Routing in VANET's City Scenario using Back-bone Node Hop Greedy Algorithm

S. B. Kulkarni[#1], U. P. Kulkarni[#2], Sunil Begumpur[#3]

[#1]*Professor, Dept. of CSE, SDMCET, Dharwad, India*
[#2]*Professor, Dept. of CSE, SDMCET, Dharwad, India*
[#3] *P G Scholar, Dept of CSE, SDMCET, Dharwad, India*

Abstract **— Using advanced WLAN technologies; vehicular ad hoc networks (VANETs) have become useful and valuable for their wide variety of unique applications, such as safety on roads, multimedia content sharing, etc. VANETs are constrained by the high mobility of vehicles and the frequent connectivity problems. Destination positions can be found using flooding in most of the protocols in city environments. Further, in the case of sparse and void regions, recovery strategies are used which increases hop count. The minimum weighted algorithm based on distance or connectivity to select intermediate intersections are adopted in some geographic routing protocols. However, the shortest path or the path with higher connectivity may include numerous intermediate intersections. The path with higher hop counts has maximum hop counts. In this paper, we hereby propose a hop greedy routing scheme that yields a routing path with the minimum number of intermediate intersection nodes while taking connectivity into consideration. Moreover, we introduce back-bone nodes that play a key role in providing connectivity status around an intersection. Apart from this, by tracking the movement of source as well as destination, the back-bone nodes enable a packet to be forwarded in the changed direction. Simulation result shows the benefits of the proposed routing strategy in terms of high packet delivery ratio, low packet failure ratio and shorter end-to-end delay.**

Keywords— **Back-bone Assisted Hop Greedy (BAHG), Destination discovery, greedy routing, unicast routing, Vehicular ad hoc network (VANET).**

I. INTRODUCTION

Today, a vehicle is not just a mechanical machine with few electronic devices; rather, recent advancement in wireless communication technologies has brought a major transition of vehicles from a simple moving engine to an intelligent system carrier. A wide spectrum of novel safety and entertainment services are being driven by a new class of communications that are broadly classified as vehicle-to-vehicle communication and vehicle-to-infrastructure communication. Currently, intelligent transportation system components provide a wide range of services such as freeway management, crash prevention and safety, driver assistance, and infotainment of drivers and/or passengers [1]. Recent trends swing toward advertisement, marketing, and business of services and products on wheels [26]. Consequently, these applications appear to be very lucrative and promising in terms of commerce and research. The significant use of vehicular communications in safety and infotainment applications has resulted in the development of a new class of media access control and network layer protocols. The current domain of vehicular research includes routing, congestion control, collision avoidance, safety message broadcast, vehicular sensing, security, etc.

1.1 Routing challenges in City Environment

The routing issues in a city network would not be exactly the same as in a highway. The outskirts may have sparse vehicular density, whereas town has to deal with vehicular congestion. The evening may have the highest vehicular traffic, and midnight may be seen as the most silent period of the day. It is a most difficult job to predict the exact traffic density of a region. The structure of the road (i.e., straight or curved), number of intersections, number of lanes, length of the road (i.e., based on road ID), availability of public transport, and driver behaviour have a great impact on the node density and network connectivity of a vehicular network. In a city network, intersections place a unique challenge to routing protocols.

1.2 Drawbacks

A routing protocol has to key on some parameters to decide the routing path. When the routing path is the shortest distance path, it may involve a very high number of changes of directions, resulting in higher hop counts. If the connectivity is chosen as the parameter, the most connected road segment would be overcrowded by frequently routing data packets through the same path. As a consequence, the data packets experience longer queuing delays. Earlier approach suggested in the literature involves broadcasting request messages to fetch the destination position information and connectivity information. However, in a city, flooding is not advisable as multiple nodes would probe for destination position and connectivity information. As a result, every blind search (i.e., flooding) would disrupt all the ongoing communications. In our approach, we choose hop count as the metric to find the routing paths.

1.3 Advantages of proposed System

The hop greedy routing protocol exploits the transmission range and avoids intersections that are used to change the direction of the routing path. It is ensured that the selected intersections have enough connectivity. As the sender decides the routing path proactively, it is not possible to predict the

actual connectivity value without probing the whole network. We adopt an indirect method to compute the connectivity parameter for each intersection. We found that connectivity increases with the increase in the number of lanes[33]. We therefore obtain the connectivity parameter based on the number of lanes. However, packet congestion will occur as the path with the highest connectivity may be used by multiple source–destination (src–dst) pairs. Hence, we specify a connectivity threshold, and paths having connectivity parameter beyond this threshold are assigned the same connectivity status. Thus, we develop an approximation algorithm to choose a path based on both hop count and connectivity. Apart from the routing algorithm, we introduce a back-bone mechanism in which some specialized nodes perform functions such as tracking the movement of end nodes, detecting void regions on road segments, storing packets on unavailability of forwarding nodes, and selecting the most suitable intersection node as the forwarding node. Since the routing algorithm selects a path using destination position, we employ a unicast request-reply-based destination probing mechanism. To implement this approach, we divide the city into many zones that are outlined by the multilane road structures. Some dense intersections on the boundary of the zones are chosen as the boundary intersections. As the position of each boundary intersection is known, the unicast request messages initiated by the source can be easily sent to each boundary intersection. The back-bone nodes stationed at boundary intersections then take the responsibility to spread the request messages within the respective zones. The fact that unicast packets do not provide burst traffic and are shielded by request to send/clear to send (RTS/CTS) handshake [25] is the basic motivation to adopt unicast to carry out all control packet transmissions. Once the destination receives the request message, it finds a suitable path to the source and sends the reply. On receiving the reply message, the source forwards data on a routing path computed by the hop greedy routing algorithm. Finally, the routing protocol includes an update mechanism that takes care of interzone movement of end nodes.

II. RELATED WORK

The VANET has witnessed several challenges toward the development of suitable routing solutions. Originally, many routing protocols were solely designed for mobile ad hoc networks and later enhanced to suit the VANET scenarios [5], [16], [17]. Later on, few novel protocols were developed for adverse VANET environments [3]–[10], [24], [25], [28]. Currently, researchers are working on a more concrete version of routing protocols with a higher performance index. However, noteworthy pioneering works such as greedy perimeter stateless routing (GPSR) [5], greedy perimeter coordinator routing (GPCR) [6], geographic source routing (GSR) [4], vehicle assisted data delivery (VADD) [15], anchor-based street- and traffic-aware routing (A-STAR) [8], connectivity-aware routing (CAR) [2], [3] greedy traffic-aware routing (GyTAR) [24], road-based using vehicular traffic (RBVT) [28], static-node-assisted adaptive data dissemination in vehicular networks (SADV) [21], etc. have laid the foundation for routing in VANETs.

