Reducing Interferences in VANETs

Dmitry Zelikman and Michael Segal, *Senior Member, IEEE*

***Abstract*—Mobile Ad-Hoc Networks (MANETs) are networks which are created on-the-fly (Ad-Hoc) between various mobile nodes, and do not require an infrastructure. The mobile nodes can move around and the network would automatically reconfigure itself to allow connectivity. Vehicular Ad-Hoc Networks (VANETs) are a subclass of MANETs that is expected to have a key role in the Intelligent Transportation Systems (ITSs) of the future. VANETs provide vehicle-to-vehicle and vehicle-to-roadside communication in order to support safety as well as comfort applications. Despite being a subclass of MANETs, VANETs have fundamentally different behavior [1]. This work presents a scheme consisting of a MAC protocol and a clustering algorithm designed for reducing interferences in VANETs. Our scheme, which is intended for safety applications in highway environments, employs dynamic multi-hop clustering, allows better utilization of network resources and improves network performance.**

Index Terms* —*VANET; Multihop Network; Clustering Infrastructure; Interference Reduction;

I. Introduction

Vehicular Ad-Hoc Networks (VANETs) are a class of wireless networks that is expected to have a key role in the Intelligent Transportation Systems (ITSs) of the future. Already in recent years the US FCC and European ETSI have allocated spectrum for such systems, and an IEEE communications standard for them is under development. VANETs provide vehicle to vehicle and vehicle to roadside communication in order to support two main types of applications: safety applications such as road hazard notification and sending emergency messages from an accident site, and comfort applications such as advertisements, parking space availability, traffic estimation and traffic-jam notifications.

Despite being a subclass of Mobile Ad-Hoc Networks (MANETs), VANETs have fundamentally different behavior [1] in that their nodes are limited to move along roads, have no power constraints, have small network diameter, and may undergo rapid topology changes, for example when cars bypass each other in an intersection, or when highways split before an interchange. VANETs also have different requirements for routing. While MANETs usually use topology based table or source routing algorithms [2], these are not applicable for VANETs because of their highly dynamic nature. Most VANET routing algorithms use geographic based routing [3] and opportunistic carry-and-forward techniques [3,4] to overcome this challenge.

In this work we propose a scheme for reducing interferences in VANETs in highway environments. Unlike some other schemes which assume that only some percentage of the vehicles transmit safety messages at any given time [5], our scheme guarantees channel access for all of the vehicles, allowing all nodes constantly transmit safety data and enabling even the most demanding safety applications such as crash avoidance [6].

Dmitry Zelikman is with the Dept. of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel. Email: dmitryz@post.bgu.ac.il.

Michael Segal is with the Dept. of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel. Email: segal@bgu.ac.il.

We use the *Highway Model* [7] and treat the vehicles as moving on a single line. Our scheme then uses unique clustering and MAC methods to achieve low interference between the vehicles. Unlike most other works which measure interference using the *Receiver Centric Model* [7], our scheme uses the original (to the best of our knowledge) *Neighborhood Interference Model*. Topologies which have low interference calculated using the *Receiver Centric Model*, may have higher interference when the *Neighborhood Interference Model* is used. Finally, we provide simulative analysis of our scheme and discuss its performance in different scenarios.

This paper is organized is follows. In the next section we briefly explain the previous and related work to our research. Section III contains the system model. Definitions and notations are given at Section IV. We provide MAC layer and Clustering layer descriptions in Sections V and VI, accordingly. Section VII contains treatment of special cases. Simulations of our scheme are described at Section VIII. Finally, we conclude at Section IX.

II. Related Work

Blum et al. [1] discuss many characteristics of VANETs. They use simulations, and provide some implications for the development of VANET communication systems.

Naumov et al. [8] study the behavior of routing protocols in VANETs by using mobility information obtained from a vehicular traffic simulator based on real road maps of Switzerland. Haas et al. [6] investigate the communication requirements for crash avoidance in vehicular networks.

In order to support the above safety requirements, an appropriate Media Access Control (MAC) scheme is needed. Several MAC schemes for VANETs were proposed in the literature. Ray et al. [9] describe DCR (dynamic channel reservation), a TDMA scheme which allocates vehicles unique slots in 2-hop radius and thus avoids collisions, and allows fast broadcasting of emergency messages. In a continuation paper [10], they describe a dynamic scheme for dividing channels between the two opposite lanes. Mittag et al. [11] compare single-hop and multi-hop beacon dissemination types on a static VANET network for both sparse and dense vehicle topologies, and uses MHVB as the dissemination Protocol. Resta et al. [12] examine multi-hop emergency message dissemination delay using several dissemination strategies.

