulations have been performed adopting a city mobility model, applied to a Manhattan street map

[2]This way, a certain point can be reached avoiding an empty street by going around a building block via three streets, as in fig.3.

formed by 5 horizontal streets and 5 vertical streets. Distances between adjacent streets are equal to 400 mt., thus characterizing a regular grid. The number of nodes is 250; the communication range is 250 mt. A small street length with respect to the communication range has been chosen to advantage GREEDY and GPSR, which have not been specifically designed for vehicular networks and do not involve mechanisms to deal with mobility along streets. Nodes move along the streets and can change their direction at crossroads; vehicle speed is 50 Km/h. Speed does not affect routing of a certain packet, according to the observation that packets travel node-by-node at a speed much higher than vehicle speed. However, it can affect STAR performance in case vehicle speed is so high that a node neighborhood can change dramatically between two successive beacons and this cause a node to perform wrong routing decisions using obsolete information. We call this the *"vanishing neighbor" effect*, as a node can try to forward a packet to a node that was its neighbor but has moved out of communication range. In sec.VI, a mechanism to cope with this problem is proposed. Simulated time is 200 sec. Results are averaged on 105 packets, exchanged between 5 source-destination pairs.

As discussed in sec.III, the main problem faced by position-based algorithms is that packets can reach a local maximum, where no neighbor exists appropriate to take in charge further packet forwarding. The probability of this event depends on the distribution of vehicles along streets. Simulations have been performed in three different scenarios: with all crossroads usable by vehicles, and with 4% and 8% of crossroads without traffic.

Measures have been performed varying 4 parameters, namely:

- the threshold `lowPR` that can assume values 0 or 1;
- the `maxTTL` adopted to diffuse traffic information that can assume values 10 or 20;
- the threshold `lowPE` that can assume values -2 or -4;
- the parameter `CROSS RANGE ACCEPT` that can assume values 10, 20 or 30.

Other parameters discussed in sec.IV-C are set to values reasonable in a real environment, as shown in Table I. The detection of dense traffic conditions has been disabled because of the problem with the mobility model discussed apropos the `PRVD` parameter, and the consequent impossibility of simulating queues of vehicles. Hence, `highPR` has a very high value while `highPE` does not even need to be defined. In Table II, we associate to each simulated scenario a label used in following plots, for the sake of readability.

TABLE I
PARAMETER SETTINGS USED IN SIMULATIONS

`BD`	0.25	`DIJLW`	20
`BI`	0.25	`TID`	150
`BEXP`	$3 \times$ (`BI` $\times(1+$`BD`$)$)	`PRVD`	20
`APRA`	230	`highPR`	100
`DIJCA`	100	`DIJSW`	4
`DIJNCR`	50	`maxAP`	5
`DIJHW`	-1	`maxTI`	30

TABLE II
SIMULATED SCENARIOS

Scenario	`lowPR`	`maxTTL`	`lowPE`
A	0	10	-2
B	0	10	-4
C	0	20	-2
D	0	20	-4
E	1	10	-2
F	1	10	-4
G	1	20	-2
H	1	20	-4

The performance indexes measured with simulations are:

- **percentage of delivered packets:** this index shows the effectiveness of the routing algorithm;
- **percentage of lost packets:** packets can be lost because of collisions, or because the routing algorithm fails in getting them delivered to their destinations. This index only accounts for packets lost due to failures of the routing algorithm, with STAR performing only one path re-computation before deciding to drop the packet;
- **average beacon size:** large beacons increase the probability of collisions with both other beacons and data packets; hence, this index can explain other packet losses not due to routing failures.

The average path length has also been measured; this index was almost constant throughout all simulations. However, this result is not really significant as it has been measured only for delivered packets, for which the routing algorithm succeeded in find an appropriate route through streets.

Finally, we compared the best parameter setting obtained for STAR among the eight scenarios listed above, against the performance achieved by GREEDY, GPSR and SAR under the same conditions.

