INTERSECTION-BASED ROUTING PROTOCOL FOR VEHICULAR AD-HOC NETWORKS

(ARAÇLAR ARASI AĞLAR İÇİN KAVŞAK YÖNLENDİRMELİ TAŞIMA PROTOKOLÜ)

by

Çağdaş YAMAN, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

INSTITUTE OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Sep 2016

This is to certify that the thesis entitled

INTERSECTION-BASED ROUTING PROTOCOL FOR VEHICULAR AD-HOC NETWORKS

prepared by **Çağdaş Yaman** in partial fulfillment of the requirements for the degree of **Masters in Computer Engineering** at the **Galatasaray University** is approved by the

Examining Committee:

Prof. Dr. Tankut ACARMAN (Supervisor)
Department of Computer Engineering
Galatasaray University ------------------------

Prof. Dr. Mehmet Turan SÖYLEMEZ
Department of Computer Engineering
İstanbul Technical University ------------------------

Assist. Prof. Dr. Murat AKIN
Department of Computer Engineering
Galatasaray University ------------------------

Date: ------------------------

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Prof. Dr. Tankut Acarman. The door to Prof. Acarman’s office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

Also, I would like to thank Infotech Information and Communication Technologies for providing the data required for simulation studies.

I would like to thank Ahmet Birsen who supported me through the whole study.

I would like to thank my colleagues for their support on multiple issues that i had.

Finally, I would like to thank NASA for discovering many planets on goldilocks zone, and cheering the world.

May 2016
Çağdaş Yaman

TABLE OF CONTENTS

LIST OF SYMBOLS

AODV	**:** Ad hoc On-Demand Distance Vector Routing
BTP	**:** Basic Transport Protocol
C2C-CC	**:** Car 2 Car Communication Consortium
CBT	**:** Channel Busy Time
CCH	**:** Control Channel
DSRC	**:** Dedicated Short Range Communications
D-VADD	**:** Direction First Probe VADD
EDD	**:** Expected Disconnection Degree
ETSI	**:** European Telecommunications Standards Institute
GPCR	**:** Greedy Perimeter Coordinator Routing
GPS	**:** Global Positioning System
GPSR	**:** Greedy Perimeter Stateless Routing
GyTAR	**:** Greedy Traffic Aware Routing Protocol
H-VADD	**:** Hybrid Probe VADD
IBRP	**:** Intersection-Based Routing Protocol
IEEE	**:** Institute of Electrical and Electronics Engineering
ITS	**:** Intelligent Transportation Systems
JBR	**:** Junction-Based Geographic Routing Algorithm
LLC	**:** Logical Link Control
LOS	**:** Line Of Sight
L-VADD	**:** Location First Probe VADD
MAC	**:** Medium Access Control
MURU	**:** Multi-Hop Routing Protocol for Urban Vehicular Ad Hoc Networks
NLOS	**:** Non Line Of Sight
NS-3	**:** Network Simulator 3
PS	**:** Perimeter Search
SUMO	**:** Simulation of Urban Mobility
VANET	**:** Vehicular Ad-Hoc Network
VADD	**:** Vehicle-Assisted Data Delivery

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The smart city applications attract attention more and more in today’s world, and vehicular network is one of the big and important part of them. VANET provides many safety and infotainment applications with DSRC (Dedicated Short-Range Communications). Since VANET’s structure is ad-hoc based, it lacks the standard infrastructure of the classical computer networks. Some of the applications in VANET requires multi-hop routing between distant nodes in network. However, reliable routing is a major problem in VANET, because of random distribution and high mobility of the nodes. This thesis intends to provide a reliable routing algorithm in urban areas for VANET. With the help of GPS devices and digital maps, the proposed algorithm uses the network nodes at road intersections to achieve a successful geo-routing. In order to verify effectiveness of the proposed algorithm, two simulation studies are conducted. While SUMO is used to simulate urban city environment and vehicle behavior, NS-3 is used to simulate the vehicular network. MAC and Physical Layers of the network are implemented as IEEE 802.11p standards. The Network and Transport Layers is implemented as ETSI ITS standards. Within this context, the first results show that a reliable routing can be achieved with a simple digital map and even cheap GPS devices. The second study indicates that the proposed algorithm has affectively more success than some well-known routing algorithms.

RÉSUMÉ

Les applications de la ville intelligentes attirent l'attention de plus en plus dans le monde d'aujourd'hui, et réseau véhiculaire est l'une des parties grandes et importantes d'entre eux. VANET fournit de nombreuses applications de sécurité et de l'infotainment avec DSRC (Dedicated Short-Range Communications). Depuis la structure VANET est basée ad-hoc, il manque l'infrastructure standard des réseaux informatiques classiques. Certaines des applications dans VANET nécessitent multi-hop routage entre les nœuds distants en réseau. Cependant, le routage fiable est un problème majeur dans VANET, en raison de la répartition aléatoire et une grande mobilité des nœuds. Cette thèse se propose de fournir un algorithme de routage fiable dans les zones urbaines pour VANET. Avec l'aide de dispositifs GPS et des cartes numériques, l'algorithme proposé utilise des nœuds de réseau aux intersections routières pour obtenir une geo-routing réussie. Afin de vérifier l'efficacité de l'algorithme proposé, deux études de simulation sont effectués. Alors que SUMO est utilisé pour simuler le comportement de l'environnement de la ville et véhicule urbain, NS-3 est utilisé pour simuler le réseau véhiculaire. MAC et les couches physiques du réseau sont mises en œuvre en tant que normes IEEE 802.11p. Les couches réseau et transport est mis en œuvre en tant ETSI normes STI. Dans ce contexte, les premiers résultats montrent qu'un routage fiable peut être réalisé avec une carte numérique simple et appareils GPS, même à bas prix. La deuxième étude indique que l'algorithme proposé a affectivement plus de succès que certains algorithmes de routage bien connus.

ÖZET

Akıllı şehir uygulamaları dünya çapında ilgi görmeye başlamıştır ve araçlar arası ağlar bunun büyük bir parçası olmuştur. VANET, DSRC'nin yardımı ile güvenlik, bilgi ve eğlence tabanlı uygulamaların altyapısını oluşturmaktadır. VANET'in yapısı ad-hoc tabanlı olduğundan dolayı, klasik bilgisayar ağlarının sahip olduğu standart altyapıya sahip değildir. Bazı VANET uygulamalarının, ağdaki uzak uçlar arasında çok-hoplu yönlendirme algoritmalarına ihtiyacı vardır. Fakat uçların rastgele dağılımı ve hareketli davranışları bu ağlar üzerinde güvenilir bir yönlendirme algoritması geliştirmeyi yorucu bir problem haline getirmiştir. Bu tez VANET için şehir içi alanlarda güvenilir bir yönlendirme algoritması önermeyi amaçlamaktadır. Önerilen algoritma, dijital harita bilgisi ve GPS alıcıları yardımı ile, başarılı bir yönlendirme için kavşaklardaki ağ uçlarını kullanmaktadır. Algoritmanın başarısını sınamak için, iki adet simülasyon çalışması gerçekleştirilmiştir. SUMO, şehir içi altyapısını ve araç davranışlarını simüle etmek için kullanılmıştır. NS-3 ise araçlar arası ağ yapısını simüle etmiştir. Araçlar arası ağlarda, MAC ve fiziksel katman IEEE 802.11p standartlarına uygun, ağ ve taşıma katmanları ise ETSI ITS standartlarına uygun olarak tasarlanmıştır. İlk çalışmanın sonuçları araçlar arası ağlar içinde yalnızca ucuz GPS alıcıları ve dijital harita bilgisiyle güvenilir bir yönlendirme algoritması geliştirilebileceğini göstermiştir. İkinci çalışma ise önerilen algoritmanın literatürde iyi bilinen algoritmalardan daha başarılı olduğunu göstermiştir.

1. INTRODUCTION

Safety-critical information in VANET has to be disseminated in a timely and lossless manner. But short-lived connections between the vehicular nodes and severe risk of packet collision caused by simultaneous media access of vehicular nodes in the same radio transmission range degrades the performance of wireless broadcasting scheme in VANET. Especially, transmission of a data packet in an urban area is very challenging. Urban structures can occlude the wireless signals and a data packet cannot be delivered to the receiver in a single-hop. Therefore, the data packet needs to be relayed by the intermediate vehicular nodes. This thesis presents local routing algorithm using the vehicular nodes at the intersection zones situated in the neighborhood of the sender and receiver vehicular nodes for relaying purposes. In general, the navigation software routes the driver who is driving a land vehicle towards a neighbor intersection zone, and then to another one until reaching to his/her destination. This is very well known shortest path algorithm, i.e., a global routing algorithm that uses the road topology information and minimizes the travelled road distance. Routing is calculated at the beginning of the trip. On the other hand, the local routing algorithm does not require the whole road topology map but only the connection (or link, road ID) information about the intersections at the vicinity. In this thesis, a local routing is developed by using the navigation road map data involving the road topology, and the wireless network (VANET) measurements. The intersection zone can be subject to a high number of vehicular nodes and relaying node cannot access wireless media due to the channel busy condition. The presented algorithm takes into account the wireless network condition measured by the vehicular nodes.

This thesis is organized as follows: Section 3 presents related work about routing algorithms in VANET. Local routing algorithm is presented in Section 4 Early simulation setup and results are presented in Section 5. A comparison study between common routing protocols is presented in Section 6.

