Research Article

Distributed Algorithm for Constructing Efficient Tree Topology for Message Dissemination in Vehicular Networks

S. Kamakshi and Sairam Natarajan

School of Computing, SASTRA University, Thanjavur, Tamil Nadu 613401, India

Correspondence should be addressed to S. Kamakshi; kamakshi@cse.sastra.edu

Received 24 August 2013; Revised 5 January 2014; Accepted 9 January 2014; Published 20 February 2014

Academic Editor: C. S. Shankar Ram

Copyright © 2014 S. Kamakshi and S. Natarajan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Vehicular ad hoc network is an ad hoc network constituted among moving vehicles that have wireless dedicated short range communication (DSRC) devices in order to provide ubiquitous connectivity even if the road-side infrastructure is unavailable. Message dissemination in vehicular ad hoc networks is necessary for exchanging information about prevailing traffic information, so that the vehicles can take alternate routes to avoid traffic jam. A major challenge in broadcast protocols is that they result in flooding of messages that reduce the speed of dissemination due to collision. Dissemination of messages using tree topology reduces the number of rebroadcasts. Dynamicity Aware Graph Relabeling System model provides a framework to construct spanning tree in mobile wireless network. In this paper, we propose a new distributed algorithm for constructing an arbitrary spanning tree based on Dynamicity Aware Graph Relabeling System model, which develops a maximum leaf spanning tree in order to reduce the number of rebroadcasts and dissemination time. Our simulation results prove that the number of vehicles rebroadcasting the message is curtailed to 15% and the dissemination time required to achieve 100% reachability is curtailed by 10% under average vehicle density.

1. Introduction

Vehicular Ad hoc Networks (VANETs) are helpful in implementing Intelligent Transport Systems (ITS) that improve the safety of the transportation as well as provide comfort to the drivers by giving valuable on-time suggestions regarding the current traffic. Some applications developed for the safety of vehicle users are Traffic Signal Violation Warning, Curve Speed Warning, Cooperative Forward Collision Warning, Pre-crash Warning, Intersection Collision Warning, and so forth [1]. All these road safety applications require information dissemination to all vehicles moving towards the zone. The early notification of messages enables the drivers to make cautious and proper decisions in advance regarding the change in driving directions so that traffic jam and accidents may be avoided. Also it reduces waiting time in traffic jam thereby reducing the fuel consumption. Broadcasting protocols serve the purpose of dissemination of messages to all the vehicles as much as possible. But if every vehicle that receives a new message rebroadcasts it again, simple flooding may lead to exponential growth

in the number of messages floating in the network [2]. This results in congestion in the network. The major challenge in disseminating these messages is to reduce the number of rebroadcasts in the network. Several improvements in the broadcasting protocols are devised to reduce the number of rebroadcasts. Many of these protocols calculate a probability for rebroadcasting based on the information known to them [3, 4]. Some of these protocols are position based which assumes that all the vehicles know exact positions either by using GPS or their relative positions by using received signal strength [5]. Also several heuristics are employed to forward the messages to the zone of relevance [6]. Dissemination of messages using tree topology is a mechanism for reducing the number of rebroadcasts to a minimum possible value [7, 8]. Since awareness of mobility at algorithmic level will improve the performance of the algorithm, Dynamicity Aware Graph Relabeling System (DA-GRS) model is adopted [9]. Use of centralized algorithm for constructing and maintaining the spanning tree reduces scalability. Also it is not possible to store and update the entire spanning tree in all the vehicles, since movement of the vehicles make frequent changes in the

topology. Hence, a new distributed algorithm is devised for constructing and maintaining the spanning tree.

In this paper, we propose a new distributed algorithm for constructing a maximum leaf spanning tree based on DA-GRS model by making maximum number of nodes as leaf nodes in order to reduce number of rebroadcasts and dissemination time. Our algorithm needs each vehicle to store information about its adjacent vehicles only. Since each vehicle stores only a partial view of the spanning tree, it requires only less storage space and the algorithm is scalable. By reducing the number of internal nodes of the spanning tree, the number of rebroadcasts has been minimized. This also reduces the bandwidth consumption required for spanning tree construction and maintenance. Hence, effective bandwidth utilization is increased and overhead cost is reduced.