The position-based routing protocol GPSR [5] relies on the location service to acquire the position information of the destination. Basically, it uses two strategies, namely, greedy forwarding and perimeter routing, to send packets from source to destination. In greedy forwarding, a neighbor is chosen as the forwarding node if it has the shortest Euclidian distance to the destination among all neighbors. On the other hand, if no neighbor is witnessed closer to the destination than the sender itself, then perimeter routing is exercised. In GPCR [6], packets are forwarded by applying a restricted greedy forwarding procedure.

During the selection of a forwarding node, a junction node termed as the coordinator node is preferred over a nonjunction node. Note that the coordinator node is not necessarily the closest node to the destination. However, the recovery strategy in GPCR [6] remains the same as GPSR [5]. The A-STAR [8] features the best use of city bus route information to identify anchor paths. The main idea behind such arrangement is that more packets can be delivered to their destinations successfully using paths having more connectivity. Geographic source routing[4] uses a static street map and location information about each node. The sender computes a sequence of intersections using Dijkstra's shortest path algorithm [29] to reach to the destination. The sequence of intersections is placed in the data packet header. The improved GyTAR[24] is an intersection based geographical routing protocol that finds a sequence of intersections between source and destination considering parameters such as the remaining distance to the destination and the variation in vehicular traffic. The data forwarding between the intersections in GyTAR adopts either an improved greedy forwarding mechanism or a carry-and-forward mechanism, depending upon the availability of the forwarding node.

In CAR [3], the source broadcasts request messages to probe the destination. The request message caches the change of direction information and gathers the connectivity and hop count information. On receiving request message, the destination decides the routing path and replies to the source. Then, the data packets are forwarded along the path, as suggested by the destination. Additionally, the standing and moving guards take care of the position updates. In the following sections, we will discuss various issues and challenges faced by different VANET routing protocols. From the current research trends [1], [4]–[10], [16]–[19], [22], [24], [25], [28], [32]–[33] and comparisons [1], [25], it is evident that position-based routing protocols are more suitable for city environments than other routing protocols. Routing protocols like GPSR [5], GPCR [6], GSR [4], A-STAR [8], and GyTAR [24] work well in city environments. However, these protocols encounter different problems that motivate us to design a new robust scheme.

III. BACK-BONE-ASSISTED HOP GREEDY IN CITY VEHICULAR AD HOC NETWORKS

In this section, we present a position-based connectivity aware back-bone-assisted hop greedy (BAHG) routing protocol for VANET's city environments. The proposed routing protocol finds a routing path consisting of the minimum of intermediate intersections. The protocol is designed considering certain features in a city map, such as road segments, intersections, etc. To maintain connectivity at the intersections and to detect void regions, we rely on a group of nodes called back-bone nodes. Basically, we adopt a request-reply scheme to obtain destination position, which is then used to compute the routing path. To avoid the impact of mobility on routing decisions, an update procedure is specifically designed to supervise the movement of source as well as destination. Overall, the objective of the hop greedy routing algorithm is to reduce the hop count, which ultimately reduces the end-to-end delay. In addition, the protocol also ensures successful delivery of data packets to the destinations.

A. Assumption

It is assumed that each vehicle is aware of its position through the GPS. Moreover, it is assumed that GPS errors are minimized by various standard procedures like augmentation, precise monitoring, timekeeping, and carrier phase tracking. In addition to that, these nodes are equipped with a digital map and a navigation system.

B. System Design

In our system, intersections, major roads, and minor roads are assigned unique IDs.

1. Back-Bone Nodes and Connectivity Preservation

Connectivity is the key requirement for any routing protocol for reliable and fast delivery of packets. This section describes mechanisms to ensure connectivity of a routing path. A routing path involves many intermediate intersections at which the packet direction is changed. Selection of a wrong intermediate intersection may result in the dropping of packets. Similarly, if the source or destination changes its original position, the ongoing communication may get disrupted. Apart from this, the high mobility of vehicles may create temporary void regions on a road segment. As a result, routing paths passing through such road segments are seriously impaired. In our approach, we allow some of the nodes to take care of the foregoing connectivity issues. Such nodes are called as back-bone nodes. Based on the specific action they perform, they are classified into back-bone nodes at intersection and back-bone nodes at road segments.

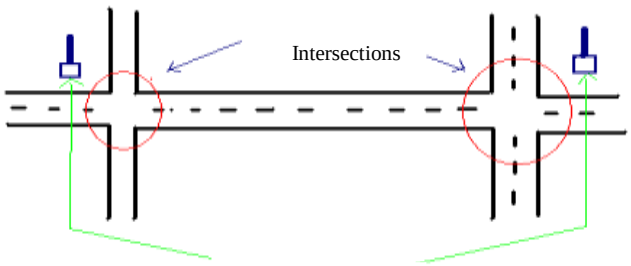


Fig. 1 Road Map indicating Back-Bone Nodes and Intersections.

1.1 Back-bone nodes at intersection

These nodes are used to maintain connectivity at an intersection. It is necessary for a back-bone node to declare its presence as soon as it enters the intersection region. For this purpose, the periodic beacons cannot be used because the beacon interval might be larger than the duration of stay of a node at an intersection. To overcome this issue, back-bone nodes use positional beacons, as described in [25].

1.2 Back-bone setup

Back-bone nodes of this kind are of three types, namely, stable, primary, and secondary back bones. A stable back-bone node is selected from the stream of vehicles waiting at the intersection during red traffic signal. Among the waiting vehicles, the vehicle closest to the intersection declares itself as the stable back bone. However, primary and secondary back bones are selected from the fleet of vehicles crossing the intersection when the signal turns green. The primary back bone is the one located at the intersection, whereas the secondary back bone is outside the intersection. Initially, a random node declares itself as the primary back bone. Then, the primary backbone node selects a secondary back-bone node comparing the average vehicle speed, the position, and the moving direction of all its neighbours. When the current primary back-bone node leaves the intersection region, it notifies the secondary back bone to become the new primary back bone. This notification also informs vehicles at or around the intersection about the new primary back bone.

1.3 Packet forwarding

When there is a need to choose a forwarding node from an intersection, a back-bone node is always preferred. This is because back-bone nodes can maintain the communication history and store packet in the absence of a forwarder at the intersection. A forwarding node checks its neighbour list to probe the available back-bone nodes. It compares the packet forwarding time with the staying time of each back-bone node. If the forwarding node is moving, it prefers stable back-bone nodes as the forwarder. Otherwise, it prefers the moving back bones (i.e., primary and secondary). The primary back bone has higher priority over the secondary back bone. Among the stable back bones, the back bone closest to the intersection has the highest priority.

1.4 Message queuing and retrieval

The stable back-bone nodes take the responsibility of packet buffering. In the absence of a suitable forwarding node, the packet is stored in a stable back-bone node. On availability of a forwarding node in the desired direction, packet is retrieved and forwarded. The stable back-bone nodes maintain the database of all communications with a timestamp. They store source and destination addresses along with the time of arrival of packets. If a similar packet arrives with a new timestamp, the previous database information is updated. While a packet is being routed along the selected path, either destination or source may change its position and moves to a new road segment. To allow back-bone nodes to keep track of the movements, both source and destination inform about their identity in their beacons. Whenever source or destination changes direction, the back-bone node updates the corresponding entry in its communication history. When a packet is being forwarded, the back-bone nodes provide the updated information. This enables a packet to be forwarded in the new direction. In Fig. 2, nodes B1, B2, B3, and B4 represent the back-bone nodes that take care of the activities at intersections.