B. Simulation results

In figg.4 and 5*(a)*, the performance indexes achieved for the eight considered scenarios and different values

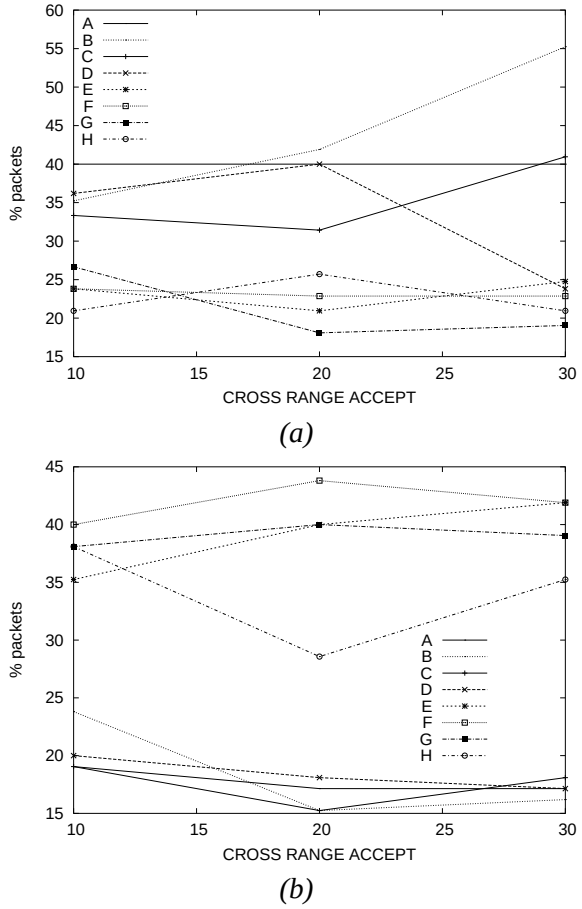


Fig. 4. Percentage of *(a)* delivered and *(b)* lost packets with 4% of crossroads without traffic.

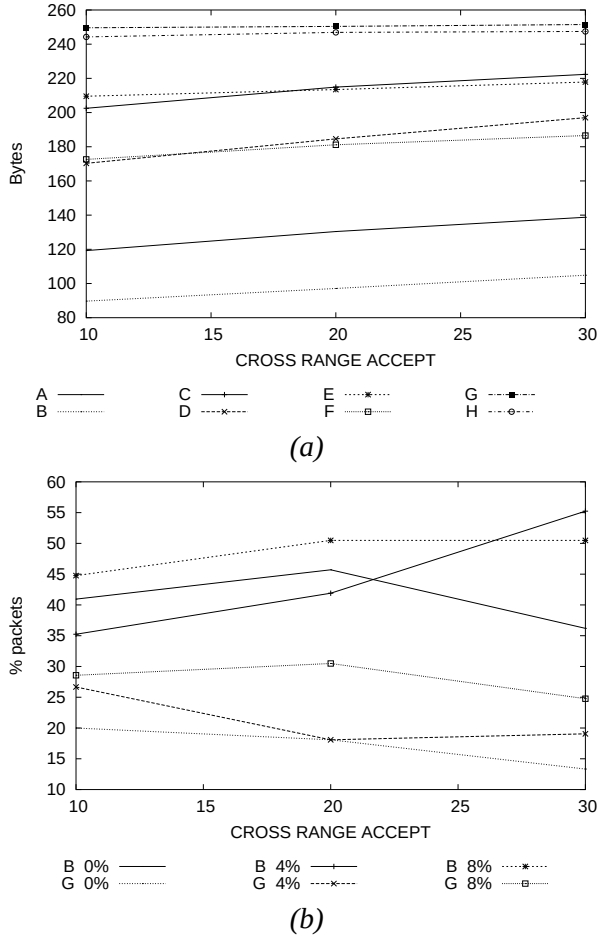


Fig. 5. *(a)* Average beacon size with 4% of crossroads without traffic. *(b)* Percentage of delivered packets: comparison of the best and the worst parameter settings for different percentage of crossroads without traffic

of `CROSS RANGE ACCEPT` are shown, in case 4% of crossroads are not used by vehicular traffic.