In addition to media access techniques, there are several results for power control and clustering directed at reducing interference in vehicular networks. Artimy et al. [13] propose a scheme to control the transmit power of vehicles based on an estimation of the local vehicle density. They perform simulations and study the impact of their algorithm on the connectivity of the network. Moreno et al. [14,15] describe a scheme of fairness that decreases node's power if it prevents other nodes to transmit sufficient amount of safety information. Chigan et al [16] describe a method to iteratively adjust the transmit power of several static directional antennas mounted on the vehicle to achieve a collision-free transmission, and Mittag et al. [17] propose a segment-based power adjustment approach based on distributed vehicle density estimation. Allouche and Segal [18] present the Distributed Construct Underlying Topology (D-CUT) algorithm, a self-organized algorithm whose aim is to provide efficient and reliable hierarchical topology by minimizing the interference between network participants. Bononoi et al. [5] propose a distributed dynamic clustering algorithm which creates a virtual backbone in the vehicular network.

Khabbazian et al. [19] discuss the problem of reducing interference on the highway model, and generalize the problem to two dimensions. Tan et al. [20] study the problems of average and maximum interferences on the highway model. Their algorithm for minimizing the maximum interference achieves the lower maximum-interference bound for the highway model which was shown by Rickenbach et al. [7] to be $\sqrt{d}$ where d is the maximum node degree in the system. Kranakis et al. [21] observe the problem of uniformly distributing sensor nodes on the highway model and connecting each node to the two nodes immediately beside it on the highway. They prove a tight bound on the expected interference in that case to be $\Theta\left(\sqrt{\log n}\right)$ where n is the number of nodes in the system.

III. System Model

Let us consider safety applications in a long stream of cars. These applications will generally require one of the three message dissemination types given in Table I. The first two types are good for handling emergencies while the third type is meant for avoiding emergencies by continuously transmitting each vehicle's position and speed to its neighborhood [6].

Our goal in this work is to reduce interference in the above stream of cars in such a way that would allow using all three of the possible message dissemination types while keeping low delay in the network. It has been shown ([6]) that CSMA/CA based MAC is not efficient enough to handle the high message frequency of the smart driving application, but a TDMA based MAC can theoretically do it. In order to achieve our goal, we propose a scheme consisting of two layers: a TDMA based MAC layer designed for fast multihop channel access, and a clustering layer which performs topology control and reduces interference while keeping the network connected. We also use two separate channels: one for communications inside of clusters and, another of communications between clusters.

We model the traffic using the *Highway Model* [7]. This model treats a road as a one-dimensional object. All vehicles on the highway have only one positional coordinate which is their position along the highway. To enable use of this model, all nodes are assumed to be equipped with a GNSS system. Finally, in order to measure the effectiveness of our solution, an Interference Model is needed. Several different interference models can be found in the literature [7]. Most notable of them are the *Sender Centric*, *Receiver Centric* and *Edge Interference* Models. In this work we use a combination of the sender and receiver centric models named *Neighborhood Interference*. To the best of our knowledge, we are the first to use this metric. In this model, the interference of node *v* is determined by the number of other nodes that cannot transmit altogether with *v*.

IV. Definitions & Notations

1. **Outgoing Neighbor:** For an arbitrary node **v**, any node **u** that hears transmissions generated by **v** is an outgoing neighbor of **v**. Each node may have many outgoing neighbors.

TABLE I. POSSIBLE MESSAGE DISSEMINATION TYPES

#	Dissemination Type	Characteristics
1	Forward Moving Message (e.g. make way for an ambulance)	• Long distance (Chain of cars - Multihop) • Low latency • Low message frequency (<1 Msg/Sec)
2	Backwards Moving Message (e.g. accident ahead)	
3	Smart Driving / Crash Avoidance	• Relatively short distance (1-2 Hops) • Very low latency • High message frequency (~5-10 Msg/Sec)