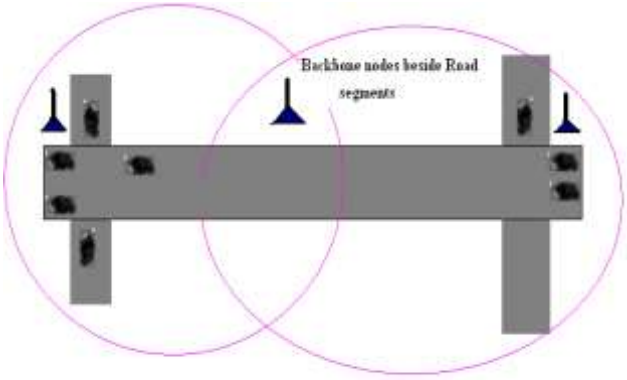


Fig. 2 Back-bone nodes engaged in void region detection and forwarding packets at intersections.

1.5 Back-bone nodes at road segment

If any part of a road segment longer than the transmission range is devoid of nodes, it can be noticed by the nodes present at the periphery of the void region. Nodes closest to the void region from both directions declare themselves as back-bone nodes. These backbone nodes are termed as "void-guard" back-bone nodes. The purpose of a "void-guard" back-bone node is to inform the presence of a void region to the neighbouring back-bone nodes stationed at intersections. For all such transactions among the back-bone nodes, a piggybacked beacon message is used. On being aware of an unconnected road segment, the back-bone node at the intersection prohibits packets from being forwarded to the identified road segment. In this case, the packet is forwarded by selecting a new route. In Fig. 2, nodes R1 andR2 are back-bone nodes of this type.

2. Hop Greedy Algorithm in Destination Reply and Data Dissemination

This section demonstrates how the reply message is forwarded to the source and the data packet is forwarded to the destination by adopting the hop greedy algorithm. On receiving a request message, the destination decides the reply path to the source using the hop greedy algorithm. The reply path is not necessarily the same as the path followed by the request message. The destination is aware of the source position; hence, a direct path to the source is computed without involving any boundary intersection. The reply message contains the list of intersections through which it has to traverse. If the source moves to a new position before the reply message is received, an update procedure discussed later renders a new path to the reply message. On receiving the reply message from the destination, the source transmits the data packets by computing a path to the destination adopting the same hop greedy algorithm. The update procedure also handles the change of position of the destination if the data packets do not locate it at the old position. In addition to that, the data packet carries the source position information. Such information enables the destination to decide whether to send a new reply message to the source if the destination has moved to a different zone.

3. BAHG Position Update

Before receiving the reply message, the source may change its position. Some back-bone nodes must be aware of the direction of the source movement. When a forwarder chosen among the back-bone nodes learns about such changes, it forwards the reply message toward the new direction. Ultimately, the source is able to receive the reply message. Likewise, the destination may change its position before receiving the data packet, and its movements are tracked by the back-bone nodes. The destination may move substantially far from its original position. In such cases, the hop count will be elevated if the packet is forwarded using the updates received from the back-bone nodes. Thus, a fresh reply message is forwarded to the source if the destination changes its zone. On receiving this reply message, the source can compute a better path to the destination. This can marginalize the hop count, irrespective of the destination movement.

IV. HOP GREEDY ROUTING ALGORITHM

In this section, we introduce the hop greedy routing algorithm that ensures the minimum number of hops from the source to the destination while considering connectivity. This algorithm is transformed to a single source minimum weight algorithm similar to [29]. The street map is converted into a graph G. All intersections as well as the source and the destination of the data packet are considered as the vertices. For two vertices u and v, there is an edge (u, v) in G if u and v are connected through a road segment. Each edge is associated with a weight that is the sum of two parameters: "hop count" and "delta count." Let R denote the transmission range. For an edge (u, v) of length L, the hop count is given by $(L/R) + 1$ if a direct line-of-sight path exists between u and v. However, for

a curved path, the hop count is equal to the possible number of small line-of-sight paths. The parameter delta count represents the degree of disconnection of an edge. It is evident that the higher the number of lanes, the higher the connectivity. We specify a threshold value for the number of lanes, and an edge having a number of lanes higher than the threshold is considered as a connected edge. Otherwise, it is necessary to determine the disconnection level.

Let *hc* and *dc* denote the hop count and delta count of a road segment. To determine the weight, the delta count can be normalized as $dc = \sigma * hc$. Note that $\sigma > 1$ as the delay of a disconnected edge is larger than the delay of a connected edge. We derive the value of σ in the next section. "S" denotes the source vertex from which a minimum weight path is sought to the destination vertex denoted as "D." The pseudocode of the main algorithm is shown in Fig. 6. All notations used in the algorithm are summarized in Table 1.

TABLE 1
SUMMARY OF NOTATIONS

NOTATION	DEFINITION
V[G]	Set of vertices of graph G
E[G]	Set of edges of graph G
Q	Min-Priority Queue formed using *V[G]*
b, c	Any two vertices in *V[G]*
Prev[b]	Predecessor of vertex *b*
Wt[b]	Weight of vertex *b*
T_Prev	Temporary vertex
T_Wt[b]	Temporary weight of a vertex *b*
VS	Set of vertices already visited
Adj[b] (*Adj:* Adjacency Matrix of Graph G)	Set of all vertices c such that there is an edge (*b, c*) in *E[G]*
LOS(*b, c*)	Returns true if vertex c are in line-of-sight or they share a common Road ID
HOP-COUNT (*b, c*)	Returns number of hops in the edge formed by vertex *b* and vertex *c*.
LANE (*c*)	Returns number of lanes connected to a vertex
th	Threshold for number of lanes connected at a vertex.
σ	Ratio of delta-count and hop-count
hc	Hop-count of a road segment
dc	Delta-count of a road segment

```
H – INITIALIZE (G, s)
1       for each vertex v ∈ V[G]
2       do Prev[v] ← Nil
3        Wt[v] ← ∞
4       Wt[s] ← 0
```

Fig. 3 Initialization Procedure

```
DELTA – COUNT (v, hc)
1       f ← LANE (v)
2       if  f  > th then return 0
3       else return σ * hc
```

Fig. 4 Procedure to evaluate connectivity (delta count)

```
H – RELAX (u, v)
1       k ← Prev[u]
2       LS ← LOS(k, v)
3       if k!= Nil AND LS = true
4             then T_Prev ← k
5       else T_Prev ← u
6       hc ← HOP – COUNT (T_Prev, v)
7       dc ← DELTA – COUNT (v, hc)
8       T_Wt[v] = Wt[T_Prev] + hc + dc
9       if Wt[v] > T_Wt[v]
10            then Wt[v] ← T_Wt[v]
11                 Prev[v] ← T_Prev
```

Fig. 5 Relax Procedure

```
H – DIJKSTRA (G, s)
1       H – INITIALIZE(G, s)
2       VS ← ϕ
3       Q ← V[G]
4       while Q!= ϕ
5            do u = EXTRACT – MIN(Q)
6            VS = VS U {u}
7            for each vertex v ∈ Adj[u]
8            do   H – RELAX(u, v)
```

Fig. 6 Hop Greedy Procedure

Fig. 3 to 6 represents the pseudo codes of our proposed algorithms.