The rest of the paper is organized as follows. Section 2 presents the related works. Section 3 provides details of the DA-GRS model, the description about the data structure used, description of the messages used in our algorithm, behavior of our proposed protocol, proof of our claim, and the pseudocode of our proposed protocol. Section 4 presents our simulation results and Section 5 concludes our paper.

2. Related Work

Researchers took lot of efforts in reducing the broadcast storm problem, reducing the latency and overhead in dissemination of messages, and increasing the coverage and reachability of vehicles in VANETs. To achieve these, the improvements are done in link layer or network layer by using the information available from the physical layer. For reducing latency in message dissemination, Doukha et al. devised a protocol that combines message dissemination in unicast and broadcast modes in VANETs. and also for reducing collision of data packets with beacon packets, they used the Enhanced Distributed Channel Access (EDCA) mechanism of IEEE 802.11e, which gives more priority in transmitting data packets than the beacon packets [10]. To reduce the broadcast storm problem Ghafoor et al. presented a protocol for forwarding packets in urban vehicular environments that elects next relay node at each road junction based on distance to destination, signal strength, and direction of movement [11]. To reduce the bandwidth consumption and broadcast storm problem in high density networks, Hogue et al. in [12] described a protocol that delays the rebroadcast using the probability computed based on the neighborhood knowledge. But each node needs to append the list of its one-hop neighbor IDs in every message for the calculation of the probability, which increases the overhead. In order to reduce this overhead in obtaining the neighbor information for the calculation of probability, Mylonas et al. computed the probability for rebroadcasting based on the speed of the vehicle which is proportional to the vehicle density [3].

In MANETs, since the devices work in battery power, the major criteria for selection of routes should consider the remaining battery level in the device. Hence in the protocol developed by Kalpana and Punithavalli [13], for routing in

mobile ad hoc networks, they select forwarding nodes based on the node's battery lifetime and the distance among them. In order to ensure reliability they used the rebroadcasting of the nodes as acknowledgements for the receipt of the message. But in VANETs the power may be generated from the fuel and the movement of the vehicle. Hence, the battery level is not a major criterion in VANETs.

Bür and Kihl analyzed the performance of the information dissemination algorithms. They studied the behavior of naive broadcast, intelligent broadcast, modified intelligent broadcast, and epidemic broadcast [14]. In naive broadcast whenever the vehicle hears a warning message, it rebroadcast the message immediately. In intelligent broadcast, the repeaters (vehicles) wait for some random amount of time before they rebroadcast. During this wait time if it hears the same message from the vehicles behind, it will not rebroadcast the message. In modified intelligent broadcast algorithm, the waiting time is calculated based on distance so that the vehicles that are farther away will rebroadcast earlier than others. Hence, it provides full coverage in fewer steps. In epidemic broadcast, the wait time is randomly chosen from an interval of 0 to $t_{\max}$ where $t_{\max}$ is inversely proportional to distance.

Tonguz et al. present a completely distributed broadcasting protocol, named DV-CAST, for vehicular networks based only on local information [15]. The protocol operates in two modes: store-carry-forward mode and broadcast suppression mode. In sparse network, when the vehicle is not connected to any other vehicle, it stores the received message and carries the message till it connects to another vehicle and then forwards the message. This mode addresses the disconnected network. When the protocol is in connected network, that is, in dense network, it switches over to broadcast suppression mode and uses the weighted p-persistent scheme [4], a probability-based approach, to handle the broadcast storm problem. The vehicle calculates the forwarding probability based on the distance and its transmission range.

In [9], Casteigts and Chaumette presented a model, called DA-GRS, for constructing a noncyclic undirected graph for dynamicity aware mobile ad hoc networks. The nodes are aware of the events that happen in their neighborhood and respond to those events by labeling their edges accordingly. Ruiz et al. [7, 8] used the DA-GRS model for constructing the spanning tree and established that the tree topology constructed in Mobile Ad hoc Networks (MANET) and the VANETs have same link characteristics and hence tree topology can be used for message dissemination in VANETs also. They devised a parameter-less broadcasting protocol, named BODYF, for dissemination of messages in dynamic spanning forest. They have not concentrated on the construction of spanning forest and gave only an outline of construction of such forest using DA-GRS model.