Scenarios tend to separate into two subsets, depending on the value of the `lowPR` threshold. In particular, a high threshold corresponds to a higher value of lost packets. This can be explained considering that when `lowPR` equals 1, then absence of vehicular traffic is signaled in a certain area although one vehicle is available for packet forwarding indeed. As a consequence, the number of streets that appear usable to forward data decreases, thus yielding greater probability of routing failure. Moreover, a higher signaling threshold implies a higher number of advertisements for lack of traffic, and as a consequence larger beacons that increase collision probability, as shown in fig.5*(a)*. As a matter of fact, by comparing graphs 4*(a)* and 5*(a)*, it can be noticed that for increasing beacon size the percentage of delivered packets decreases because of losses due to collisions. The best behavior is achieved with the scenario tending to minimize the beacon size – namely, scenario **B** – by being prudent in both detecting and signaling absence of traffic (low `lowPR` threshold and high persistence of the traffic condition) and propagating the notification only in a restricted area (low `maxTTL`). Obviously, the worst scenario is thus **G**.

As far as the `CROSS RANGE ACCEPT` parameter is concerned, similar considerations can be drawn. A small value makes difficult to detect empty streets, thus making the advertisement mechanism ineffective. On the other hand, a high value increases the advertisement overhead. When streets exist with scarce traffic, this event can be detected by means of observations on orthogonal streets (where vehicles travel). As a consequence, the probability of delivering packets increases with higher `CROSS RANGE ACCEPT` value for increasing number of empty streets, because this increases the probability that the abnormal traffic condition is detected and advertised. As a side effect, effective advertisement also reduces collisions: if a packet is appropriately routed till the beginning by exploiting accurate information about the traffic distribution, then the probability of it arriving in a local maximum is decreased. When a packet reaches a

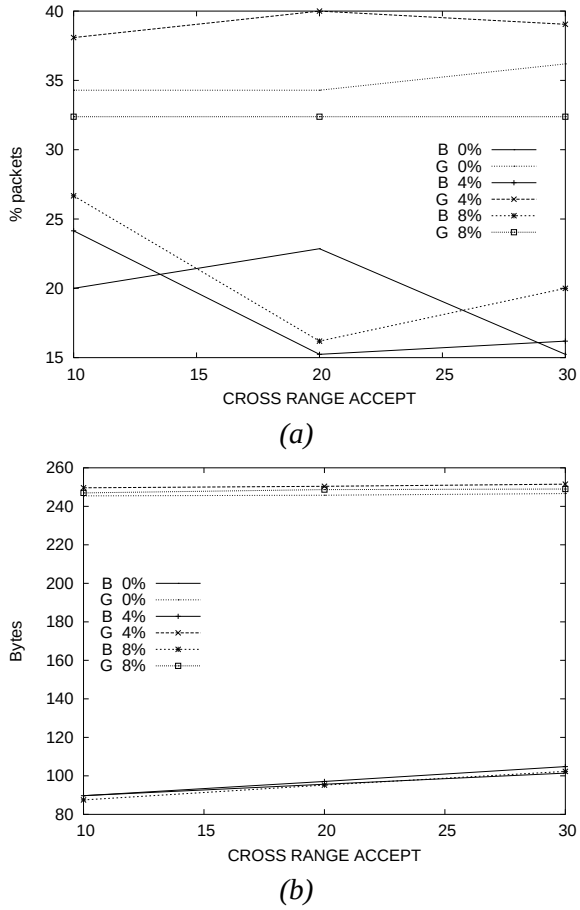


Fig. 6. Comparison of the best and the worst parameter settings for different percentage of crossroads without traffic: *(a)* percentage of lost packets and *(b)* average beacon size

local maximum, an alternative route must be computed, thus forcing the packet to follow a longer route and increasing the probability that along that route the packet can suffer a collision.