The algorithm starts by initializing the weight of all vertices in *V* [*G*] using the initialization routine shown in Fig. 3. Initially, the weight estimate of each vertex is set to ∞, except for the source vertex, whose weight estimate is set to 0. The predecessor of each vertex denoted as *Prev*[*v*] is also initialized. In Fig. 6, line 2 shows initializing the set of visited vertices as an empty set, and in line 3, the min priority queue is populated with the vertices in *V* [*G*]. The vertices are keyed by their weight estimates. Starting with the source vertex, in each iteration, the vertex having the smallest weight is extracted from the min priority queue, and the weight estimates and predecessors of its neighbors are updated by executing the H-RELAX procedure, whose pseudo code is given in Fig. 5.

Suppose a vertex *u* executes H-RELAX to relax an edge (*u, v*). Suppose vertex *k* represents the predecessor of *u*. If *k* and *v* are in the line of sight or they share a common road ID (i.e., they are located on a same major road), then *k* will be chosen as the predecessor of *v*. Otherwise, *u* will be chosen as

the predecessor. The potential predecessor is stored in variable *T_Prev*. This step aims at minimizing the number of intermediate intersections along a path. In other words, the algorithm seeks to find the straightest possible path between source and destination. It is shown that selecting more number of intermediate intersections results in a longer path. The actual predecessor is decided only after the new weight of *v* is computed and compared against the stored value. Line 8 in Fig. 5 estimates the weight for *v* considering *T_Prev* as its immediate predecessor in the path from "S" to *v*. For the edge (*T_Prev*, *v*), the hop count is obtained as previously described, whereas the delta count is computed using the pseudocode given in Figs. 4 and 5. If the estimated weight of *v* is less than the stored weight, then *T_Prev* becomes the predecessor of *v*, and its weight components are updated. This way, the H-RELAX procedure processes all the neighbors of the vertex having the minimum weight. Note that the functions INSERT and DECREASE-KEY are implicit in lines 3 and 8 of Fig. 6, respectively. These two functions, along with the EXTRACT-MIN shown in line 5 of Fig. 6, are taken from [29].

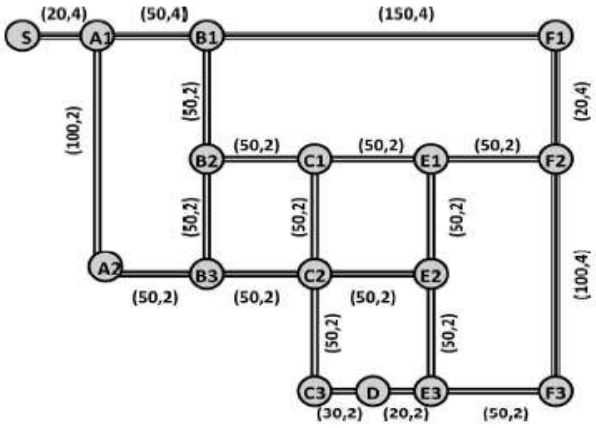


Fig. 7 Weighted Graph G with Edge Distance and Number of Lanes.

values that appear along all road segments in Fig. 7. As the weight of "S" is minimum, in the first iteration of the H-DIJKSTRA procedure, the H-RELAX procedure is called for all neighbors of "S." Here, "A1" is the only neighbor of "S." At "A1," the total number of lanes is equal to 4, and hence, the delta count is evaluated as 0. There is only one hop between "S" and "A1." The weight of "A1" is updated, and "S" is set as its predecessor vertex. In the second iteration, "A1" is extracted and executes H-RELAX for its neighbors "A2" and "B1." "B1" finds itself in the line of sight of "S," which is the predecessor of "A1." Hence, the hop count estimate of "B1" will be computed as the number of hops between "S" and "B1." Similarly, the delta count estimate is computed to be 0. "S" is set as the predecessor vertex of "B1." In a similar fashion, all the vertices are processed. Table II shows the final values of the predecessor, delta count, and hop count of each vertex after the execution of the hop greedy (H-DIJKSTRA) algorithm. From Table II, using the predecessor values, the path to be followed from "S" to "D" is determined as S → F1 → F3 → D. The algorithm follows the same time complexity and correctness pattern as the Dijkstra algorithm [29] by adding two weight parameters to form a single weight.

3. *Normalization of Delta Count*

To find the value of σ, we are interested in calculating the average delay incurred for an unconnected road segment. Let *R* denote the transmission range. We derive the average delay considering road segments of length less than *R*. For road segments having length greater than *R*, the delay calculation is repeated for each *R*. We define an unconnected road segment between two intersections I1 and I2 as follows. Since the length of the road segment is less than *R*, forwarders must be chosen from the intersections I1 and I2. However, if I2 does not have any vehicles, then the forwarder at I1 chooses the farthest vehicle moving on the road segment between I1 and I2 as the next forwarding node, which then carries the message until it reaches I2. We divide the transmission range into *w* cells, where each cell can occupy at most one vehicle. The length of a cell denoted as *l* is given by the sum of the average vehicle length and the minimum safety distance. *w* is a factor of *R* such that $w = R/l$.

TABLE 2
RESULT OF H- DIJKSTRA EXECUTION ON THE GRAPH IN FIG. 7

Vertex	S	A1	B1	F1	B2	B3	A2	F2	F3	C2	C1	E2	E1	E3	D	C3
Predecessor	NIL	S	S	S	B1	B1	A1	F1	F1	B3	F2	B3	F2	F3	F3	F3
Delta Count	0	0	0	0	σ	σ	σ	0	0	2σ	σ	2σ	σ	σ	σ	σ
Hop Count	0	1	1	1	2	2	2	2	2	3	3	3	3	3	3	3

In Fig. 7, the source vertex is "S," and the destination vertex is "D." The algorithm H-DIJKSTRA is executed to find the minimum weight path from the source "S" to the destination "D." The parameter *th* is set to 4. The distance and number of lanes of a road segment are represented as a pair of