3. Spanning Tree Construction

VANET may be modeled as a digraph $G = (V, E)$, where vehicles are represented as the set of vertices (nodes) V , and the communication links between vehicles are represented as

the set of edges E . If a vehicle B is within the communication range of vehicle A , then an edge $\langle A, B \rangle$ is included between them. The vehicle knows about the other vehicles in its neighborhood by periodic broadcast of beacon messages which includes its public identity. Upon receiving these broadcast messages, each vehicle stores the information about the other vehicles in its neighborhood.

3.1. DA-GRS Model Description. Dynamicity Aware Graph Relabeling System (DA-GRS) is a distributed model for constructing tree topology in dynamic networks [9]. It is a self-stabilizing model that considers mobility at algorithmic level. The nodes are said to be aware of the changes in their neighborhood and execute the algorithm when they see a change such as appearance or disappearance of nodes or communication links (inclusion or removal of nodes or edges) with their neighbors. They construct a noncyclic undirected spanning tree with each node connecting to at least k other nodes based on the occurrences of events like appearance and disappearance of nodes and edges. An edge has two labels: *value* and *activity*, where *value* can be 0 or 1 and *activity* can be *on* or *off*. A node has two labels: *state* and *counter*, where *state* may be F (*integrated*) or N (*not integrated*) and *counter* is incremented for each additional link included in the tree. The nodes react to the changes in the topology in the following way.

Rule 1. When a new node A in state F , that has counter less than k , detects another node B in its neighborhood, an edge is included between the two nodes; the edge's *value* label in both nodes A and B are set as 1, the *activity* label in both nodes A and B are set as *on*, and their *counter* values are incremented.

Rule 2. When a node A in state F loses one of its communication links (disconnection of a link), the corresponding edge is removed, the activity label is set as *off*, value is set to 0, and the *counter* is decremented.

Rule 3. When a node A in state F loses its last communication link, its state is set as N and *counter* is set to 0.

Rule 4. When a node A in state N comes within the communication range of node B in state F , an edge is included between the two nodes, the edge's *value* label in both nodes A and B are set as 1, the *activity* labels in both nodes A and B are set as *on*, and their *counter* values are incremented. Also the state of A is set to F .

3.2. Data Structure Description. Like the algorithm is distributed, the data structure used to represent the network is also distributed. Each node (vehicle) stores the information about the nodes (vehicles) that are in its communication range, which provides partial view about the network. Hence, the storage requirement is less. Each node has a unique publicly known identity called UID. As in DA-GRS model, each node has three properties, namely, Node-status, Token-status, and parent. Also each node maintains a list of edges

corresponding to the nodes present in its neighborhood (i.e., communication range). The Node-status property may be either "*Integrated*" or "*Not Integrated*." Initially, the node status is set as "*Not Integrated*." When a node becomes part of the spanning tree, its status is changed into "*Integrated*." When a node becomes isolated from the spanning tree (when its node degree becomes zero), its status is again reverted to "*Not Integrated*." The Token-status may be either *true* or *false*, representing whether it is currently possessing token or not. The parent property is updated during circulation of the token in order to keep track of the node from which it received the token. Token is implemented as an object moving around in the network. The token object is sent from one node to another in the form of messages. A node possessing the token sets its Token-status property as *true*. Upon sending or deleting the token this property is reset to *false*. For each edge in the list maintained by a node, it is labeled as either 0 when it is not part of the spanning tree, or 1 when it is included in the spanning tree.

3.3. Message Description. There are six types of messages used in the protocol for constructing and maintaining the spanning tree: Beacon message, Token message, Any Token message, I Have Token message, Delete Token message, and Token Deleted message. "Beacon message" is used to announce the existence of the node (vehicle). It contains the UID of the node which is unique for every node. The beacon messages are transmitted periodically. In order to avoid the collision of the beacon messages and the broadcast storm problem [2], the beacon interval is adjusted based on the current traffic density. The traffic density is estimated from the speed of the node itself [3]. If the speed of the node is between 25 km/h and 40 km/h, it implies that the traffic density is high (dense traffic) and hence the beacon messages are transmitted once in every 300 ms. If the speed of the node is between 40 km/h and 75 km/h, it implies that the traffic density is moderate (average traffic) and hence the beacon messages are transmitted once in every 150 ms. If the speed of the node is between 75 km/h and 100 km/h, it implies that the traffic density is low (sparse traffic) and hence the beacon messages are transmitted once in every 100 ms. Also in dense traffic the change in neighborhood will be less when compared to the sparse traffic, where the change in neighborhood will be high due to the speed of the vehicle.