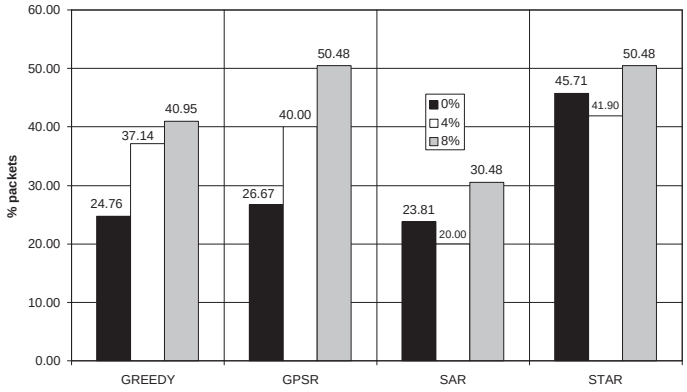
Comparable behavior has been achieved with both 0% and 8% of empty streets. In fig.5*(b)* and 6, we analyze how the distribution of vehicular traffic impacts on STAR performance, by considering only scenarios **B** (best) and **G** (worst). As expected, best and worst scenarios tend to separate into two distinct groups. Inside each group, one can observe that the percentage of delivered packets increases with the percentage of empty streets. This proves that STAR is effective in avoiding packets reaching local maximums and, on the other hand, it can take advantage of high node density on the remaining streets to improve the guarantee of packet delivery. The percentage of lost packets shows an opposite behavior. Fig.6*(b)* makes apparent the difference in beacon size between the two analyzed scenarios: scenario **G** produces beacons whose size is three times that of the beacons generated under scenario **B**, and is almost the maximum beacon size allowed. On the other hand, the fact that the beacon size is independent of the traffic distribution means that duplicate advertisements are suppressed, thus avoiding bandwidth waste and guaranteeing that STAR is scalable.

It is worth to notice that, through comparison between figg.4*(a)* and *(b)* – as well as between fig.5*(b)* and fig.6*(a)* – the percentage of packets destroyed because of collisions varies from 20% to roughly 40%, and in some cases more packets are lost because of collisions rather than because of routing failure.

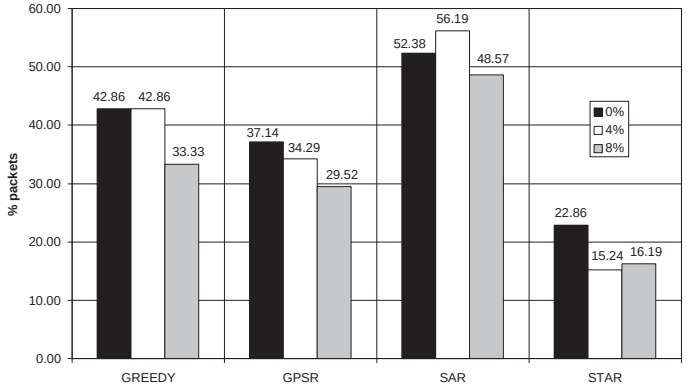
Hence, simulations bring into evidence that STAR is highly sensitive to collisions. As far as permanence of traffic information is concerned, by examining fig.4*(a)* and comparing scenarios differing only in `lowPE`,[3] we observe that in general high persistence values yield better results. However, some scenarios take advantage of low persistence values. This occurs for instance in case of vehicles that are moving toward an empty street but change their direction at a crossroad too soon to allow that the *'empty street'* event becomes persistent enough to be advertised. These considerations lead us to devise possible improvements, which are discussed in sec.VI.

Finally, we compare the best scenario, namely **B**, with the performance obtained by GREEDY – without any recovery mechanism – GPSR and SAR in the same conditions, with `CROSS RANGE ACCEPT` set to 20 mt. It has to be noticed that using a Manhattan street map simplifies the route computation. Hence, map knowledge gives both SAR and STAR less advantages over the other two policies than expected, as there are not blind alleys or street forks, for instance. Moreover, SAR implementation does not involve any recovery procedure. In analyzing simulation results, it is worth to notice that so far a realistic simulation of mobility models is still missing. The model we used does not provide the possibility of simulating queues of vehicles. As a consequence, if two vehicles move along the same street and the vehicle behind has a greater speed than the other, then the model allows that it overtakes the other vehicle, instead of queuing behind it. We also observed that vehicles tend to be more dense around crossroads, thus having greater probability of packet collisions. Lack of queues and high collision probability negatively impact on STAR performance with respect to that achievable in a real environment. Moreover, the simulation environment does not allow to simulate radio signal attenuation due to obstacles. As a consequence, messages can be exchanged between two vehicles that

[3]I.e., comparing scenarios A with B, C with D, E with F and G with H.