The smaller the road segment, the lesser the delay incurred by the store and carry approach. Considering that vehicles are located near the edge of the cell, the smallest delay corresponds to the case when the road segment contains two cells, and the forwarding vehicle chosen from first cell

has to travel only one cell, whereas delay is largest when the road segment contains $w - 1$ cells. Let the vehicles be randomly distributed, and λ denotes the average vehicle density per R (R is normalized to 1). A cell locates a forwarding node only when it contains at least one vehicle and all cells farther from the sender than it are empty. Suppose the cells are indexed as 1, 2, 3, and so on. Then, for a road segment having k cells, the delay incurred by the store and carry approach is given by

$$D_k = \sum_{i=1}^{k} \left[\left(1 - e^{-\frac{\lambda}{\omega}}\right) * \prod_{j=i+1}^{k} e^{-\frac{\lambda}{\omega}} \right] * d_j \quad \longrightarrow (1)$$

where the expression $(1 - e^{-(\lambda/w)})$ represents the probability of selecting a forwarder in cell i. The variable d_j represents the time elapsed before the forwarder reaches the intersection and is given by

$$d_j = (k - i) * \frac{l}{V_{\text{av}}} \quad \longrightarrow (2)$$

where V_{av} represents the average velocity of the vehicle.
Then, the average delay considering all possible cases is obtained as

$$D_{\text{av}} = \frac{1}{\omega - 2} \sum_{k=2}^{\omega-1} \left[\frac{1}{q} \sum_{s=1}^{q} D_{s,k} \right] \quad \longrightarrow (3)$$

In (3), for a road segment having k cells, we choose q densities, which are denoted as λ_s, $s = 1, 2 \ldots q$, to find the delay $D_{s,k}$, which is evaluated using (1) by replacing $\lambda = \lambda_s$.
Note that $D_{s,k}$ decreases with the increase in density, and in fact, it remains zero after a certain density that indicates that the road segment is connected. Hence, q density values represent densities for which $D_{s,k} > 0$. According to the foregoing description of an unconnected road segment, the total delay consists of the transmission delay at one hop and the delay incurred by the store and carry approach. The packet transmission delay in one hop is expressed as

$$D_{\text{hop}} = T_{\text{Packet}} + T_{\text{Other}} \quad \longrightarrow (4)$$

where T_{Other} includes other delays associated with unicast packet transmission, as specified in IEEE 802.11b, such as interframe spaces, transmission time of RTS and CTS, propagation delay, back-off delay, and physical layer overhead. T_{Packet} represents the transmission time of data, request, or reply packets.

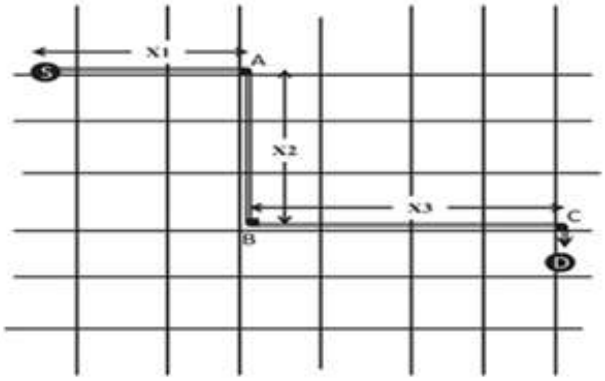
Let D_{conn} and D_{unconn} denote the delay incurred if the segment is connected and disconnected, respectively. D_{conn} is same as one hop delay D_{hop}. On the other hand, D_{unconn} is given by the sum of D_{av} and D_{hop}. Now, the value of σ is determined as

$$\sigma = \frac{D_{\text{unconn}}}{D_{\text{conn}}} = \frac{dc}{hc} \quad \longrightarrow (5)$$

Considering a minimum density of 10 vehicles per R, which is set to 300 m, D_{av} is calculated as 0.12264 s. For a data packet of size 512 B, σ is calculated to be 40.182.

4. Algorithm Evaluation

Fig. 8 shows a routing path from the source "S" to the destination "D." The intersections where the routing path changes directions are considered as intermediate intersections (i.e., A, B, and C). In the following evaluation, we refer to the group of road segments between two intermediate intersections as a single road segment.



Routing Path S → A → B → C → D
Fig. 8 Routing path and Road segment

If an intersection does not have any forwarder, the back-bone nodes store the request message until a node arrives at the intersection.

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the BAHG protocol using an ns-2.35 simulator [23]. A city traffic scenario is considered to demonstrate the protocol performance.

5.1. Simulation Environment

The simulation scenario is shown in Fig. 12. We used the open- source NS-2 simulator to generate vehicle movements. Vehicles belong to one of seven vehicle types, where each type is represented by certain maximum velocity (ranges from 5 to 35 m/s). A total of 11 moving vehicles, 10 Road Side Units and 1 Back-bone node are generated by the simulator. Once generated, they start moving along the specified route. We considered randomly selected src–dst pair that begins to communicate and remain active until the end of the simulation. Simulation is conducted for multiple scenarios represented by different vehicular movements, and the outcomes are averaged to obtain the performance graphs.

TABLE 3
SIMULATION PARAMETERS

Parameters	Values
Area 200mX200m	200X200m
Channel	WirelessChannel
Propagation Model	TwoRayGround
Antenna	OmniAntenna

Ad hoc Routing protocol	AODV
Number of nodes	32
Simulation Time	20ms and 100ms
Data Rate	10Mbps
Vehicle Speed	5 – 35 m/sec
Pkt. Gen. Rate (CBR)	0.5 – 5 packets/msec
Transmission Range (Node)	100 meters

5.2 Routing Metrics

1) Packet delivery ratio (%): This is the ratio of the total number of packets received at the destination to the total number of packets generated by the source.

2) End-to-end(Revocation) delay: This is the delay elapsed between packet generation at the source and successful reception at the destination.

5.3 Simulation Results and Analysis

In this section, the packet delivery ratio, packet loss ratio, authentication delay and end-to-end(revocation) delay are investigated with respect to simulation time and number of packets. In the simulations, the initial src–dst distance implies the distance between the source and the destination when the source generates the first packet. The second factor destination dislocation distance refers to the distance travelled by the destination during the entire communication period.

Fig 9 shows neighbouring nodes and their distances from the current node and also distance from one node to all other movable nodes in the simulation environment.

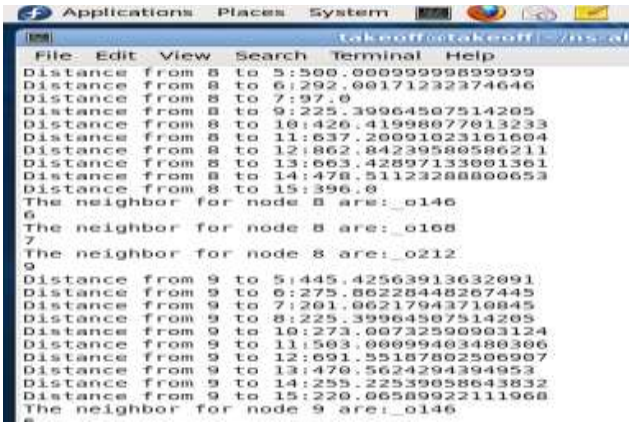


Fig 9 Snapshot showing distances of neighbouring nodes from current node.

Fig 10 Snapshot showing intermediate nodes between source and destination nodes.

Snapshot in fig. 10 shows routing from one node to the other node. While finding shortest routes from source to destination, it also lists the intermediate nodes through which routing takes place.