The "Token message" contains the token object and is used to circulate the token object in the spanning tree. Whenever a node receives a Token message, it becomes the owner of that token object. The node possessing a token object can only send an "Any token" message. The "Any token message" is used to initiate the spanning tree construction/merge process. It contains the UID of the node, its direction of movement (N/S/E/W), and its speed of movement. The "I Have Token message" is used to respond for Any Token message. It also contains the UID of the node, its degree, its direction of movement, and its speed of movement. The "Delete Token message" is used to delete the token of the sub trees that are merged together, except one of the sub trees which is currently sending the message.

The “Token Deleted message” is sent by the nodes that received the “Delete Token message” after the deletion of the token and is used to confirm the deletion of the token by the subtrees. The safety message, which carries the information about the incident (accident, traffic jam, etc.), contains the time stamp at which the message is generated and the type of the message or duration of validity of the message in order to decide whether the message needs to be forwarded further down in the spanning tree. Also when a safety message is received for the first time, it is propagated further down the tree. If it is a duplicate message, it just discards that message [14].

3.4. Protocol Description. Initially all nodes are considered as forest of spanning trees consisting of isolated nodes and every node possesses a token. The nodes periodically broadcasts beacon messages with its identity UID to announce its existence. These beacon messages are used for knowing all the other nodes in its proximity. One of the nodes that possess token initiates the spanning tree construction process. The initiating node broadcasts an “Any Token” message to check whether any other node has token in its proximity. The nodes that possess a token send “I Have Token” message to the broadcaster. The broadcaster waits for t ms to collect the replies. After collecting the replies from all those nodes that possess token, the broadcaster merges all these subtrees that are formed by the vehicles moving in the same direction to form a larger tree. For the vehicles that are moving in the opposite direction depending on the speed of the movement of the vehicles, if they still remain in its proximity for more than 5 s, then they are included in the spanning tree in order to disseminate message faster to the forth coming vehicles that are fragmented from the network; otherwise they are not included in order to avoid frequent changes in the spanning tree. Whenever the trees are combined to form a single tree, all tokens are deleted except the token of its farthest node [14], which has highest node degree. Since each fragment can have only one token, the merging process never results in cycle. The parameter node degree initially contains the number of edges incident with it, but during the process of construction it is updated to represent the candidate edges that may be included in the spanning tree; that is, the number of edges which are not included so far and whose inclusion will not create any cycle in the constructed spanning tree.

For handling dynamicity of the network topology and to handle the fragments in the network, the token is circulated among the nodes in the spanning tree in depth first manner [8]. Whenever a node receives a token, if it has noticed any change that occurred in its neighborhood since its previous turn, it starts the broadcast process [9]. Otherwise it passes the token to its next neighbor that is not yet traversed. When all its neighbors are traversed, it returns the token to its parent. By initiating the broadcast process only when the node noticed changes in its proximity since its previous turn, it reduces the bandwidth consumption required for transmitting “Any Token” message and also reduces the delay in the spanning tree construction process.

Also the disappearance of link is identified by nonreceipt of beacon messages. When a node A does not hear the beacon message from a known neighbor node B for more than 300 ms to 900 ms depending on the current traffic condition, it considers the link between A and B as failed and checks whether it belongs to the subtree that possesses token or not by checking the token movement direction. If it belongs to the subtree that has no token, then a token is generated; otherwise it does nothing. For handling the network fragmentation, it follows simple store-and-forward mechanism [15]. The messages received are stored for a shorter duration in which they have validity. When two spanning trees are merged, the messages that are still having validity are exchanged between them. The duration of validity of the message depends on the type of information it carries.