2. **Incoming Neighbor:** For an arbitrary node **v**, any node **u** whose transmissions are heard by **v** is an incoming neighbor of **v**. Each node may have many incoming neighbors.
3. **Neighbor:** Either an incoming or an outgoing neighbor of a node.
4. **Full Neighbor:** For an arbitrary node **v**, a node which is both an incoming neighbor and an outgoing neighbor of **v** will be designated a Full Neighbor of v. Each node may have several full neighbors.
5. **Outgoing Neighborhood:** $\boldsymbol{N_v^+}$ – The group of all outgoing neighbors of node v including node v itself.
6. **Incoming Neighborhood:** $\boldsymbol{N_v^-}$ – The group of all incoming neighbors of node v including node v itself.
7. **Interference Neighborhood:** $\boldsymbol{IN_v}$ – The group of all nodes which should be silent when node v transmitts. This includes all outgoing neighbors of v because its transmission may be meant for them, and all of their incoming neighbors which may interfere v's transmission. For clarity of explanation $\mathrm{IN_v}$ will include v itself as well:

$$IN_v = N_v^+ \cup \{i \in V \mid i \in N_j^- \land j \in N_v^+\} \qquad (1)$$

8. **Size of Interference Neighborhood:** L_v - Number of nodes in IN_v.
9. **Successor:** A node v is a successor of node u if it is located after node u in the direction of traffic, and no other node is located between them, i.e., node v is the node immediately following node u. On Fig. 2 below, node B is a *Successor* to node C.
10. **Predecessor:** A node u is a predecessor of node v, if and only if node v is a successor of node u.
11. **Friends:** Two nodes u and v will be friends, if there is no other node which is positioned between them on the highway. Each node may have up to two friends which are his *Successor* and *Predecessor* nodes. Should the system contain several nodes with the exact same position, ties may be broken arbitrary so that each node will have no more than two friends. Friendship relations do not imply neighboring relations.
12. **Edge Node:** A node with less than two friends will be designated as an *Edge Node*. Such nodes are created when the distance between two nodes is not covered by the current transmission power of the nodes. An edge node which has only a *Successor* will be named a *Head Node*, and an edge node which has only a *Predecessor* will be named a *Tail Node*.

We will now show how we build our MAC based on the above definitions.

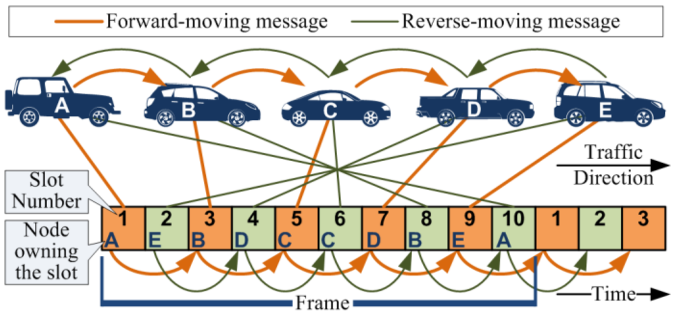

Fig. 2 – Slot Assignment Scheme

V. MAC Layer Description

A. The Dual-Slots MAC Layer

TDMA allows simple and contention-less channel access and allows avoiding the hidden terminal problem by allocating unique time slots in two-hop radius around every node. However, when the destination of a message is several hops away, the message has to wait in every intermediate node while this node's TX slot arrives. We eliminate this delay by using the Dual-Slots layer. The Dual-Slots scheme assumes that each node may forward messages only to its Friends on the highway (Fig. 2). To enable multi-hop forwarding with minimal delay, every node selects two slots – an odd slot is used to forward messages in the direction of the traffic, and an even slot is used to forward messages in reverse direction. The number of slots per frame is determined by the clustering algorithm during the clustering process, and is given as input to the MAC Layer.

The slots are assigned to nodes in such a way that when a node receives a message traveling in a certain direction, it would immediately be able to forward the message to its next hop in the same direction (see Fig. 2). In the following notations slots are numbered from 1, and nodes are numbered starting from 0.

Consider a cluster with N nodes and $2K$ slots in its TDMA frame. Vehicle i, ($i \in [0, N-1]$) would then select the odd slot ($(2i+1) \ mod \ 2K$) for forwarding messages to its *Predecessor* and the even slot ($2k - (2i \ mod \ 2K)$) for forwarding messages to its *Successor*. The set of two slots belonging to a single node will be referred to as a *Dual-Slot*. The set of K consecutive *Dual-Slots* as defined above is designated to be a *Frame* in our Dual-Slots MAC layer. The number of Dual-Slots in a Frame is the *Frame Length*. Fig. 2 summarizes the above slot assignment scheme for 5 nodes.

B. Swapping Slots

Supporting vehicle bypassing is a basic requirement in multi-lane roads. Vehicles bypassing each other are regarded as nodes swapping their positions and their *Friends* on the highway. In order to keep the frame structure intact, such nodes must also swap their selected slots. Nodes detect changes to their *Friends* by receiving beacons from new friends. Since all the nodes have to transmit beacons, the time-slot swapping itself can only occur at the end of each frame. Therefore, at the end of each frame every node compares the position reported in the beacons of its *Friends* and decides on the slots it will use on the next frame.