2. THE GOAL OF THESIS

This thesis proposes an intersection-based routing algorithm for Vehicular Ad-Hoc Networks. The algorithm uses GPS positions and digital map information to find a proper route between source and destination nodes. Forwarding nodes calculate the scores for every neighbor junction if exists and forwards the packet to the junction with the best score. Score value of a junction is a function of node count, channel busy time and distance to destination. After the junction selection, the nodes between two junction uses greedy routing to reach the selected junction. The proposed algorithm is evaluated trough two simulation studies that are created with two different simulation tool, NS-3 and SUMO. The urban city environment is simulated with the SUMO, and the VANET is simulated with NS-3.

3. RELATED WORKS

In the open literature, several routing mechanisms are designed to deliver packet from source to destination. These mechanisms can be categorized according to working principles.

3.1 Greedy Perimeter Stateless Routing (GPSR)

GPSR is one of the well-known protocols considered for MANETs (Karp and Kung (2000)). GPSR requires the information exchange between neighbor nodes and the exact destination of the destination node. The algorithm is composed of two methods, namely greedy and perimeter forwarding. The source node always starts with the greedy algorithm which forwards the packet to the closest node to destination's location in the neighbor list. But the greedy algorithm fails when packet reaches a local optimum that hasn't any close neighbor to destination but itself. (See Figure 3.1)

When greedy algorithm fails, GPSR begins to use perimeter search method. To use this method, the nodes must create a planar graph from positions of the neighboring nodes. A planar graph is defined as a graph whose edges never cross each other. So the planar graph is a union of polygons. If the greedy algorithm fails to forward the packet to destination, the last nodes prepares the packet for perimeter search method. A special header which indicates that the packet must be forwarded with the perimeter search method is added to the packet. The new header includes the nodes position that starts the perimeter search, the destination's position, first edge of the current polygon in the graph and the last intersection reached between the PS mode starting node-destination line and the graph. In perimeter search mode, the packet is forwarded by the right-hand rule inside a polygon until it reaches the edge which crosses the line that connects the node which perimeter mode starts with and the destination node. If packet reaches that edge, the packet is forwarded to the edge of a next polygon that owns the first edge in the counter-clockwise direction.

Although GPSR achieves to forward packet to destination, it may fail to find the shortest available path. Also Greedy Perimeter Coordinator Routing (GPCR) is proposed for urban environments; instead of graph planarization, the algorithm use digital map information to build a recover strategy. (Lochert et al. (2005)

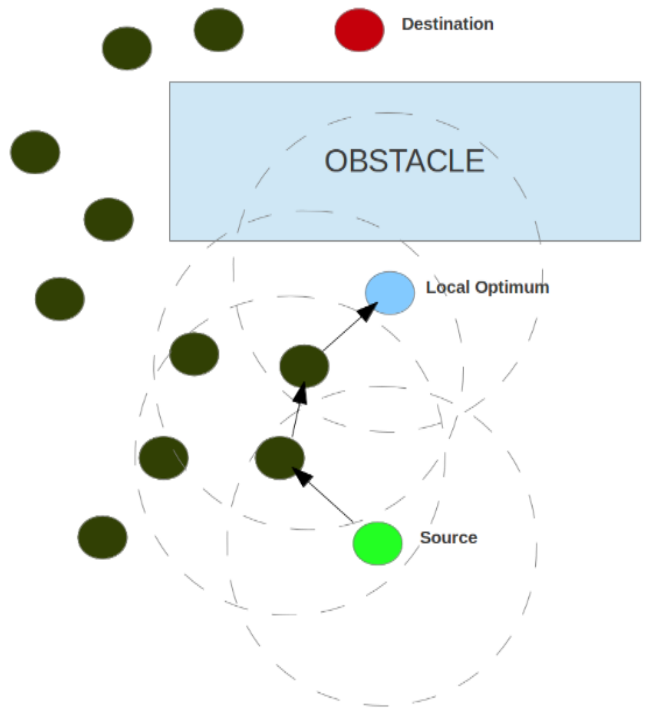

Figure 3.1: GPSR: Greedy Algorithm

3.2 Ad-hoc On-Demand Distance Vector Routing (AODV)

AODV uses small control packets to find a route, before the actual data transfer occurs (Perkins et al. (2003)). The algorithm first checks if an available path to destination exists. So it uses a cache to keep the discovered paths.

First step of the path discovery algorithm is broadcast a path request packet to the channel. The request packet holds six fields: source address, source sequence number, broadcast id, destination address, destination sequence number and hop count. The packet is forwarded trough all nodes until the maximum hop count is reached. If the destination receives the request packet, it sends a reply packet trough the same path where the request packet come. So all the receivers of the request packet must register the path to their cache with unique source-destination pair, the source sequence number and also previous node that sends the request. Every path has a timeout and it is updated when a data packet is sent successfully trough itself.

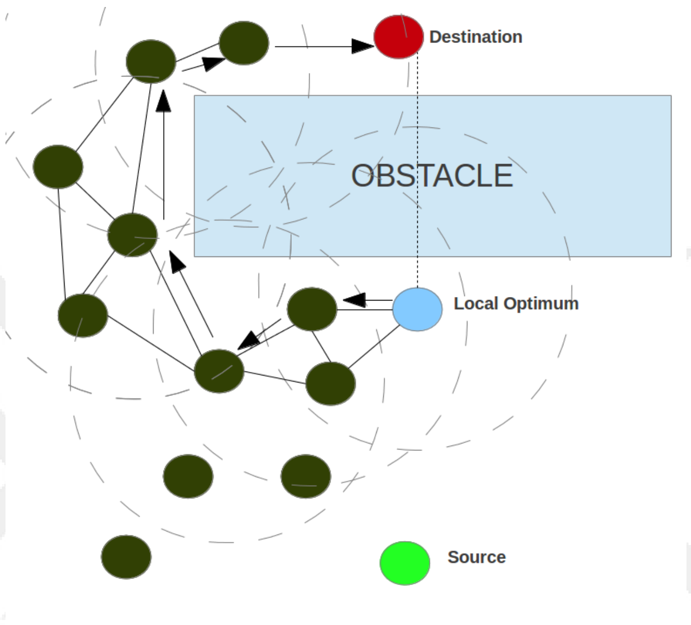

Figure 3.2: GPSR: Perimeter Search Scenario

Also, when a data transfers fails, used path removed from the cache and an error frame forwarded to previous nodes iteratively.

3.3 Junction-Based Geographic Routing Algorithm for Vehicular Ad-hoc Networks (JBR)

JBR uses selective greedy forwarding (Tsiachris et al. (2013)). With the help of digital map, every forwarding node divides their neighbors into coordinators and simple nodes. A coordinator indicates a node in junction. Coordinator nodes has high priority in forwarding process. If a node has the coordinator nodes in range that are closer to destination than itself, it chooses the closest one among them. If there aren't any coordinator nodes, then forwarding nodes chooses the closest simple node in the neighbor list.

If a packet reaches a local optimum at a simple node, the algorithm looks for the nodes at different directions than the one which the packet arrives. When local optimum problem occurs at a coordinator, the algorithm first checks if the destination is in the same road with the coordinator. If the coordinator and the destination are not in the same road, coordinator search for other coordinators or simple nodes that are not in the same road itself and the closest one is chosen. If the coordinator and the destination are in the same road, than coordinator seeks for closer simple nodes to forward the packet.

3.4 Vehicle-Assisted Data Delivery (VADD)

Vehicle-Assisted Data Delivery (VADD) protocol is a carry and forward based routing algorithm which requires a geographic map (Zhao and Cao (2008)). The protocol chooses a mode between three different modes by using the ego vehicle's location: Straightway Mode, Intersection Mode and Destination Mode. “Carry and forward” is a recovery method that is used if there is no routing possibility with wireless channel.

VADD can be divided three sub-protocols by the priority of the selection parameters. Location First Probe (L-VADD) primarily chooses the closest node to intersection at preferred direction without checking the vehicles heading direction. The first priority is always the smallest distance to the given destination. If there isn't any contact in the given direction, current vehicle that holds the packet continue to carry it until it finds a better node to forward.

Direction First Probe VADD (D-VADD)'s first choice is the nodes that move towards preferred direction. D-VADD solves the loop problem that can occur when a node is heading trough the opposite direction and has the closest distance to destination's direction. But the protocol may also increase the delivery-time.

Hybrid Probe VADD (H-VADD) has a loop detection mechanism. Normally, H-VADD acts same as the L-VADD for the fast-delivery of the data. If mechanism detects a routing loop, D-VADD is used as the routing behavior.

3.5 MURU

MURU enhances the path discovery method of AODV by including a digital map (Mo et al. (2006)). MURU uses the dissemination of the small control packets for finding a temporary path to destination. Different from AODV, dissemination process is bounded with the road geometry trough shortest path to destination.

Also, the protocol calculates the trajectories of the vehicles on any registered path by using the digital map data and speed information of the vehicles to find the availability time of that path. A metric called Expected Disconnection Degree (EDD) is used to formulate such event with time.

3.6 Greedy Traffic Aware Routing Protocol (GyTAR)

GyTAR is an intersection-based routing protocol which uses a similar algorithm with the proposed protocol (Jerbi et al. (2006)). It requires a digital map and beaconing information to calculate the best choice of action at each forwarding step. GyTAR has two parameters to calculate the score of connected junctions, which are traffic density and distance to the destination. Between two junctions, the protocol is forwarding greedily towards to selected junction. The score calculation can be explained as an equation:

$$S_j = \alpha * f(T_j) + \beta * f(D_j) \tag{1}$$

Where T_j is the traffic density of the junction and D_j is the distance to the destination. α and β are weighting factors.