3.5. Reduction of Internal Nodes. The number of internal nodes of the spanning tree has a great impact on the number of rebroadcasting nodes. Our proposed algorithm aims to decrease the number of such rebroadcasting nodes by increasing the number of leaf nodes. This is achieved by increasing the number of children to internal nodes. If N is the number of nodes in the tree and L is the number of leaf nodes, then the number of internal (nonleaf) nodes M is given by $M = N - L$. Let I denote the set of internal nodes. The average degree d of an internal node is given by

$$d = \frac{\sum_{j \in I} d_j}{M}, \quad (1)$$

where d_j is the degree of internal node j . Note that $d_j \geq 2$ for all internal nodes j . Hence, $d \geq 2$. Since a tree T with N nodes has exactly $N - 1$ edges and each edge contributes to 2 degrees, the total degree of all the nodes of the tree is $2(N - 1)$. Also, since the leaf nodes have degree 1, we have

$$2(N - 1) = \sum_{j \in I} d_j + L, \quad (2)$$

$$\sum_{j \in I} d_j = 2(M + L) - L - 2, \quad (3)$$

Also,

$$\sum_{j \in I} d_j = M + (M + L) - 2, \quad (4)$$

$$\sum_{j \in I} d_j = M + N - 2 \approx M + N. \quad (5)$$

Dividing (3) by M on both sides, we get

$$\frac{\sum_{j \in I} d_j}{M} = \frac{2M + L}{M} = 2 + \frac{L}{M}. \quad (6)$$

Substituting (1) in the above equation,

$$d = 2 + \frac{L}{M}. \quad (7)$$

That is,

$$L = M(d - 2). \quad (8)$$

Similarly, dividing (5) by M on both sides,

$$\frac{\sum_{j \in I} d_j}{M} = \frac{M + N}{M} = 1 + \frac{N}{M}. \quad (9)$$

Substituting (1) in the above equation,

$$d = 1 + \frac{N}{M}. \quad (10)$$

That is,

$$M = \frac{N}{(d - 1)}. \quad (11)$$

Substituting (11) in (8) we get

$$L = N \frac{(d - 2)}{(d - 1)}. \quad (12)$$

Since $d \geq 2$ and hence $(d - 2)/(d - 1)$ is a strictly increasing function, the number of leaf nodes L increases when d increases and hence the number of internal nodes M decreases. Therefore, the number of internal nodes is inversely proportional to the degree d .

In our spanning tree, we make, for every possible internal node, the maximum possible children. Hence, with maximum possible degree d for every internal node increases the number of leaf nodes. In [16], Lu and Ravi use Kruskal's Algorithm for developing a spanning tree by connecting two different connected components. Initially all nodes are considered as separate forests of trees. Then they constructed a spanning tree by adding k -edges that connects k different subtrees and combined to form a larger spanning tree. They used Union and Find algorithms to check whether inclusion of the edge does not form cycle. In our implementation we adapted this concept in a distributed version. In our distributed version, the possession of token by two adjacent nodes ensures that they belong to different components. Also during the process of combining several connected components into a single connected component to form larger spanning tree, only one of the component's tokens is retained and other tokens are deleted. This ensures that each connected component possesses only one token. Hence, there is no need for a separate implementation of Union and Find algorithms in our distributed version of the algorithm. Also the next broadcaster of the construction process is the node that is far away from the current node with highest node degree. This further increases the number of nodes integrating into the spanning tree by making one more node as internal node.

TABLE 1: Simulation parameters.

Parameter	Value
Transmission range	250 m
Simulation area	2000 m × 2000 m
Vehicle speed	25 km/h–100 km/h
Simulation time	2 hours

TABLE 2: Parameters for setting up mobility scenarios.

Parameter	Dense	Average	Sparse
Beacon interval	300 ms	150 ms	100 ms
Average number of vehicles injected per km ²	378	257	198
Distance between vehicles	10 m	25 m	35 m
Average speed of the vehicle	25 km/h	65 km/h	100 km/h

3.6. *Pseudocode.* For more details see Pseudocode 1.

4. Experimental Results

The algorithm is implemented in JAVA. Table 1 lists all the simulation parameters. The transmission range is fixed as 250 m. The simulation area is restricted to 2000 m × 2000 m. The simulation area consists of straight roads, curvatures, and road junctions with one way and two ways as well as four ways. Simulation is run for 2 hours. The vehicle speed is restricted as 25 km/h to 100 km/h. The Intelligent Driver Model (IDM) is used for computing the acceleration and deceleration and the probability for lane change [17]. As suggested by Zhang et al. after a detailed study of mobility of vehicle in terms of duration of intervehicle link availability and the number of intervehicle link changes for maintaining an active link, random moving model is adopted in sparse scenario when the vehicle is moving relatively fast [18].