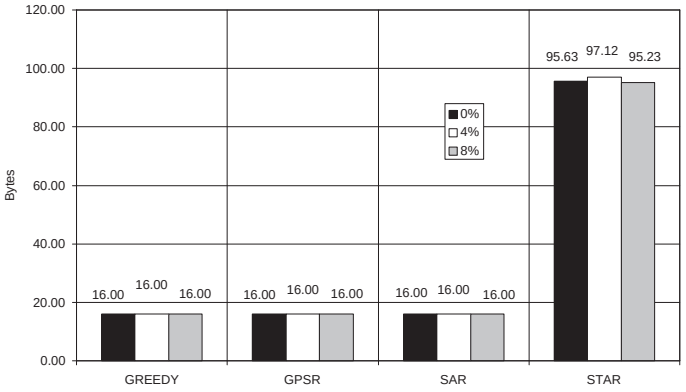


(a)

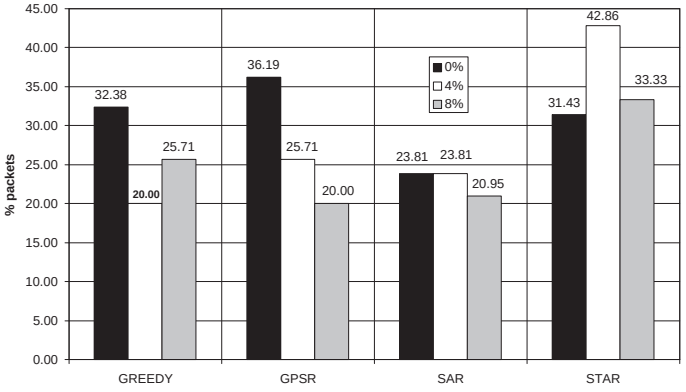


(b)

Fig. 7. Algorithm comparison: *(a)* percentage of delivered packets and *(b)* percentage of lost packets



(a)



(b)

Fig. 8. Algorithm comparison: *(a)* average beacon size and *(b)* percentage of lost packets due to collisions

actually would not be in communication range because of a building between them. This characteristic – together with streets short with respect to the communication range – advantages GREEDY and GPSR, while impacts to a lower extent on both SAR and STAR as they follow APs – and thus streets – to forward packets.

In fig.7*(a)*, the percentage of delivered packets is shown. STAR is comparable with GPSR, that is a GREEDY approach involving a recovery procedure, but it is advantaged by knowledge of street map. Both GREEDY and GPSR behave worse in the 0% scenario, because in that case vehicle density is lower than in the other scenarios as vehicles can be distributed all over the considered area. By fig.7*(b)*) it can be noticed that STAR is far better than the other algorithms with respect to routing failure probability, thus confirming that traffic information is of help to perform accurate routing decisions. Indeed, the remaining packets not delivered to their destinations have been lost because of collisions (fig.8*(b)*). On the other hand, although GPSR recovery procedure provides comparable guarantees of packet delivery in dense networks, the routes achieved with it are 1 to 2 hops longer than those computed with STAR.[4] SAR does not behave well because of both the lack of recovery mechanisms and the simplicity of the street map. If street length were higher, GREEDY and GPSR would drop many more packets because they are unable to find a longer route following streets to forward data to destination. STAR is similarly affected by short streets as SAR, in the comparison with both GREEDY and GPSR.

Distributing information about vehicular traffic drastically increases the size of network-layer beacons (fig.8*(a)*) with respect to the other considered algorithms. However, beacons have almost a constant size independently of the traffic distribution along streets. This can be explained by the fact that many advertisements probably refer to the same streets empty of traffic; the mechanism to suppress duplicate advertisements prevents beacons to carry redundant information, thus effectively limiting beacon size. Because of larger beacons, collisions suffered by STAR quite significantly overcome those

[4]With path lengths of 7-8 hops on average.

observed with other algorithms (fig.8*(b)*), unless for scenario 0% where a lower average node density reduces collision probability.