Also, GyTAR has a recovery strategy for local optimum problem. When such problem occurs, the forwarding node carries the packet trough closest junction to the destination and then does the calculation again.

4. ALGORITHM

Due to fast changing topology of VANET and structures occluding the wireless signals in an urban area, communication links are short-lived in VANET. But the wireless access protocol in vehicular environment needs to be designed to assure lossless and fast packet delivery. The proposed protocol leverages the navigation map data and routing information used to reach at the destination point. A digital road map data is constituted by nodes and edges, or namely, road links are connecting the intersections and overall the road topology is formed. Since navigation can provide the feasible travel paths to reach the destination, the proposed routing protocol transmits the data packet in VANET via hopping or relaying at the intersection zones. In general, the condition of being Line-Of-Sight (LOS) with respect to the intersection from the oncoming path is applied to the routing selection of the data packet in VANET. For instance, the data packet is forwarded to the nearest LOS intersection for relaying, and then routed in the direction towards the second intersection in order to reach to the destination. The information of the intersection's position is obtained from the navigation based routing.

The protocol assumes that the vehicle has a GPS receiver, a dedicated short range communication (DSRC) modem and a navigation road map data. It works as part of ETSI ITS Network Layer protocol, a.k.a. Geo-networking protocol. It answers the "geo-unicast" and "geo-broadcast" requests of transport layer (ETSI 2011a). Also protocol uses location table and beaconing protocol for neighbor discovery.

The cost function is presented to compare the intersections listed among the alternative routes. These alternative routes are the physical road intersections for the vehicle to be routed to its destination and it is provided by the navigation data or choosing the neighbor intersections situated at the vicinity of the source node for a given heading angle interval towards the destination. For example, in a given semi-circle neighborhood of the source node with an antenna transmission range, three intersections exist. In this example, the cost function is computed for each individual intersection in order to maximize or minimize the travelling physical road distance. Additionally, VANET performance metric terms are added to the cost function in order to distinguish sparsely connected intersection zones, or in contrary, heavily congested or dense intersection zones. For instance, the cases where there are few vehicular nodes for possible relaying can be discarded, or densely occupied zones by the vehicular nodes and generating a high amount of data traffic can be avoided by routing to another intersection. The cost function is given as follows:

$$J_i = \alpha * D_i + \beta * N_i + \theta * C_i \tag{2}$$

Where J_i is the cost function calculated for the i-th intersection, D_i is the distance score, N_i the network connectivity metric, C_i the ratio of Channel Busy Time (CBT) versus the MAC Queue Size (MQS), and α, β, θ denotes the weighting factor.

The metric about network connectivity is provided by storing the beaconing messages of the neighbor vehicular nodes during each 100 milliseconds (ms) that is a default beaconing period. Beaconing message involves position, velocity, time, heading angle and map matched road ID of each vehicular node. This message is received in the antenna transmission range of the sender node. Then, at the end of each beaconing period, i.e., 100 ms, the sender vehicular node computes the network connectivity of each surrounding intersection for i=1,2,…,n, by counting the available vehicular nodes whose position is in the range of 30 meters with respect to the center of the particular intersection. Then, it is divided by a number representing the maximum number of vehicles could be occupying the intersection, for simplicity, this maximum number is taken as N=100 in order to convert network connectivity as a per unit number.

CBT is a metric for measuring the current congestion on the channel is the CBT ratio (ETSI 2011b). It measures the amount of time, for which the channel was sensed to be busy, and calculates the ratio with respect to a certain time interval. The channel congestion is measured by the CBT metric, which provides the ratio, for which the channel is detected as busy with respect to a certain amount of time, i.e., beaconing period. MQS is derived by the size of queue of the vehicular node, for instance, the size of data packet queued to be transmitted once the media is sensed to be free for access. MQS of each vehicular node is added to the beaconing message and then shared in VANET. Each vehicular node computes the average of MQS transmitted in its antenna transmission range and according to the heading angle and the road ID information provided in the beaconing message, for each different heading angle and road ID, MQS is separately computed. In this scheme, for each road link, link quality score is derived.

Distance metrics calculated as an exponential function of a ratio of two Euclidean distances, i.e., the distance between the i-th junction and destination versus the distance between relay (or source) and destination, which are denoted by D_{id} and D_{rd} in (3), respectively. This metric simply defines the possible relaying or hopping count until reaching to the destination. If a packet can be delivered from it source to its destination via a single hop, it is more beneficial in terms of maximizing the cost function.

$$D_i = e^{-\frac{D_{id}}{D_{rd}}} \tag{3}$$

The protocol has two modes: source mode and relay mode. If the source node is in the intersection zone, it selects two neighbor intersections from the navigation map. Then, it calculates the cost of each intersection, and selects the intersection with the highest cost, i.e., maximum distance, maximum connectivity and maximum link quality. Following selection of the intersection for relaying purposes, it finds the closest neighbor from the location table and sends the data in a unicast scheme. The relay node calculates cost if only it is in an intersection. Otherwise, a greedy forwarding (Karp and Kung (2000)) is accomplished towards the neighbor intersections.

5. EARLY WORK

For evaluating and analyzing the proposed protocol, we use two different simulators, one for mobility, and one for the network. SUMO (Simulation of Urban Mobility) is used to simulate behavior of urban traffic. We used a part of real navigation map of Kadıköy municipality, İstanbul having 143 nodes (intersections) and 261 edges (road links). Map is converted from an ARCGIS Shape File to SUMO XML format. One vehicle per each intersection is entered into the simulated area from 80 selected intersections at each every 27 seconds and the maximum speed to be reached by the vehicles is limited by the legal speed limit that is 50 km/h. Hence, the vehicle density in the simulated area is 8 vehicles per second.

Network is simulated by ns-3 (network simulator 3). Mac layer is configured as IEEE 802.11p CCH, and ETSI ITS Network and Transport layer standard is implemented, Geo-networking and BTP headers are used for transmission. Total packet size is 164 bytes and it is constituted by 32 bytes for transport layer dummy payload, 4 byte for BTP headers, 88 byte for geo-unicast header and 40 byte for LLC and MAC headers. Data dissemination frequency is 10 Hz, and beaconing frequency is 10 Hz.

Also, simulation has a proper NLOS propagation loss model which tracks obstacle count and length (Sommer et al. (2011)). Building information is added from a real building database of Kadıköy municipality. A 3D visualization is developed to enhance mobility tracking of both node and data packet, a screen shot of the simulated urban road topology, buildings, intersections, road links with their IDs, and vehicular nodes with their IDs is given in Figure 5.1. (Also See Annexes A)

For analyzing the effect of link quality awareness, two scenarios are implemented. In the first scenario, the data traffic in VANET is constituted by periodic beaconing messages and packet transmission from the source node to the destination node. In the second scenario, an additional data traffic is generated that causes a congestion in one main street between the simulation time duration of 10 and 30 seconds. To congest data traffic in the selected intersection, a vehicle waiting at the red light generates a dummy data packet of 2048 byte at each 100 micro seconds, or more clearly 1000 times during each minute. This abrupt data traffic generation prevents other neighbor vehicular nodes accessing to the wireless media.

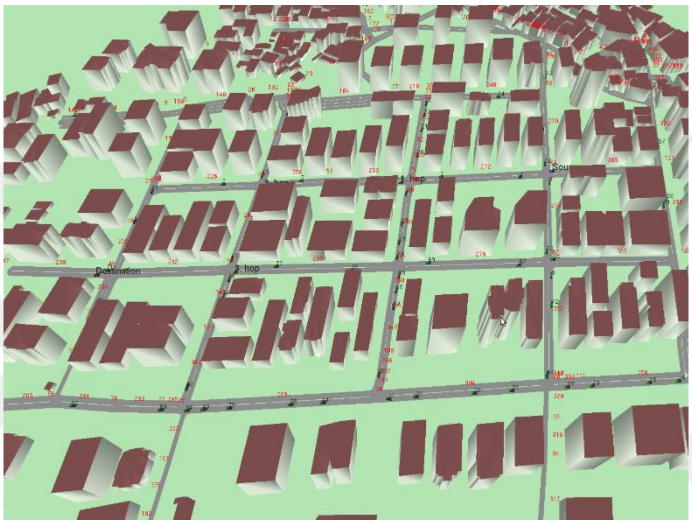

Figure 5.1: A screenshot of 3D simulator interface

The proposed algorithm is simulated in both of the scenarios and its performance is compared versus the shortest path algorithm. Shortest path algorithm is a global algorithm and it only uses the metric of the physical distance between the source and the destination node. This global algorithm is implemented such that the distance between source and destination is calculated for each link, i.e., at street level and then, routing information is added to the data packet such as virtual circuit ID to be used for routing the data packet at the intersection zones toward the given destination. Overall, global algorithm minimizes the overall physical road distance and the number of hopping at the intersections. But since global data routing only uses distance information, it is sensitive to the number of vehicles and possible data traffic congestion at the intersection, its performance can be degraded by the vehicular network conditions.

The responses of global routing algorithm are plotted in Figure 5.2. Since global algorithm is not taking into account the VANET measurements, a shortest path is chosen based on the road map data between the source and the destination vehicular node. Source node sends the data packet to the vehicular node in the vicinity of the intersection with an ID #59 for relaying purposes in a unicast scheme. This node near the intersection #59 is in the beaconing list of the source node. Then, this node at the intersection #59 relays the data packet in a greedy fashion, the vehicular node travelling between the intersection #59 and #48 relays the data packet as a second hopping. Then, the data packet is delivered to the destination node. Path of the data packet is calculated by the sender vehicular node before sending it. Then, routing is made by following the routing sequences.