Snapshot in fig. 11 shows how vehicle to vehicle communication begins. Each vehicle is given with a vehicle id (node id). The vehicle which wants to send a message (or data) is treated as source vehicle and the vehicle which receives the message is treated as destination vehicle. Both source and destination vehicle ids have to be defined before the routing takes place.

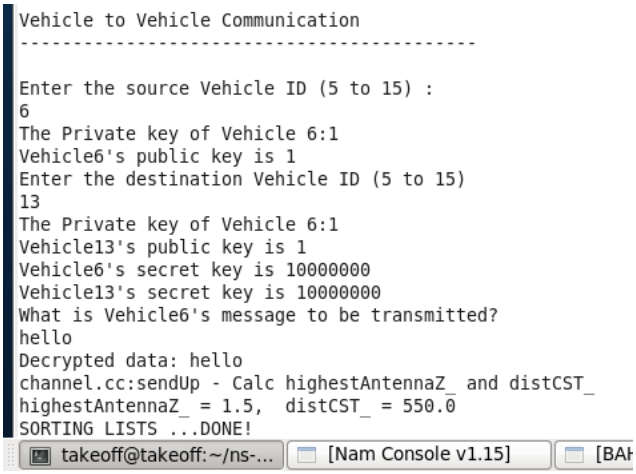


Fig. 11 Snapshot showing vehicle to vehicle communication.

$ wish main.tcl

On successful execution of the above command, an ***out.nam*** network animator file as shown in fig. 12 opens up. Fig. 13 gives trace file of out.nam file called as ***out.tr***, in which we can find out the list of events occurred when the simulation started till its completion.

In the out.tr file, we can see the presence of request-reply signals like ***RTS*** and ***CTS.***

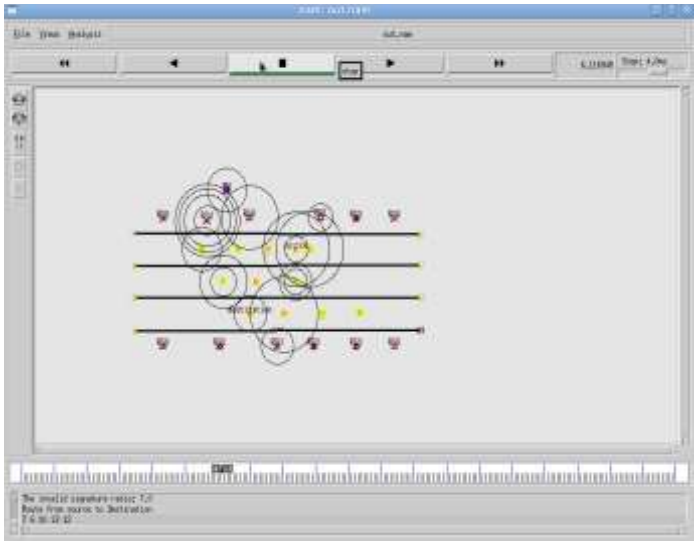


Fig. 12 Snapshot of Network Animator file showing different lanes and nodes.

Fig. 14 to fig. 18 shows snapshots of various graphs like Packet Delivery Ratio, Packet Loss Ratio, End-to-end delay and Authentication Delay. Description of the fig. 14 to fig. 18 are given in the below section.

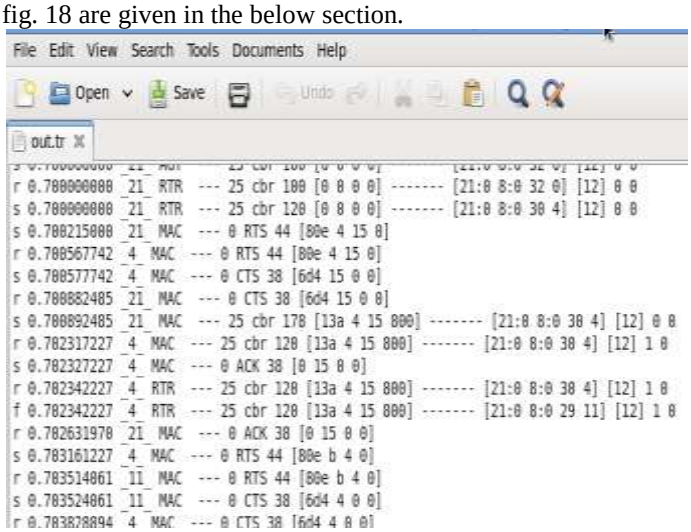


Fig. 13. Snapshot of Trace file showing the presence of RTS and CTS (Request and Reply) signals.

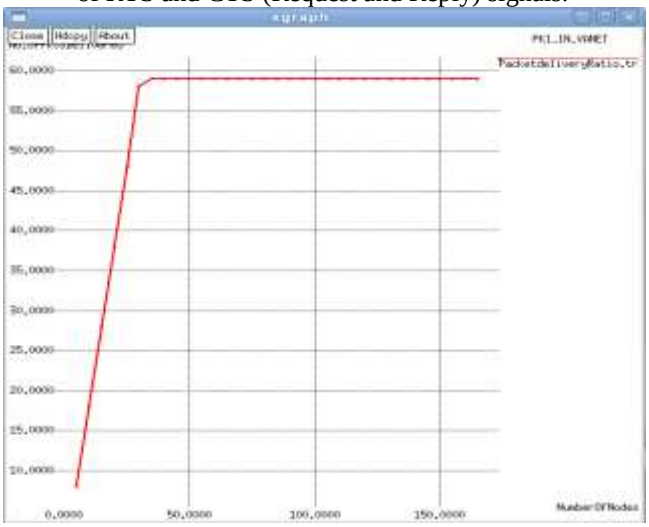


Fig. 14 Snapshot showing Packet Delivery Ratio graph.

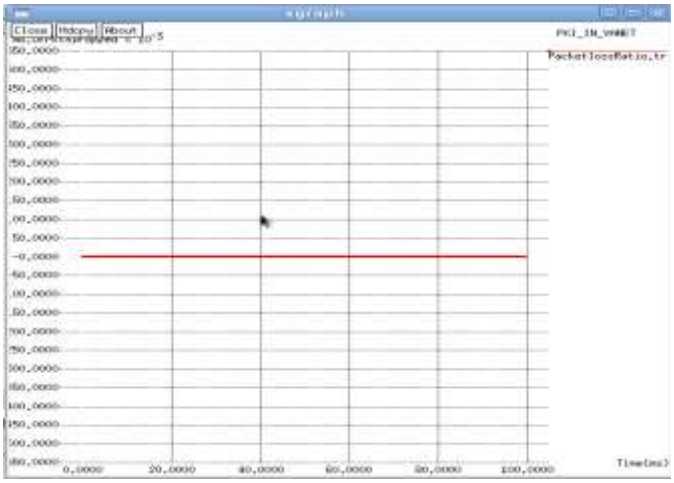


Fig. 15 Snapshot showing Packet Loss Ratio graph.