VI. Conclusions and future works

In this paper we proposed a novel routing algorithm for vehicular ad hoc networks (STAR), and we measured its performance in comparison to other algorithms existing in the literature. STAR performs better than the other considered algorithms in spite of having an inaccurate mobility model that imposed disabling detection of high traffic conditions in simulations. However, STAR greatly suffers collisions: in some scenarios more than 40% packets are lost because of collisions. Moreover, we have observed that in scenarios with zones without vehicles, performance was less good than expected. This is due to orthogonal detection issues: a vehicle may pass by a street without vehicles at a speed so high that it can fail in detecting the abnormal situation. This problem can be dealt with by dynamically adapting STAR parameters in order to achieve more prompt anomaly detection, as discussed below.

STAR can be improved to some extent. The availability of MAC layer able to reduce packet collisions would allow to achieve better performance in terms of delivered packets. Parameters ruling traffic collection and information diffusion are the core of STAR. A balance should be found between having up-to-date information and low occupation of channel resources. Instead of fixed values, a better solution consists in considering *dynamic parameters adaptation*. For instance, when a traffic anomaly is first detected, then information is spread immediately to neighbors. If the anomaly persists, it is detected several times. To limit repeated broadcasting of information about a certain anomaly, each time detection takes place the absolute value of PEV thresholds could be incremented, thus avoiding continuous alarm triggering. At the same time it is necessary to increase the initial value to which the `traffic-timer` is set every time information is refreshed by beacons, otherwise information expires in spite of anomaly persistence because of less frequent beaconing. Similarly, anomalies that persist for a long time could be advertised over a larger area (i.e., have a greater associated `maxTTL`), thus allowing to perform more accurate routing decisions. `maxTTL` could be dynamically adapted also according to the number of traffic anomalies to be advertised: the fewer the anomalies, the greater `maxTTL`, in order to control the amount of beacon traffic.

Reducing beacon frequency would save channel resources and would help in collision reduction. As a side effect, this would yield a less accurate knowledge

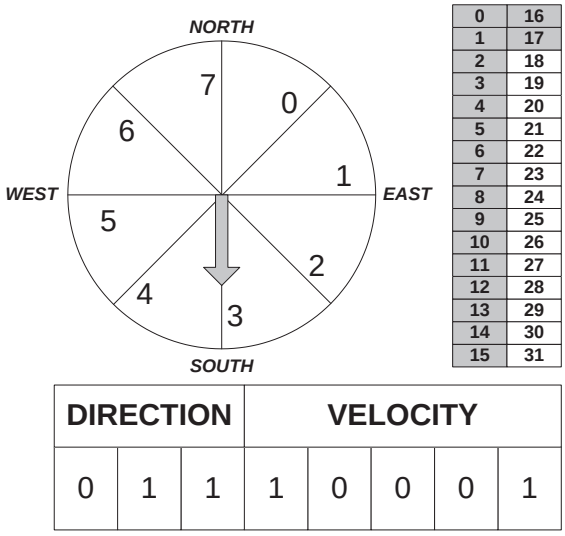


Fig. 9. Graphical representation of DV vector

about neighbors and their positions, thus leading to an higher probability of algorithm failure. Reducing beacon frequency and preserving accurate neighbors position can be obtained through *neighbors forecasting*: by introducing direction and speed of nodes in beacons, a forwarding node can compute on the fly the actual position of neighbors before choosing next hop. This also allows to know if a node is gone out of transmission range. Speed and direction of node can be represented with a one-byte vector, called *Direction and Velocity (DV) vector* (fig.9). Five bits are used to store speed and three bits are used to store direction. Speed is represented by dividing the admissible speed range into 32 subranges; the vector contains the number of the subrange including the current vehicle speed. With a little growth in beacon size (which is greater than 90 bytes in our simulation) beacon overhead can be reduced by reducing beacon frequency. This mechanism would also allow to reduce the impact of the *"vanishing neighbor"* effect.