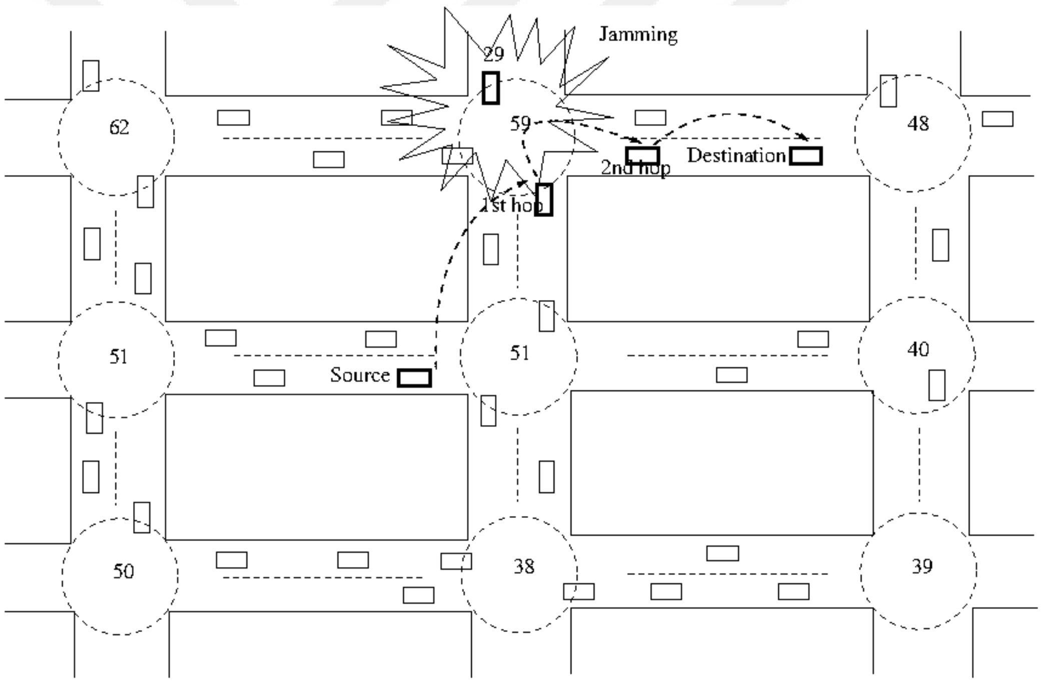

Figure 5.2: Global Algorithm's routing path subject to a burst of data traffic at the intersection #59.

On the other hand, the proposed algorithm is a local routing algorithm. The cost is calculated by the sender, then by the intermediate relaying nodes. Routing is calculated and made for each hopping separately. Therefore, the proposed algorithm takes into account the VANET performance and its metrics to improve routing at each intermediate relaying stage. The responses of the proposed routing algorithm are plotted in Figure 5.3. Due to data traffic congestion occurring at the intersection zone #59, the link quality is low, hence an alternative routing is made to the vehicular node in the neighbor intersection zone #40. Then, the vehicular node computes the routing cost to relay the data packet to its destination. The alternative intersections are the intersection #39 and #48. The relaying node computes the travelling distance cost as presented in (3), and since the distance between the intersection #39 and destination, denoted by D_{id} in (3), is higher with respect to the distance between the intersection #48 and final destination, the travelling cost is considered to be lower when the intersection #48 is chosen for relaying purposes. Then, the data packet is relayed by the vehicular node at the intersection #48.

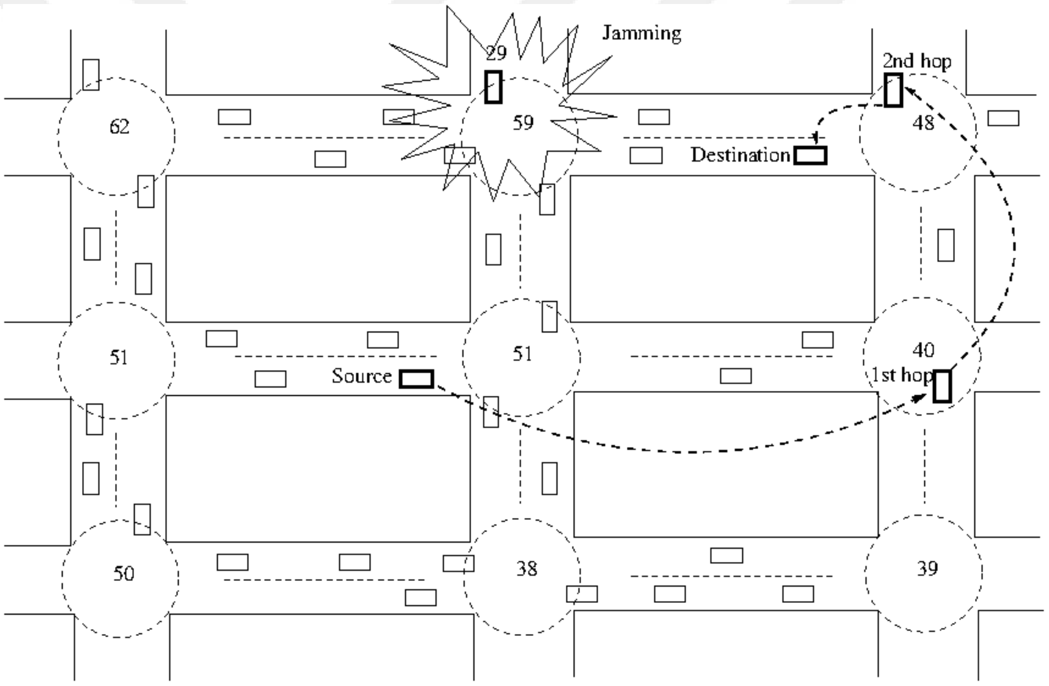

Figure 5.3: Proposed local algorithm's routing path subject to a burst of data traffic at the intersection #59.

The probability of absence of the vehicular node at the main intersections is calculated in Figure 5.4. The probability, denoted by p_0, presents the probability of absence of the vehicular node at the intersections at average during the simulation. Routing performances compared versus global routing algorithm. Some conclusions are given in below.

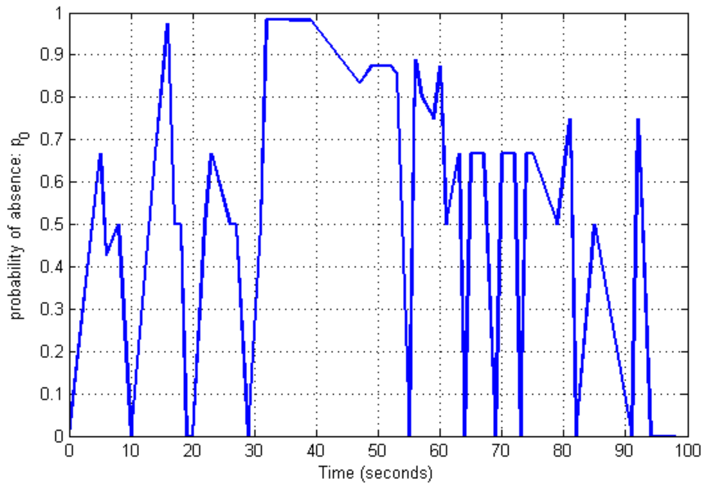

Figure 5.4: The probability of absence of the vehicular node at the main intersections

The packet reception performance is plotted in Figure 5.5. The probability of absence of the relaying node at the main intersections degrades the performance of the packet delivery in VANET. Besides this connectivity or relaying probability, during the time interval starting at 10 seconds and ending at 30 seconds, a high data traffic causing local data traffic congestion is artificially generated at the intersection #59. Packet delivery rate is plotted at the top of Figure 5.5 when there is a burst of data traffic at the intersection #59 subject to the connectivity status plotted in Figure 5.4. The delivery rate of the global routing is significantly dropped when the intersection #59 is subject to the high data traffic during the time interval of 10-30 seconds. The proposed local routing algorithm is simulated for two different sets of weighting factor. The first version of the weighting takes into account the distance cost and network connectivity equally, i.e., α = 0.5, β=0.5 and θ=0 and the second case uses the weighting factor set as: α = 0.5, β=0.3 and θ=0.2. When local routing algorithm is applied, data packet is routed through an intersection #40 even though the distance cost is lower in comparison to the intersection #59 that is subject to high data traffic. Packet delivery rate is increased after the start of high data traffic at t=10 seconds, and around at t=16 seconds the packet delivery is again recovered to its maximum value without being disturbed by the additional data traffic. When link quality, CBT versus MQS metric is not considered, the packet delivery rate is slightly higher. Then, the absence of the relaying node at the intersection during the time interval between 30 seconds and 38 seconds

(See Figure 5.4) degrades the packet delivery rate of the global and proposed local routing algorithm.