To reduce the delay in the construction process of the spanning tree, the broadcaster waits only for maximum of 300 ms to collect replies from the nodes in its proximity.

The simulation is run in dense, average, and sparse traffic scenarios. The dense traffic is similar to urban traffic where the speed is less due to greater number of vehicles in one km² area and traffic jam. The sparse traffic is similar to highway environment where the speed of movement of the vehicle is high and the number of vehicles within one km² area is less. Table 2 lists all parameters for setting up the three different mobility scenarios, namely, dense, average, and sparse. Based on the current speed of the vehicle, the type of the scenario is determined. In dense traffic, the change in the neighborhood is less frequent because of the low speed of the vehicles. In order to reduce the unnecessary beacon messages, the beacon interval time is also adjusted based on the current traffic scenario [3].

Since in dense traffic the mobility is less and neighborhood change is less often whereas in sparse traffic the mobility is high and the change in neighborhood is very often, hence the beacon messages are broadcast every 300 ms in dense traffic, every 150 ms in average traffic, and every 100 ms in

```

(1) Initialization: (Algorithm executed at every node initially)
    Set NS = "Not Integrated"
    Create a Token object
    Set TS = true
    Parent = null
    Broadcast Beacon (UID) periodically
(2) When change in neighborhood is detected: [9]
    (i) On receiving Beacon Message from node A for the first time:
        Add node A to list of neighbors
        Mark the edge to A as 0
    (ii) When no Beacon Message was received from node A for more than  $d$  ms: ( $d = 300$  ms/450 ms/900 ms)
        Remove node A from the list of neighbors
        If the neighbor list becomes empty
            Set NS = "Not Integrated"
            Create Token
            Set TS = true
        Else if this node belongs to the sub-tree that has no token
            Create Token
            Set TS = true
(3) A node that possesses the token and Initiates the construction process:
    Call BCAnyToken
(4) Procedure BCAnyToken:
    Broadcast "Any Token (UID, dir, speed)?" Message
    Wait for  $t$  ms to collect replies.
    If no reply is received
        If NS = "Integrated"
            Call CirculateToken
        Else
            Wait for change in its neighborhood
            Call BCAnyToken
    Else
        For each reply received on edge  $e$ , label that edge as 1
        Find the node  $B$ , which has the highest node degree among the newly joined nodes
        Send "Delete Token" message to all nodes other than  $B$  from which reply is received
        Delete Token
        Set TS = false
        Send "Token Deleted" Message to node  $B$ 
        Set NS = "Integrated"
(5) On receiving "Any Token" Message from node A:
    If TS = true
        Send Reply (UID, degree, dir, speed) to A
        Wait for reply from node A
        Set NS to "Integrated"
    If "Delete Token" message is received
        Delete the Token of it
        Set TS = false
    Else if "Token Deleted" Message is received
        Call BCAnyToken
(6) Procedure CirculateToken:
    Find the next child C in the tree that has not yet traversed
    If all children were traversed
        Reset all its children as not traversed
        If its parent is not null
            Pass on the token to its parent
            Set TS = false
        Else
            If any change occurred in its proximity
                Call BCAnyToken
            Else
                Call CirculateToken

```

```

Else
  Pass on the token to C
  Set TS = false
  Mark C as traversed
(7) Upon receiving the token from node A
  Set TS = true
  Set parent as A
  If any change occurred in its proximity
    Call BCAnyToken
Else
  Call CirculateToken

```

PSEUDOCODE 1

TABLE 3: Simulation results.