We are currently working on designing and optimizing the mechanisms described above. A more realistic vehicular mobility model must also be developed, in order to ensure that algorithms are simulated in a realistic way and more accurate results are obtained. Such a model would also allow to implement and evaluate the effectiveness of exchanging information about high traffic conditions. Anyway, this task is not trivial, as it involves correlating positions and speeds of different vehicles in order to adjust their reciprocal movements, avoid vehicles overlapping in the same point, and simulate queues.

STAR characteristics make it suitable for several distributed services besides of routing. As an example,

STAR could be exploited as a *Smart GPS navigation system*. Information about traffic condition is valuable not only for packet routing, but also for car route choice. The knowledge about traffic conditions can be shared with a GPS navigation system, so that it can select a path basing on traffic conditions, thus avoiding queues. In order for position-based routing to work, a wide spread of position-based systems on vehicles is needed. To achieve this goal the smart navigation service can be exploited to push customers to buy this technology. STAR approach could be also exploited to exchange information about network bandwidth available, so as to drive routing decisions for instance in case Quality-of-Service guarantees are required, or to announce services available to other nodes.

ACKNOWLEDGMENTS

We would like to thank Prof. Gian Paolo Rossi for useful discussions and his helpful comments in preparing the paper.

REFERENCES

[1] IETF MANET Working Group, *"Mobile Ad Hoc Networks"*. http://www.ietf.org/html.charters/manet-charter.html.

[2] PaloWireless, *"Global Positioning System (GPS) Resource Center"*. http://palowireless.com/gps/.

[3] IEEE 802.11 Working Group for Wireless Local Area Networks, *"Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band"*. 1999, http://standards.ieee.org/getieee802/802.11.html.

[4] Grossglauser M., Vetterli M., *"Locating Nodes with EASE: Last Encounter Routing in Ad Hoc Networks through Mobility Diffusion"*. Proc. IEEE INFOCOM'03, Mar.2003.

[5] Basagni S., Chlamtac I., Syrotiuk V., Woodward B., *"A Distance Routing Effect Algorithm for Mobility (DREAM)"*. Proc. 4th Annual ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MOBICOM'98), pp. 76-84, 1998.

[6] Li J., Jannotti J., De Couto D.S.J., Karger D.R., Morris R., *"A Scalable Location Service for Geographic Ad Hoc Routing"*. Proc. 6th Annual ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MOBICOM'00), pp. 120-130, 2000.

[7] Stojmenovic I., *"Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks"*. Technical Report TR-99-10, Computer Science, SITE, University of Ottawa, Sep. 1999.

[8] Käsemann M., Füssler H., Hartenstein H., Mauve M., *"A Reactive Location Service for Mobile Ad Hoc Networks"*. Technical Report TR-14-2002 , Department of Computer Science, University of Mannheim, Nov. 2002.

[9] Perkins C., Bhagwat P., *"Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers"*. Computer Communication Review, pp. 234-244, Oct. 1994.

[10] Perkins C., Belding-Royer E., Das S., *"Ad Hoc On-Demand Distance Vector (AODV) Routing"*. RFC 3561, July 2003. Work in progress.

[11] Johnson D., Maltz D., *"Dynamic Source Routing in Ad Hoc Wireless Networks"*. In "Mobile Computing", chapter 5, pp. 153-181, T. Imielinski and H. Korth Eds., Kluwer Academic Publishers, 1996.

[12] Haas Z.J., Pearlman M.R., *"The Zone Routing Protocol (ZRP) for Ad Hoc Networks"*. Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June 1999. Work in progress.

[13] Takagi H., Kleinrock L., *"Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals"*. IEEE Transactions on Communications, Vol. 32, N. 3, pp. 246-257, Mar. 1984.

[14] Hou T.C., Li V.O.K., *"Transmission Range Control in Multihop Packet Radio Networks"*. IEEE Transactions on Communications, Vol. 34, N. 1, pp. 38-44, Jan. 1986.

[15] Kranakis E., Singh H., Urrutia J., *"Compass Routing on Geometric Networks"*. Proc. 11th Canadian Conf. on Computational Geometry, Aug. 1999.