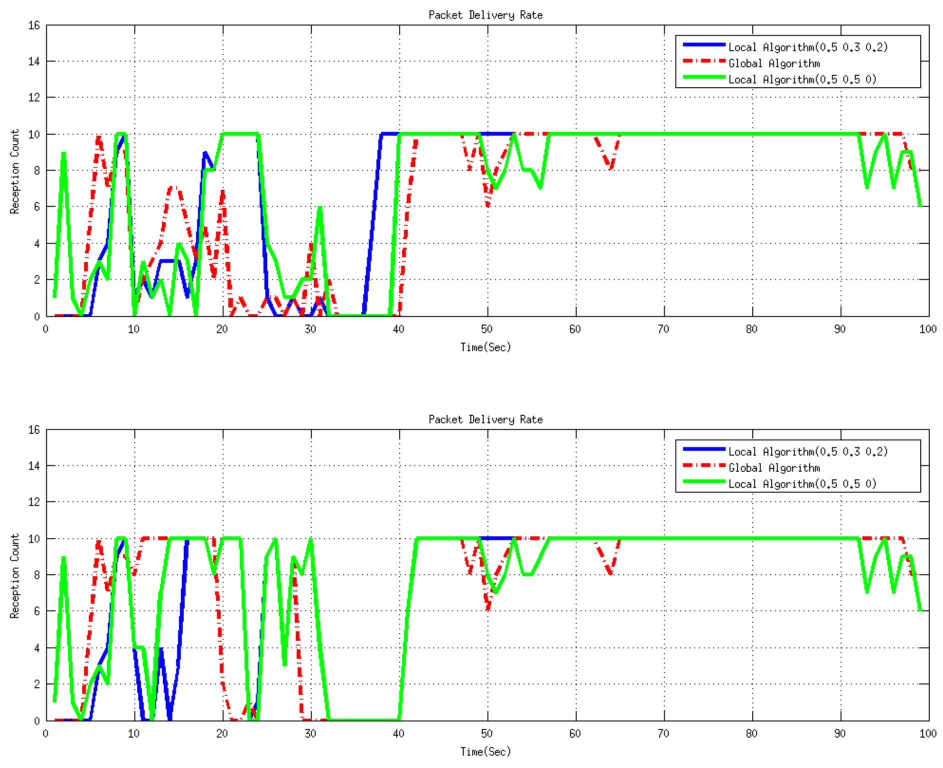

Figure 5.5: The time responses of packet delivery

When a burst of data traffic is not generated at the intersection #59, the packet delivery responses are plotted at the bottom of Figure 5.5. In that case, global algorithm is more successful except at the instances when the probability of the existence of the vehicular node for relaying is reduced significantly. But during congestion, local algorithm assures a higher rate of packet delivery.

Redundant packets are counted in Figure 5.6. When there is not a burst of data traffic causing data traffic congestion at the intersection #59, global and local algorithm has to generate almost the same number of redundant packet to assure reliable transmission. The number of redundant packet is plotted at the bottom of Figure 5.6. Local algorithm generates a slightly higher number of redundant data packet to forward to the vehicular nodes in the other neighbor intersections during the absence of the relaying node at #59. Data packet is resent if it is not relayed by the neighbor vehicular nodes and the timeout is set to 10 ms. When the intersection zone #59 wireless media is occupied by a burst of data traffic causing congestion, and in consequence packet collision occurring at the intersection zone #59, global algorithm enforces sending of redundant data packet to assure packet delivery. The number of redundant packet is plotted at the top of Figure 6. But local algorithm does not cause any redundant transmission between 18 seconds and 26 seconds due to its routing via the intersection zone #40 for relaying purposes, hence local algorithm avoids the jammed intersection #59.

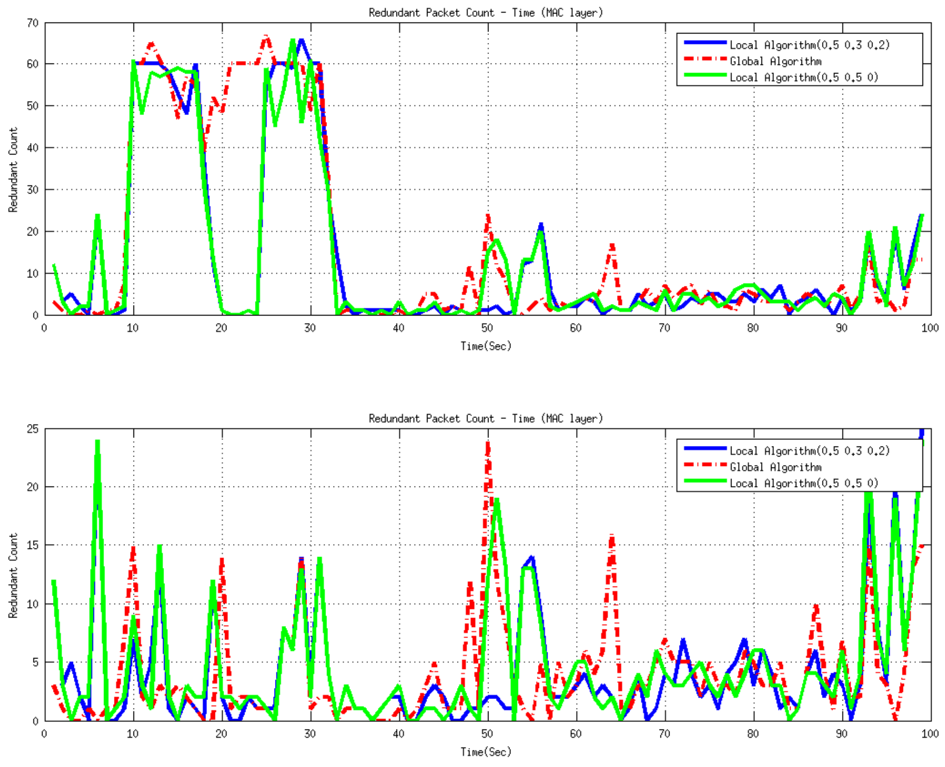

Figure 5.6: The time responses of redundant packet

The distance travelled by the data packet to reach to its destination is plotted in Figure 5.7. At each simulation time, the real road distance between the source node and destination node is calculated, then this calculated Euclidean distance is compared with respect to the total travelling distance of the data packet accumulated until reaching at its destination. Due to mobility of both source and destination node, the relative distance is varied. When there is a burst of data traffic causing data traffic congestion at the intersection #59, local algorithm can choose an intersection situated far from other neighbor intersections in order to benefit from a better network connectivity. And when there is no burst of data traffic degrading the connectivity at the intersection #59, the packet travelling distance for local and global algorithm is almost the same (in that case, packet travelling distance is plotted at the bottom of Figure 5.7. Except due to an increased number of redundant packet caused by the absence of the relaying node at the intersections satisfying the minimum distance requirement during the simulation time between 20 seconds and 30 seconds, global algorithm causes a higher packet travelling distance at average, see for instance at the bottom of Figure 5.8.

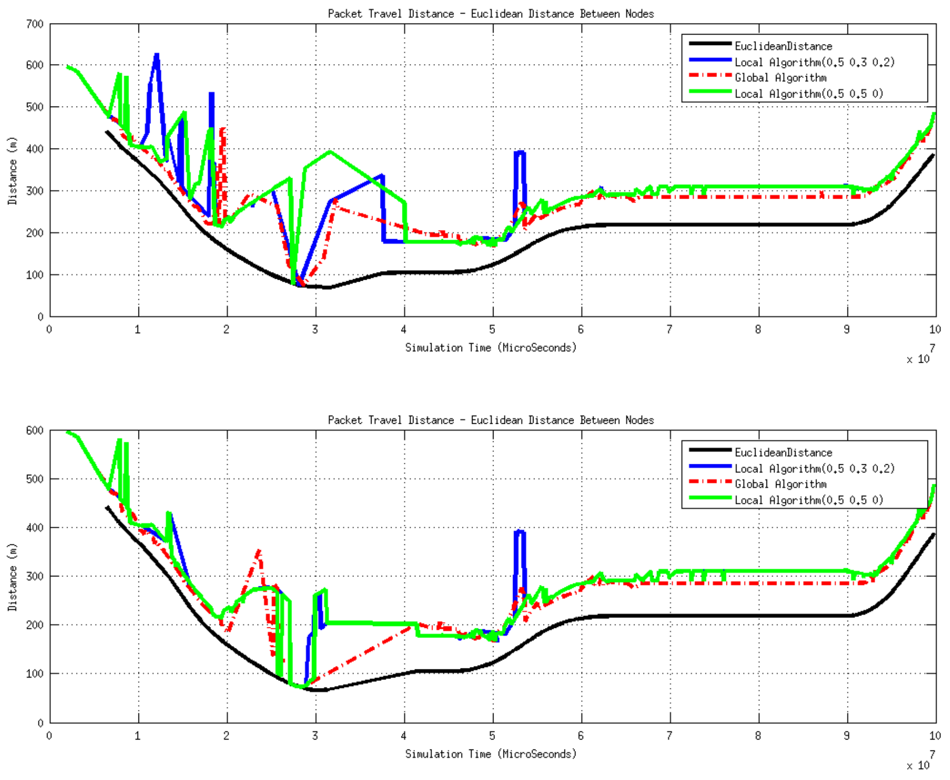

Figure 5.7: Packet travel distance compared to real distance between source and destination

The results are obtained when the vehicle density is 9 vehicles per second. The vehicle density affects the protocol performance. Therefore, packet delivery rate is simulated for different vehicle density. The vehicle density set of 4.5, 6, 9 and 18 vehicles per second is simulated. For lower values of vehicle density, i.e., the probability of absence of relaying vehicular node is higher, packet delivery rate of the global routing algorithm is comparably less than the local routing algorithm. Particularly, when additional data traffic is generated at the intermediate intersection, the global algorithm performance is degraded. When the vehicular density is 9 and the connectivity is not perturbed, global algorithm performs better. When the vehicle density is 18, global algorithm assures a higher rate of packet delivery since the local algorithm is penalized by CBT and it tries to find alternative intersections to route the data packet, which possibly causes packet delivery failure.