Parameter	Dense	Average	Sparse
Percentage of devices reached	100	99.99	96.56
Percentage of nodes rebroadcasting (on average)	15.02	25.14	29.98
Initial spanning tree construction overhead (on average)	721 ms	598 ms	460 ms
Dissemination time (on average)	1.52 s	990 ms	752 ms
Volatility of link (on average)	0.0412	0.0554	0.0612

sparse traffic. In a $2000\text{ m} \times 2000\text{ m}$ simulation area, in dense traffic 378 vehicles are injected, in sparse traffic 198 vehicles are injected, and in average traffic 257 vehicles are injected. The distance between the vehicles is kept as 10 m in dense traffic, 25 m in average traffic, and 35 m in sparse traffic. Table 3 lists the results found from the simulation run. From the results obtained during simulation, we found that in dense traffic the number of rebroadcasting nodes is only 15% whereas in average and sparse traffic due to the lesser node degree, the number of rebroadcasting nodes falls in between 25 and 30%.

As cited in Ruiz et al. [8], Weighted P -persistent Broadcasting (WPB) scheme [4] is a probability based algorithm that achieves 92.5% reachability with 27.89% of nodes rebroadcasting in sparse network and 99.6% reachability with 20.19% of nodes in dense network. The BODYF protocol suggested by Ruiz et al. [8] achieves 94% reachability with 31.57% of nodes rebroadcasting in sparse networks and 99% reachability with 24% of nodes rebroadcasting in dense network. But our algorithm achieves 96% reachability with 29.98% of nodes rebroadcasting in sparse networks and 100% reachability with 15% of nodes rebroadcasting in dense network. The reduction in rebroadcasting nodes reduces the number of messages circulating in the network and thus it decreases the bandwidth consumption.

The time required for initial spanning tree construction is maximum 721 ms, 598 ms, and 460 ms in dense traffic, average traffic, and sparse traffic, respectively. The increase in the time is due to the longer beacon interval time. If we decrease the beacon interval time, it may decrease the construction overhead, but it results in increase in

FIGURE 1: Percentage of nodes rebroadcasting in different traffic scenarios.

unnecessary beacon messages. In our simulation, the node broadcasts one safety message in every 20 s. The average dissemination time required for the broadcast messages to achieve 100% reachability is 1.52 s, 990 ms, 752 ms in dense, average, and sparse traffics, respectively. The initial spanning tree construction time and the time required for updating the spanning tree during movement are also amortized over the long run in computing the dissemination time. The volatility of link is computed for each traffic scenario to study the stability of the link. Volatility of link is the ratio of number of times the link appears to the lifetime of the link. Volatility ranges between 0 and 1, where 1 means instable and 0 means stable. Our simulation results show that the links are stable in all three scenarios and in dense traffic condition the stability of the link is higher than the other two scenarios because of the less change in the neighborhood due to low speed.

Figure 1 shows the performance of the algorithm by comparing the percentage of nodes rebroadcasting the message in dense, average, and sparse traffic scenarios. Figure 2 shows the time required to construct the spanning tree initially and it is higher in dense traffic than in sparse and average traffic. Figure 3 shows the dissemination time required to achieve 100% reachability in dense, average, and sparse traffic.

FIGURE 2: Initial spanning tree construction overhead in different traffic scenarios.

FIGURE 3: Dissemination time in different traffic scenarios.

5. Conclusion and Future Work

In this paper we have devised a new distributed algorithm for constructing spanning tree with maximum leaf nodes possible thereby reducing the number of rebroadcasting nodes in order to curtail the unnecessary retransmissions and hence wastage of bandwidth. We compared the performance of the algorithm in different traffic scenarios like dense, average, and sparse. From our simulation results, we conclude that our algorithm effectively reduces the number of rebroadcasting nodes with no reduction in reachability and with less control overhead. It also constructs a spanning tree with higher stability by considering the direction of vehicle movement in case of two way and four way roads. The limitation of our algorithm is its overhead in initial spanning tree construction due to the waiting time of “ t ” ms for collecting replies