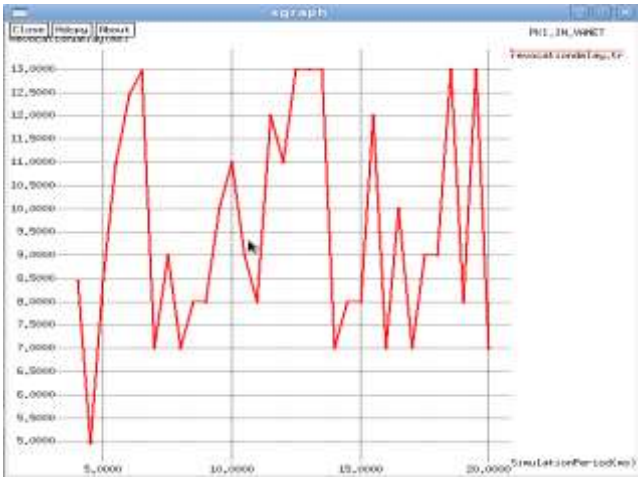


Fig. 16 Snapshot showing graph of End-to-end delay for 20ms.

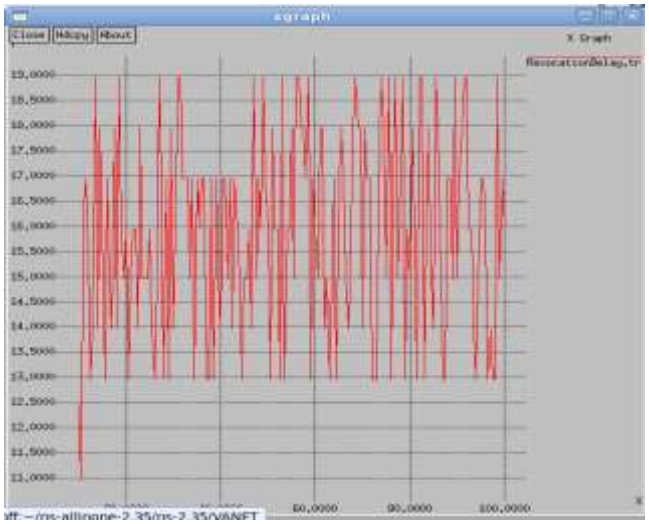


Fig. 17 Snapshot showing graph of End-to-end delay for 100ms.

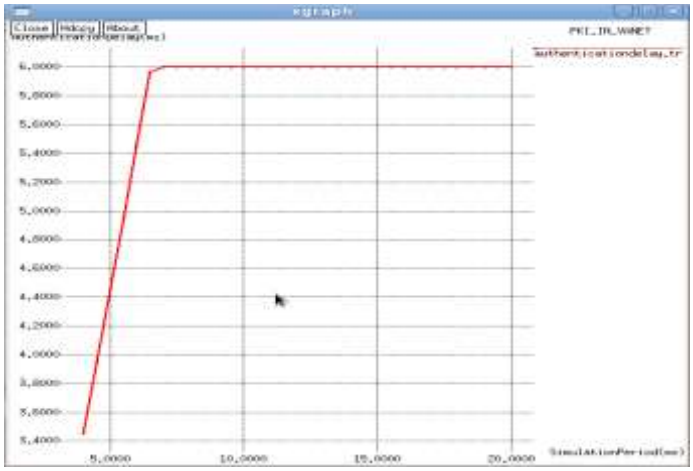


Fig. 18 Snapshot showing graph of Authentication delay.

4.3.1 End-to End (Revocation) Delay

Fig. 16 and 17 shows the average end -to- end delay with respect to simulation duration. Here, every source sends one packet per millisecond. It is observed that the end-to-end delay BAHG shows the least variation. In city scenarios, the distance between two adjacent intersections is much smaller than the transmission range of a node. Then, the routing path may involve numerous intersections, and the hop count increases and contributes to the end-to-end delay. The BAHG protocol incurs minimum delay as the source can predict the hop count before actually sending any data packet. The lowest hop count path translates to the minimum end-to-end delay path, which is evident from Fig. 16 and 17.

It is noticed that, with the increase in distance between the source and the destination, the end-to-end delay increases. The initial src–dst distance also has an impact on the location service, which gradually becomes ineffective and takes longer to provide the destination position information. Moreover, the hop count increases with the increase in distance from the source to the destination and is one of the reasons behind the increase in end-to-end delay.

As far as BAHG is concerned, the routing path is selected considering the minimum number of hops. It directly follows that fewer number of intersections are involved to forward a packet from the source to the destination, keeping the end-to end delay as low as possible.

When the destination moves away from its current position, the location-service based protocol uses the updated position along with the move. Therefore, the packet intended for the old position is redirected to the new position. If a new path is sought when a data packet is already halfway, the hop count will be elevated, and it will lead to a longer delay. For longer destination dislocation, the delay will be larger for location-assisted protocols like GPCR and GyTAR. This is due to the swinging movement of packets. However, BAHG is not affected by such phenomenon, because short movements of destination are managed by the back-bone nodes. In addition, whenever the destination changes its zone (i.e., longer movements), the updated position information of the destination is supplied to the source in the form of a reply message. Therefore, BAHG achieves the lowest end-to-end delay, which remains low across all dislocation distances.

4.3.2 Packet Delivery Ratio and Packet Loss Ratio

In Fig. 14, the packet delivery ratio is measured with varying simulation period. One common factor responsible for this drop is the dependence on intersection node for packet forwarding. An increase in the src–dst distance reduces the chance of finding an intersection node. BAHG does not suffer from the so-called intersection node probing problem as it avoids forwarding through intersections. The outcome is expected as BAHG makes minimal use of intersections. However, the update mechanism of the BAHG protocol with the help of back-bone nodes enables the source to be connected with the destination through a reliable path. Therefore, the packet delivery ratio of BAHG is maintained high, irrespective of the destination dislocation distance. From fig. 15, we found that, Packet Loss Ratio is almost zero because it makes use of Back-bone node and Road Side Units. Fig. 18 shows Authentication Delay w.r.t simulation time. The time taken to authenticate a particular message from valid source to valid destination is less. Authentication provides security to the transmitted message.

V. CONCLUSION

In this paper, we conclude that hop greedy routing protocol reduces end-to-end delay by yielding a routing path that includes the minimum number of intermediate intersections. The zone wise partitioning of a city road network is an important design framework for the efficient functioning of the destination discovery procedure. The hop greedy algorithm finds the best possible path in terms of both hop count and connectivity. The concept of back-bone node is used to address connectivity issues in void/ sparse regions because of unavailability of forwarder nodes. Moreover, by employing unicast request messages, the proposed routing scheme eliminates packet loss and congestion noticed in contemporary routing protocols that use broadcast request messages. Destination movements are frequently updated. The simulation results confirmed that the packet generation rate, the distance between the source and the destination, and the distance of destination movement do not have a large impact on the performance of the proposed scheme in terms of packet delivery ratio and end-to-end delay.