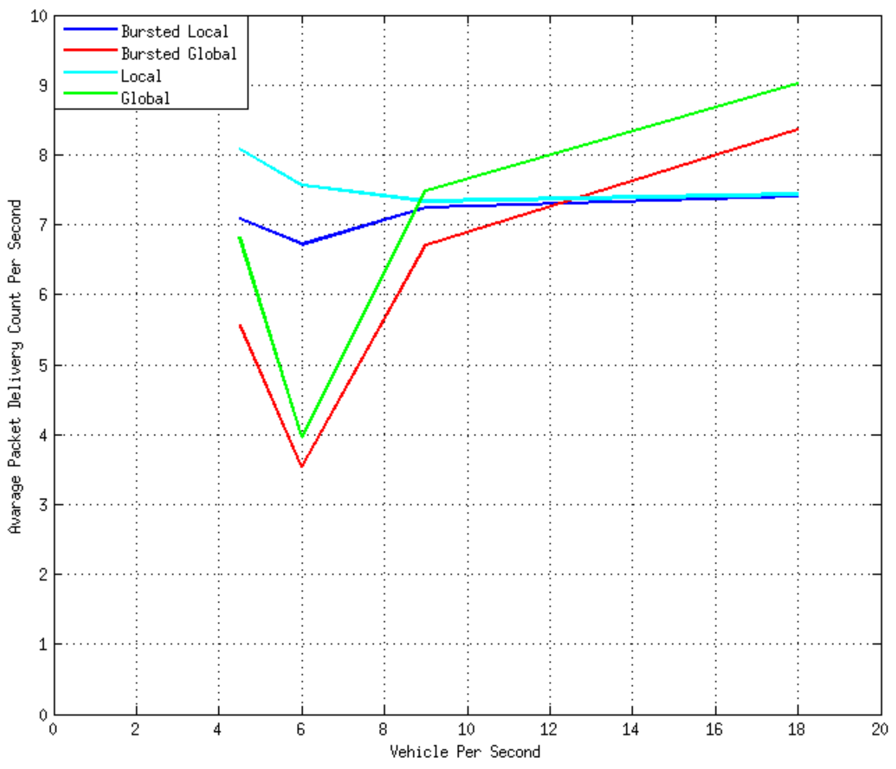

Figure 5.8: Average packet delivery rate for different vehicle densities

6. ADVANCE SIMULATION STUDY

For a better evaluation of the proposed protocol, a comparison study is planned with two other protocol, GPSR and GyTAR. SUMO and NS-3 are used as simulation platforms again but different versions and configuration.

An ideal Manhattan topology model is created for urban simulation with six horizontal and six vertical road links. The empty blocks is filled with equally distributed buildings. The gap between parallel roads is 200 meters, so there are 25 blocks with a dimension of 200x200 meters. Roads are bidirectional, a traffic flow is created from every junction and every start point with 13.5 second period (120 vehicles per 13.5 seconds). Vehicles are identical with constant 2.8 ms^{-2} acceleration, 4.5 ms^{-2} constant deceleration and 20 ms^{-1} maximum speed.

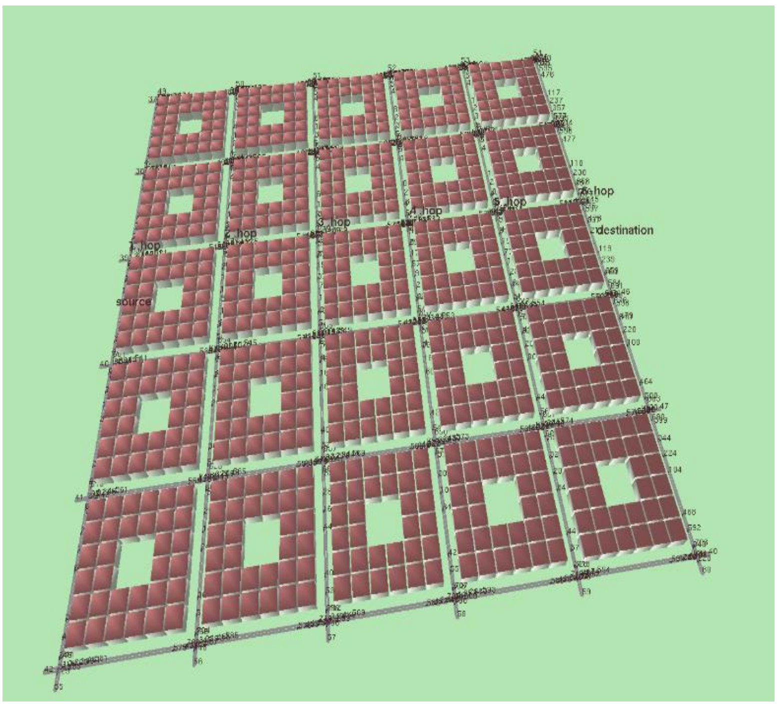

Figure 6.1: A 3D figure of Manhattan topology

GPSR is implemented as fourth (transport) layer of the OSI Network Model. First three layer is used as the same with the earlier simulations discussed at Section 5. So the control packets of both protocols are carrying 27 bytes long ETSI Common Header for Geo-Networking. Additionally, the shortest path algorithm (Dijkstra) is implemented also discussed at Section 5. Due to similarity between GyTAR and the proposed protocol, GyTAR is implemented as the proposed protocol with different weighting factors (zero weighting for CBT function).

For first implementation of the proposed protocol IBRP, the intersection list is retrieved from the static map and selected as the intersections which connected the ego vehicles current location. The protocol could only forward the packet one intersection at a time with this method. To solve this problem, the protocol's behavior is changed. The intersection list for scoring process is retrieved from table of neighbor as all the intersection that has at least one neighboring vehicle. Then, the proposed protocol calculates the intersection score as discussed as Section 4 and chooses the optimal junction.

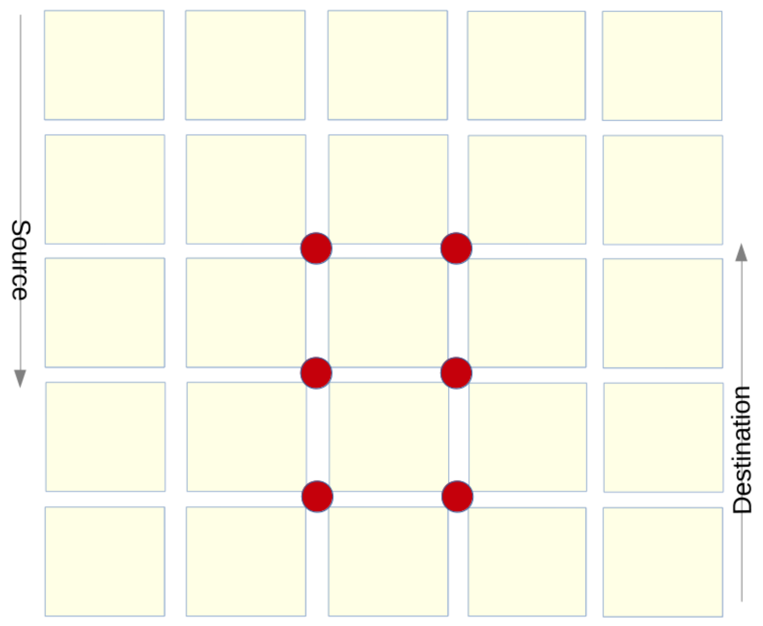

Figure 6.2: Figure of source and destination setup, the red dots are the places of the nodes that creates dummy traffic.

In every simulation, same two nodes (source and destination) are chosen for the monitored packet traffic. An application is created to provide a controlled data trading, resending and acknowledgement. Application sets a sequence number for every packet; the resented packets has the same sequence number with the original packet. The source and destination nodes moves parallel to each other through the most outer vertical roads of the topology. See Figure 6.2.

For a better understanding of how the selected protocols react to the different network traffic, four nodes is put to exact middle of the four chosen intersection to send dummy packets (See Figure 6.2.). By this four nodes, 1024 bytes of data is created regularly at different rates (10Hz, 20Hz, 25Hz, 33.3Hz, 50Hz, and 100Hz).

Table 6.1: Comparison Results for 10 Hz Dummy Traffic

	Delivery Ratio	End-to-End(s)	Avg. Dist.	Avg. Hop	Max. Seqno	Redundant #
IBRP	0.824	18.53	1339.25	7.16	997	169
GPSR	0.679	12.77	1235.41	6.04	997	309
GyTAR	0.773	18.75	1326.46	7.10	997	218
Dijkstra	0.528	17.92	1264.67	7.33	997	461

Table 6.2: Comparison Results for 20 Hz Dummy Traffic

	Delivery Ratio	End-to-End(s)	Avg. Dist.	Avg. Hop	Max. Seqno	Redundant #
IBRP	0.823	18.65	1341.13	7.17	997	171
GPSR	0.674	12.77	1227.83	6.02	997	313
GyTAR	0.782	18.61	1337.55	7.15	996	211
Dijkstra	0.526	18.09	1273.71	7.37	997	466

The low-traffic scenarios has similar results which shows a saturation at the packet delivery ratio values. The proposed algorithm has the best result in packet delivery ratio with approximately 82.5 per cent and the second best is GyTAR implementation with 78.5, while GPSR and Global Routing have low success ratios at delivery. GPSR has the shortest average hop count and distance travelled, because there is no boundary for the node selection and GPSR always chooses the closest node to destination. GPSR algorithm needs less information about the topology while IBRP and GyTAR uses all information about the distribution of neighbor nodes to find the best route. Redundant packet counts is affected by the packet delivery ratio, so their results have direct proportion as presented. Global Routing (Dijkstra - Shortest Path) has the lowest score in low-traffic scenarios, because it forwards the packets only from the shortest path available to destination without using any information about the current state of the node distribution. See Tables 6.1, 6.2, 6.3 and 6.4.