from the neighbors. But unnecessary waiting time is also reduced by initiating the broadcast process only when any change occurs in the neighborhood. Thus, our distributed spanning tree algorithm may be used in all mobile networks including vehicular networks for disseminating messages containing current traffic information, which provides higher reachability with better bandwidth utilization. Since the information disseminated is about the current traffic, for which confidentiality, privacy, and security aspects are not mandatory, they are not given much importance in the current work.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, “A comprehensive survey on vehicular ad hoc network,” *Journal of Network and Computer Applications*, vol. 37, pp. 380–392, 2014.
- [2] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM ’99)*, pp. 151–162, 1999.
- [3] Y. Mylonas, M. Lestas, and A. Pitsillides, “Speed adaptive probabilistic flooding in cooperative emergency warning,” in *Proceedings of the 4th Annual International Conference on Wireless Internet*, Maui, Hawaii, USA, 2008.
- [4] N. Wisitpongphan, O. K. Tonguz, J. S. Parikh, P. Mudalige, F. Bai, and V. Sadekar, “Broadcast storm mitigation techniques in vehicular ad hoc networks,” *IEEE Wireless Communications*, vol. 14, no. 6, pp. 84–94, 2007.
- [5] S. M. Bilal, C. J. Bernardos, and C. Guerrero, “Position-based routing in vehicular networks: a survey,” *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 685–697, 2013.
- [6] C. Liu and C. Chigan, “RPB-MD: providing robust message dissemination for vehicular ad hoc networks,” *Ad Hoc Networks*, vol. 10, no. 3, pp. 497–511, 2012.
- [7] P. Ruiz, B. Dorronsoro, P. Bouvry, D. Khadraoui, and L. Tardón, “BODYF—a parameterless broadcasting protocol over dynamic forest,” in *Proceedings of the High Performance Computing & Simulation Conference (HPCS ’08)*, pp. 297–303, Nicosia, Cyprus, 2008.
- [8] P. Ruiz, B. Dorronsoro, P. Bouvry, and L. Tardón, “Information dissemination in VANETs based upon a tree topology,” *Ad Hoc Networks*, vol. 10, no. 1, pp. 111–127, 2012.
- [9] A. Casteigts and S. Chaumette, “Dynamicity aware graph relabeling systems (DA-GRS), a local computation based model to describe MANET algorithms,” in *Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing Systems*, pp. 231–236, Phoenix, Ariz, USA, 2005.
- [10] Z. Doukha, S. Moussaoui, N. Haouari, and M. E. A. Delhoum, “An efficient emergency message dissemination protocol in a vehicular ad hoc network,” in *Networked Digital Technologies: Proceedings of the 4th International Conference, NDT 2012, Dubai, UAE, April 24–26, 2012, Part I*, vol. 293 of *Communications in Computer and Information Science*, pp. 459–469, Springer, New York, NY, USA, 2012.

- [11] K. Z. Ghafoor, J. Lloret, K. C. Lee, K. Abu Bakar, and R. H. Khokhar, "Intelligent beaconless geographical forwarding for urban vehicular environments," *Wireless Networks*, vol. 19, no. 3, pp. 345–362, 2013.
- [12] L. Hogue, P. Bouvry, M. Seredynski, and F. Guinand, "A bandwidth-efficient broadcasting protocol for mobile multi-hop ad hoc networks," in *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICN/ICONS/MCL '06)*, pp. 71–79, IEEE, Morne, Mauritius, April 2006.
- [13] G. Kalpana and M. Punithavalli, "Reliable broadcasting using efficient forward node selection for mobile adhoc networks," *International Arab Journal of Information Technology*, vol. 9, no. 4, pp. 299–305, 2012.
- [14] K. Bür and M. Kihl, "Evaluation of selective broadcast algorithms for safety applications in vehicular ad hoc networks," *International Journal of Vehicular Technology*, vol. 2011, Article ID 730895, 13 pages, 2011.
- [15] O. K. Tonguz, N. Wisitpongphan, and F. Bai, "DV-CAST: a distributed vehicular broadcast protocol for vehicular ad hoc networks," *IEEE Wireless Communications*, vol. 17, no. 2, pp. 47–57, 2010.
- [16] H. Lu and R. Ravi, "Approximating maximum leaf spanning trees in almost linear time," *Journal of Algorithms*, vol. 29, no. 1, pp. 132–141, 1998.
- [17] M. Treiber and A. Kesting, "An open-source microscopic traffic simulator," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 3, pp. 6–13, 2010.
- [18] L. Zhang, A. Lakas, H. El-Sayed, and E. Barka, "Mobility analysis in vehicular ad hoc network (VANET)," *Journal of Network and Computer Applications*, vol. 36, no. 3, pp. 1050–1056, 2013.