[16] Nelson R., Kleinrock L., *"The Spatial Capacity of a Slotted Aloha Multi-hop Packet Radio Network with Capture"*. IEEE Transactions on Communications, Vol. 32, N. 6, pp. 684-694, Jun. 1984.

[17] Karp B., Kung H.T., *"Greedy Perimeter Stateless Routing for Wireless Networks"*. Proc. 6th Annual ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MOBICOM'00), Aug. 2000.

[18] Bose P., Morin P., Stojmenovic I., Urrutia J., *"Routing with Guaranteed Delivery in Ad Hoc Wireless Networks"*. Proc. 3rd ACM Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M'99), pp. 48-55, 1999.

[19] Tian J., Han L., Rothermel K., Cseh C., *"Spatially Aware Packet Routing for Mobile Ad Hoc Inter-Vehicle Radio Networks"*. Proceedings IEEE 6th Intl. Conf. on Intelligent Transportation Systems (ITSC), Vol. 2, pp. 1546-1552, Oct. 2003.

[20] GIS.com, *"The Guide to Geographic Information Systems"*. http://www.gis.com/.

[21] Ko Y.-B., Vaidya N.H., *"Location-Aided Routing (LAR) in Mobile Ad Hoc Networks"*. ACM/Baltzer Wireless Networks (WINET) Journal, Vol. 6, N. 4, pp. 307-321, 2000.

[22] Blazevic L., Giordano S., Le Boudec J., *"Self Organized Terminode Routing"*. Technical Report DSC/2000/040, Swiss Federal Institute of Technology, 2000.

[23] De Couto D.S.J., Morris R., *"Location Proxies and Intermediate Node Forwarding for Practical Geographic Forwarding"*. Technical Report MIT-LCS-TR-824, MIT Laboratory for Computer Science, Jun. 2001.

[24] Mauve M., Widmer J., Hartenstein H., *"A Survey on Position-Based Routing in Mobile Ad Hoc Networks"*. IEEE Network Magazine, Vol. 15, No. 6, pp. 30-39, Nov. 2001.

[25] Fall K., Varadhan K., *"The Network Simulator – NS-2"*. The VINT Project, `http://www.isi.edu/nsnam/ns/`.

APPENDIX

```
when (beacon received) do
    update neighbors-table;
    update PRV;
    /* PEV update */
    for (i IN north, east, south, west) do
        if (PRV[i] < lowPR)
                if (PEV[i] ≤ 0)
                    PEV[i] ← PEV[i]-1;
                else
                    PEV[i] ← 0;
        else if (PRV[i] > highPR)
                if (PEV[i] ≥ 0)
```

```
                    PEV[i] ← PEV[i]+1;
                else
                    PEV[i] ← 0;
        else
                PEV[i] ← 0;
    od
    /* Traffic information generation */
    for (i IN north, east, south, west) do
        if (PEV[i] > highPE or PEV[i] < lowPE)
                if (traffic information is already present in Traffic Table)
                    remove traffic information;
                create new traffic information (TI);
                TI.position ← node.position;
                TI.TTL ← maxTTL;
                TI.direction ← i;
                if (PEV[i] > highPE)
                    TI.Tbit ← high;
                else if (PEV[i] < lowPE)
                    TI.Tbit ← low;
                TI.ASbit ← 0;
    od
    merge traffic entries in beacon with traffic-table entries;
od

when (beacon-timer expires) do
    create new beacon;
    put node identifier and position in beacon;
    ∀ entry in traffic table:
        if (!ASbit and TTL > 0)
            add traffic-table entry to beacon;
            ASbit ← True;
    broadcast beacon;
od

when (data packet forwarding) do
    if (packet is for me)
        send packet to upper layer;
    else
        if (current AP is now reached)
                remove it from the packet;
        if ((no APs in packet header) or (local maximum))
                build updated graph from map and traffic-table;
                recalculate next APs and write them in packet header;
        if (local maximum)
                DROP packet;
        else
                greedy forward packet to a neighbor;
od
```