Table 6.3: Comparison Results for 25 Hz Dummy Traffic

	Delivery Ratio	End-to-End(s)	Avg. Dist.	Avg. Hop	Max. Seqno	Redundant #
IBRP	0.832	15.90	1348.44	7.21	997	163
GPSR	0.674	10.74	1233.89	6.06	997	313
GyTAR	0.791	15.71	1340.31	7.19	997	205
Dijkstra	0.526	15.31	1246.22	7.37	997	467

Table 6.4: Comparison Results for 33.3 Hz Dummy Traffic

	Delivery Ratio	End-to-End(s)	Avg. Dist.	Avg. Hop	Max. Seqno	Redundant #
IBRP	0.826	16.31	1341.62	7.17	997	169
GPSR	0.690	10.23	1241.74	6.11	996	298
GyTAR	0.781	16.04	1343.23	7.19	997	215
Dijkstra	0.521	15.51	1252.83	7.43	997	473

The high-traffic scenarios shows that increasing traffic in network is decreasing the success ratio difference of the proposed algorithm and GPSR. In 100 Hz dummy-traffic scenario, GPSR has slightly better success than the proposed algorithm, while GyTAR has 10 per cent lower success rate than GPSR and the proposed algorithm. This is because of the information gathered from network decreases with the increasing traffic in network. The score calculations of the junction selection based algorithms is disturbed by the lack of information needed from the network topology. The proposed algorithm is less affected than GyTAR by this situation, because it uses a function of CBT to calculate junction score, so it chooses the junctions that has lower traffic density. See Tables 6.5 and 6.6.

Table 6.5: Comparison Results for 50 Hz Dummy Traffic

	Delivery Ratio	End-to-End(s)	Avg. Dist.	Avg. Hop	Max. Seqno	Redundant #
IBRP	0.769	19.74	1288.88	6.89	997	222
GPSR	0.676	12.88	1229.57	6.04	997	310
GyTAR	0.723	20.37	1279.72	6.82	997	267
Dijkstra	0.524	19.02	1269.12	7.31	997	463

Table 6.6: Comparison Results for 100 Hz Dummy Traffic

	Delivery Ratio	End-to-End(s)	Avg. Dist.	Avg. Hop	Max. Seqno	Redundant #
IBRP	0.486	15.41	1011.03	5.58	997	481
GPSR	0.488	11.53	982.29	4.88	996	452
GyTAR	0.395	13.98	868.32	4.72	997	558
Dijkstra	0.431	15.82	1098.41	5.85	997	526

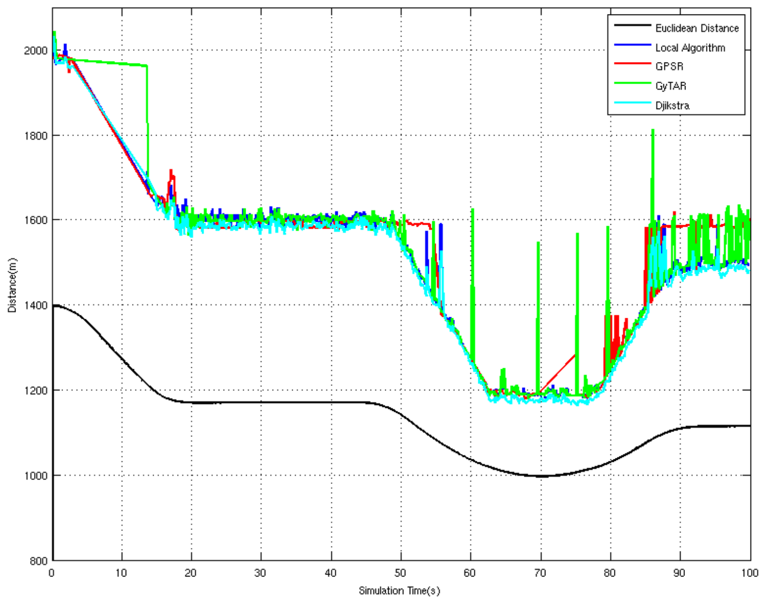

Figure 6.3: Packet travel distance compared to real distance between source and destination.

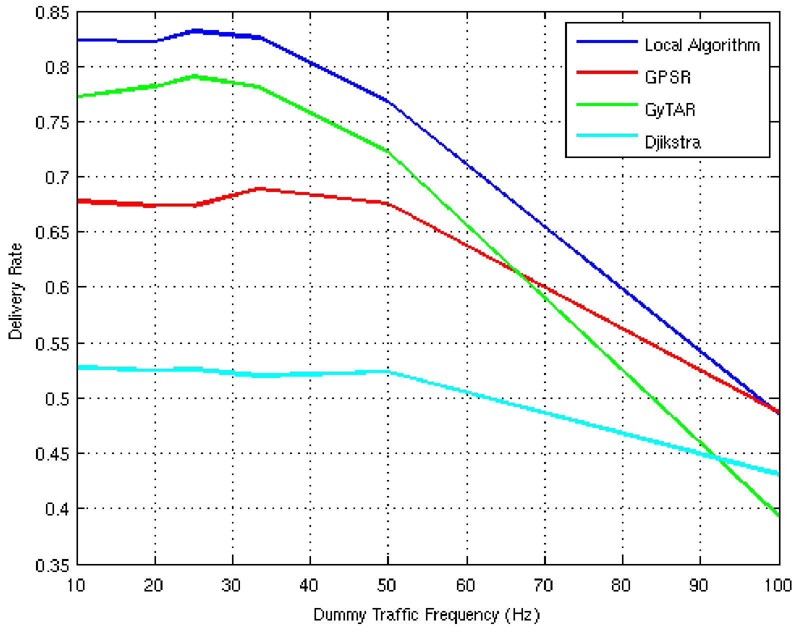

Figure 6.4: Packet Delivery Rate

7. CONCLUSION

Local routing algorithm is developed in this paper. VANET measurements are leveraged to choose the intersection for relaying purposes in order to assure fast and lossless packet delivery. A timeout mechanism causes retransmission of the data packets in VANET if the relaying vehicular node in the selected intersection cannot deliver the data packet in a given timeout duration. The local routing algorithm and reliable transmission protocol can be implemented in an urban area by using a commercial navigation map data, connectivity information of road IDs and an inexpensive GPS receiver.

The local algorithm is compared with respect to the shortest path algorithm and its effectiveness is simulated when the intersection zone’s data traffic is congested. The presented local routing algorithm may help to reduce congestion and to improve

REFERENCES

(ETSI), E.T.S.I. (2011a, June). *Intelligent Transport System (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to multipoint communications; Sub-part 1: Media independent functionalities*

(ETSI), E.T.S.I. (2011b, July). *Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part.*

Jerbi, M., R. Meraihi, S.-M. Senouci, and Y. Ghamri-Doudane (2006). Gytar: Improved greedy traffic aware routing protocol for vehicular ad hoc networks in city environments, *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks.* VANET ’06, New York, NY, USA, pp. 88–89. ACM.

Karp, B. and H. T. Kung (2000). GPSR : Greedy perimeter stateless routing for wireless networks, *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom ’00, New York, NY, USA, pp. 243–254.

Lochert, C., M. Mauve, H. Füßler, and H. Hartenstein (2005, January). Geographic routing in city scenarios, *SIGMOBILE Mob. Comput. Commun. Rev*, 9(1), 69–72.

Mo, Z., H. Zhu, K. Makki, and N. Pissinou (2006, July). MURU: A multi-hop routing protocol for urban vehicular ad hoc networks, *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking Services,* pp. 1–8.

Perkins, C., E. Belding-Royer, and S. Das (2003). Ad hoc on-demand distance vector (AODV) routing. *Internet RFCs*, 1-38.

Sommer, C., D. Eckhoff, R. German, and F. Dressler (2011, Jan). A computationally inexpensive empirical model of IEEE 802.11p radio shadowing in urban environments, *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on,* pp. 84–90.

Tsiachris,S, G.Koltsidas and F.-N.Pavlidou(2013). Junction-based geographic routing algorithm for vehicular ad hoc networks, *Wireless Personal Communications 71*(2), 955–973.

Zhao, J. and G. Cao (2008, May). VADD: Vehicle-assisted data delivery in vehicular ad hoc networks, *IEEE Transactions on Vehicular Technology 57* (3), 1910–1922.