ACKNOWLEDGMENT

Special Thanks to Prof. J. V. Vadavi, HOD, Dept. Of Computer Science, Dr. S. B. Vanakudre, Principal, SDMCET, Dharwad for providing necessary facilities to carry out the work. We extend our heartily acknowledgement to Dr. S. V. Gorabal and Prof. Santosh Bujari for the technical assistance.

REFERENCES

[1] J. Bernsern and D. Manivannan, “Unicast routing protocols for vehicular ad hoc networks: A critical comparison and classification,” Pervasive Mob. Comput., vol. 5, no. 1, pp. 1–18, Feb. 2009.

[2] Q. Yang, A. Lim, S. Li, J. Fang, and P. Agrawal, “ACAR: Adaptive connectivity aware routing for vehicular ad hoc networks in city scenarios,” Mob. Netw. Appl., vol. 15, no. 1, pp. 36–60, Feb. 2010.

[3] V. Naumov and T. R. Gross, “Connectivity-aware routing (CAR) in vehicular ad hoc networks,” in Proc. IEEE INFOCOMM, 2007, pp. 1919–1927.

[4] C. Lochert, H. Hartenstein, J. Tian, H. Füßler, D. Hermann, and M. Mauve, “A routing strategy for vehicular ad hoc networks in city environments,” in Proc. IEEE Intell. Veh. Symp., 2003, pp. 156–161.

[5] B. Karp and H. T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in Proc. ACM MOBICOM, 2000, pp. 243–254.

[6] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, “Geographic routing in city scenarios,” ACM SIGMOBILE Mobile Comput. Commun. Rev., vol. 9, no. 1, pp. 69–72, Jan. 2005.

[7] H. Menouar, M. Lenardi, and F. Filali, “Movement prediction-based routing (MOPR) concept for position-based routing in vehicular networks,” in Proc IEEE VTC, 2007, pp. 2101–2105.

[8] G. Liu, B. S. Lee, B. C. Seet, C. H. Foh, K. J. Wong, and K. K. Lee, “A routing strategy for metropolis vehicular communications,” in Proc. ICOIN, LNCS, Aug. 2004, pp. 134–143.

[9] C. C. Hung, H. Chan, and E. H. K. Wu, “Mobility pattern aware routing for heterogeneous vehicular networks,” in Proc. IEEE WCNC, 2008, pp. 2200–2205.

[10] K. C. Lee, J. Häerri, U. Lee, and M. Gerla, "Enhanced perimeter routing for geographic forwarding protocols in urban vehicular scenarios," in Proc. IEEE GlobeCom Workshops, 2007, pp. 1–10.
[11] W. Kieß, H. Füßler, and J. Widmer, "Hierarchical location service for mobile ad-hoc networks," ACM SIGMOBILE Mob. Comput. Commun. Rev., vol. 8, no. 4, pp. 47–58, Oct. 2004.
[12] M. Käsemann, H. Füßler, H. Hartenstein, and M. Mauve, "A reactive location service for mobile ad hoc networks," Dept. Comput. Sci., Univ. Mannheim, Mannheim, Germany, Tech. Rep. TR-14-2002, Nov. 2002.
[13] X. Jiang and T. Camp, "An efficient location server for an ad hoc networks," Colorado School Mines, Golden, CO, Tech. Rep., MCS-03-06, May 2003.
[14] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "Ascalable location service for geographic ad hoc routing," in Proc. ACM MOBICOM, 2000, pp. 120–130.
[15] J. Zhao and G. Cao, "VADD: Vehicle-assisted data delivery in vehicular ad hoc networks," IEEE Trans. Veh. Technol., vol. 57, no. 3, pp. 1910– 1922, May 2008.
[16] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks," in Ad Hoc Networking, C. E. Perkins, Ed. Reading, MA: Addison-Wesley, 2001, ch. 5.
[17] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in Proc. 2nd IEEE Workshop Mob. Comput. Syst. Appl., 1999, pp. 90–100.
[18] H. Menouar, M. Lenardi, and F. Filali, "Improving proactive routing in VANETs with the MOPR movement prediction framework," in Proc. ITST, 2007, pp. 1–6.
[19] H. Menouar, M. Lenardi, and F. Filali, "A movement prediction-based routing protocol for vehicle-to-vehicle communications," in Proc. 1st Int. V2VCOM, San Diego, CA, Jul. 2005.
[20] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in Proc. ACM/IEEE MOBICOM, 1999, pp. 151–162.
[21] Y. Ding, C. Wang, and L. Xiao, "A static-node assisted adaptive routing protocol in vehicular networks," in Proc. ACM VANET, 2007, pp. 59–68.
[22] H. Füßler, M. Mauve, H. Hartenstein, and D. Vollmer, "A comparison of routing strategies for vehicular ad-hoc networks," Dept. Comput. Sci., Univ. Mannheim,Mannheim, Germany, Tech. Rep. TR-02-003, Jul. 2002.
[23] The Network Simulator-ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/
[24] M. Jerbi, S. M. Senouci, T. Rasheed, and Y. Ghamri-Doudane, "Towards efficient geographic routing in urban vehicular networks," IEEE Trans. Veh. Technol., vol. 58, no. 9, pp. 5048–5059, Nov. 2009.
[25] P. K. Sahu, E. H. Wu, J. Sahoo, and M. Gerla, "DDOR: Destination discovery oriented routing in highway/freeway VANETs," in Springer Telecommun. Syst.—Special Issue Vehicular Communications, Networks, Applications, 2010, pp. 1–18.
[26] U. Lee, J. Lee, J. S. Park, and M. Gerla, "FleaNet: A virtual market place on vehicular networks," IEEE Trans. Veh. Technol., vol. 59, no. 1, pp. 344–355, Jan. 2010.
[27] The CitySense Sensor Network Project. [Online]. Available: http://www.citysense.net
[28] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea, "VANET routing on city roads using real-time vehicular traffic information," IEEE Trans. Veh. Technol., vol. 58, no. 7, pp. 3609–3626, Sep. 2008.
[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 24.3: Dijkstra's algorithm," in Introduction to Algorithms, 2nd ed. Cambridge, MA: MIT Press, 2001, pp. 595–601.
[30] N.Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz, "Routing in sparse vehicular ad hoc wireless networks," IEEE J. Sel. Areas Commun., vol. 25, no. 8, pp. 1538–1556, Oct. 2007.
[31] M. M. Artimy, W. Robertson, and W. J. Phillips, "Connectivity in inter-vehicle ad hoc networks," in Proc. IEEE CCECE, May 2004, pp. 293–298.
[32] Q. Song and X. Wang, "Efficient routing on large road networks using hierarchical communities," IEEE Trans. Intell. Transp. Syst., vol. 12, no. 1, pp. 132–140, Mar. 2011.
[33] Z. C. Taysi and A. G. Yavuz, "Routing protocols for GeoNet: A survey," IEEE Trans. Intell. Transp. Syst., vol. 13, no. 2, pp. 939–954, Jun. 2012.