APPENDICES

Appendix A.

```
/*
 * GuiManager.h
 *
 *  Created on: Sep 28, 2015
 *      Author: marian
 */

#ifndef GUIMANAGER_H_
#define GUIMANAGER_H_

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <vector>
#include <semaphore.h>
#include "ns3/core-module.h"
#include "Camera.h"
#include "GuiNode.h"
#include "ns3/Topology.h"
#define HEIGHT 480
#define WIDTH  640
#define GLUT_WHEEL_UP 3
#define GLUT_WHEEL_DOWN 4

typedef struct specialColor{
        int node;
        int r;
        int g;
        int b;
        std::string tag;
}sColor;

namespace ns3 {

class GuiManager {
public:
        GuiManager();
        virtual ~GuiManager();
        void init();
        void Idle(void);
        void MouseButton(int button, int state, int x, int y);
        void MouseMotion(int x, int y);
        void Keyboard(unsigned char key, int x, int y);

        void setNode(int id, double x, double y,double hdg);
        void setNodeStatic(int id, double x, double y,double hdg);
        void clearNodes();

        void drawSceene();
        void reshape(GLint width, GLint height);
        void displaysub(void);
        void start();

        void addSpecialColor(int node,std::string tag,int r,int g,int
b);
        void clearSpecialColor();

        bool GetPause();
        Topology *getTopology();


private:
         void drawCarShape();

private:
        int main_window;
        int sub_window;
        int screen_width;
        int screen_height;

        Topology topo;

        Camera cam;
        long time;

        int car_shape;
        int point_shape;

        sem_t lock;

    bool pause;

        std::vector<GuiNode> nodevec;
        std::vector<GuiNode> nodevec_static;
        std::map<int,sColor> sc_list;
};

}
#endif /* GUIMANAGER_H_ */
```

```
/*
 * GuiManager.cpp
 *
 *  Created on: Sep 28, 2015
 *      Author: marian
 */

#include "GuiManager.h"
#include "Polygons.h"

namespace ns3 {
GuiManager *current;

void wreshape(int w, int h)
{
	current->reshape(w,h);
}

void wdrawSceene()
{
	current->drawSceene();
}

void wMouseButton(int button, int state, int x, int y){
	current->MouseButton(button,state,x,y);
}

void wMouseMotion(int x, int y){
	current->MouseMotion(x,y);
}

void wKeyboard(unsigned char key,int x, int y){
	current->Keyboard(key,x,y);
}

void wdisplaysub(void){
	current->displaysub();
}

void wIdle(void){
	current->Idle();
}

GuiManager::GuiManager() {
	time=0;
	screen_width = WIDTH;
	screen_height = HEIGHT;
	current = this;
}

GuiManager::~GuiManager() {
	// TODO Auto-generated destructor stub
}

void GuiManager::Idle(void){
	glutPostRedisplay();
```

```
}


void GuiManager::MouseButton(int button, int state, int x, int y){
        switch(button){
        case GLUT_LEFT_BUTTON:
                if(state==GLUT_DOWN)
                        cam.setLPress(true,x,y);
                else
                        cam.setLPress(false,x,y);
                break;
        case GLUT_RIGHT_BUTTON:
                if(state==GLUT_DOWN)
                        cam.setRpress(true,x,y);
                else
                        cam.setRpress(false,x,y);
                break;
        case GLUT_WHEEL_UP:
                cam.zoomIn();
                break;
        case GLUT_WHEEL_DOWN:
                cam.zoomOut();
                break;
        default:
                break;
        }
        glutPostRedisplay();
}

void GuiManager::MouseMotion(int x, int y){
        cam.getMotion(x,y);
        glutPostRedisplay();
}

bool GuiManager::GetPause(){
        return pause;
}

void GuiManager::Keyboard(unsigned char key, int x, int y)
{
        switch(key){
        case 'p':
                pause = pause?false:true;
                break;
        }

}

void GuiManager::drawCarShape(){
        car_shape=glGenLists(1);
        glNewList(car_shape,GL_COMPILE);
        glBegin(GL_TRIANGLES);

        for (int i = 0; i < sizeof(car_data)/sizeof(double); i += 6)
        {
               glVertex3f(car_data[i], car_data[i + 1], car_data[i + 
2]);
                glNormal3f(car_data[i + 3], car_data[i + 4], car_data[i 
+ 5]);
        }
        glColor3f(0.55f,0.55f,0.55f);
        for (int i = 0; i < sizeof(car_win_data)/sizeof(double); i += 
6)
        {
                glVertex3f(car_win_data[i], car_win_data[i + 1], 
car_win_data[i + 2]);
                glNormal3f(car_win_data[i + 3], car_win_data[i + 4], 
car_win_data[i + 5]);
        }
        glEnd();
        glEndList();
}

void GuiManager::setNode(int id, double x, double y,double hdg)
{
        sem_wait(&lock);
        nodevec.push_back(GuiNode(id,x,y,hdg,car_shape));
        sem_post(&lock);
}


void GuiManager::setNodeStatic(int id, double x, double y,double hdg)
{
        sem_wait(&lock);
        nodevec_static.push_back(GuiNode(id,x,y,hdg,car_shape));
        sem_post(&lock);
}

void GuiManager::clearNodes()
{
        sem_wait(&lock);
        nodevec.clear();
        sem_post(&lock);
}

void GuiManager::reshape(GLint width, GLint height)
{
        screen_width = width;
        screen_height = height;
        glutSetWindow (sub_window);
        glutPositionWindow (screen_width-100, 0);
        glutReshapeWindow (100, screen_height);
        glutSetWindow (main_window);
        glViewport(0, 0, screen_width, screen_width);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(65.0, (float)screen_width / screen_height, 0.1, 
100000.0);
        glMatrixMode(GL_MODELVIEW);
}
```

```
void GuiManager::displaysub(void){
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glMatrixMode( GL_PROJECTION ) ;
        glPushMatrix() ;
        glLoadIdentity();
        glMatrixMode( GL_MODELVIEW ) ;
        glPushMatrix() ;
        glLoadIdentity() ;
        glDisable( GL_DEPTH_TEST ) ;
        glRasterPos3f( 0,0,.09 ) ; // center of screen. (-1,0) is
center left.
        glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
        char buf[300];
        std::stringstream s;
        s<<Simulator::Now().GetMilliSeconds();
        std::string str = s.str();
        for (int j = 0; j<str.size(); j++) {
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, str[j]);
        }
        glEnable( GL_DEPTH_TEST ) ;
        glPopMatrix();
        glPopMatrix();
        glutSwapBuffers();
        glutPostRedisplay();
}


void GuiManager::init(){
        int s=0;
        char **d=NULL;
        float aspect = (float)screen_width / (float)screen_height;
        glutInit (&s,d);
        glutInitWindowSize (screen_width, screen_height);
        glutInitDisplayMode ( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);

        main_window=glutCreateWindow ("GSU C2C Simulation Interface");
        glViewport(0, 0, WIDTH, HEIGHT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(30.0, aspect, 0.1, 100000.0);
        glMatrixMode(GL_MODELVIEW);
        glClearColor(0.7f, 0.9f, 0.7f,0.0f);
        glEnable(GL_DEPTH_TEST);
        glutDisplayFunc(wdrawSceene);
        glutReshapeFunc(wreshape);
        glutMouseFunc(wMouseButton);
        glutMotionFunc(wMouseMotion);
        glutKeyboardFunc(wKeyboard);
        glutIdleFunc(wIdle);

        sub_window=glutCreateSubWindow(main_window,500,0,140,480);
        glViewport(0, 0, screen_width, screen_height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(120.0, aspect, 0.1, 100000.0);
        glMatrixMode(GL_MODELVIEW);
        glClearColor(0, 0, 0,1);
        glEnable(GL_DEPTH_TEST);
        glutDisplayFunc(wdisplaysub);

        glutIdleFunc(wIdle);
}

void GuiManager::start(){
        sem_init(&lock,0,1);
        init();
        glutSetWindow(main_window);
        topo.prepare();
        drawCarShape();
        glutMainLoop();
}

void GuiManager::drawSceene(){
        sem_wait(&lock);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        cam.get();
        topo.draw();
        glEnable(GL_LIGHTING);
        glEnable (GL_LIGHT0);
        glEnable(GL_COLOR_MATERIAL);
        for(int i=0;i<nodevec.size();i++){
                std::map<int,sColor>::iterator it =
sc_list.find(nodevec.at(i).id);
                if (it != sc_list.end()){
                        nodevec.at(i).draw(it->second.r,it-
>second.g,it->second.b,it->second.tag);
                }
                else{
                        nodevec.at(i).draw();
                }
        }
        for(int i=0;i<nodevec_static.size();i++){
                nodevec_static.at(i).draw();
        }
        glDisable(GL_LIGHTING);
        sem_post(&lock);
        glutSwapBuffers();
}

Topology *GuiManager::getTopology(){
        return &topo;
}
void GuiManager::addSpecialColor(int node,std::string tag,int r,int
g,int b){
        sColor newcolor;
        newcolor.node = node;
        newcolor.tag = tag;
        newcolor.r = r;
        newcolor.g = g;
        newcolor.b = b;
        sem_wait(&lock);
        sc_list.insert(std::pair<int,sColor>(node,newcolor));
        sem_post(&lock);
}

void GuiManager::clearSpecialColor(){
        sem_wait(&lock);
        sc_list.clear();
        sem_post(&lock);

}
}
```

BIOGRAPHICAL SKETCH

Çağdaş Yaman graduated from Izmir Atatürk Science High-School. He received his B.S degree in Computer Engineering in 2011 from Galatasaray University. He is currently a M.S student in Computer Engineering in Institute of Science and Engineering at Galatasaray University, Istanbul.

PUBLICATIONS

- A. U. Peker, T. Acarman, Ç Yaman and E. Yüksel, "Vehicle localization enhancement with VANETs," *2014 IEEE Intelligent Vehicles Symposium Proceedings*, Dearborn, MI, 2014, pp. 661-666.
- Y. Peks¸en, Ç Yaman, T. Acarman and A. U. Peker, "Multi-channel operation and GNSS correction issues," *ITS Telecommunications (ITST), 2013 13th International Conference on, Tampere*, 2013, pp. 468-473.
- R. Scopigno, A. Autolitano, T. Acarman, Ç. Yaman and S. Topsu, "The potential benefits of on-board Li-Fi for the cooperation among vehicles," *2015 17th International Conference on Transparent Optical Networks (ICTON)*, Budapest, 2015, pp. 1-6.
- T. Acarman, Ç. Yaman, Y. Peksen and A. U. Peker, "Intersection-Based Routing in Urban VANETs," *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Las Palmas, 2015, pp. 1087-1092.