For the scheme to work correctly, we assume a vehicle may bypass only one other vehicle in the time span between two consecutive beacons. This assumption of high beaconing rate is reasonable since our scheme targets the high beacon rates required by smart driving applications. For example, a beaconing rate of 5 $Beacon/Sec$, and speed difference of $180kph$, would mean that the bypassing vehicle bypasses only $10m$ between two consecutive beacons. The assumption is also required since vehicles use beacons to adjust their power so that transmissions will reach their *Friends*. If the beacon rate is too low, rapid changes in distances between nodes may break the network.

VI. Clustering Layer Description

A. Strategies

The clustering algorithm works by splitting the long stream of vehicles on the *Highway Model* into short "chains of vehicles" while keeping in mind our goal of interference reduction. These chains are our clusters. To achieve this clustering, each node selects one of two strategies:

- *Far Strategy* – Connect to both *Friends* (the *Predecessor* and the *Successor*) on the highway.
- *Near Strategy* – Connect to the closer of the two possible *Friends* and use a separate channel to communicate with the farther one. When a node uses this strategy, we designate its farther neighbor to be named its *Partner*.

For ease of notation we shall designate channel A to be the channel for intra-cluster communication and channel B to be the inter-cluster channel. Channel A will use our Dual-Slots MAC protocol; while channel B will use regular IEEE 802.11p MAC in order to allow random channel access. We are not aware of any previous works using this configuration.

We will now describe the clustering layer, namely the GIM (Greedy Interference Minimization) algorithm. GIM contains several processes for construction and maintenance of clusters.

B. Basic Network Initialization

We begin by describing the simpler case of simultaneous network initialization of all nodes. In this case nodes will not receive beacons from nodes which are already cluster members, and the only option available is the creation of new clusters. The more complicated case which includes joining existing clusters is described after all of its perquisites in subsection F.

Before the beginning of cluster formation, vehicles are not aware of their frame length, but they do know the predefined time-slot size and can choose a strategy. All vehicles begin with the *Far* strategy, meaning that every node is connected to its *Friends* on the highway. To achieve this state, vehicles access the channel in a simple slotted ALOHA scheme until each vehicle learns of its two *Friends* (except for *Edge Nodes* which detect only one *Friend*) and adjusts its transmission power to reach the one farther away. At this time, none of the nodes is a cluster member yet, and this fact is marked in the beacons they transmit.

Each cluster is allowed to have a different frame length, and has no hardcoded limit for the number of members. Once an arbitrary node *v* is connected to its available *Friends*, it begins to send REPLY messages to nodes whose beacons it receives. These nodes are not necessarily *full neighbors*, but just nodes whose TX range covers our node *v*. Using information from such REPLYs it becomes easy for *v* to calculate its Interference Neighborhood, IN_v. Notice that the size of IN_v which is denoted L_v is the number of dual-slots required in the frame from the point of view of node *v*. Since all of the nodes use slotted ALOHA at this stage, new non-cluster-member nodes may simply join into the process. This may occur if not all of the nodes are started simultaneously.

Each node v shares its value of L_v with the other nodes in IN_v. All nodes constantly monitor changes to their interference neighborhood and keep reference to the node $m \in IN_v$ with the largest L_m. Once the ID of node m becomes stable for several iterations, each node sends a SELECT message to m. Note that every node may select a different node to be m. The SELECT is sent by each node v to node m that it had individually selected.

Each node v which receives a SELECT message addressed to it, switches to the *Near* strategy which results in splitting the long connected sequence of nodes into clusters. Each cluster has two edge nodes which are responsible for communications with neighboring clusters using channel B. Once some of the nodes have switched to the *Near* strategy, the interference neighborhood of these nodes will become smaller, bringing us closer to our goal of minimizing the interference on the highway. The smaller interference also means that in order to decide on the desired frame size, all nodes have to recalculate their *Interference Neighborhood* before proceeding to the next step in our algorithm.

We now observe the *Tail Nodes* of the newly created clusters. A node knows that it is a tail node since its updated *Interference Neighborhood* does not contain a successor within transmission range on channel A. Let v be a *Tail Node*. Once the Interference Neighborhood IN_v is updated, it again selects the node $m \in IN_v$ with the largest L_m and sets L_m to be its frame size. After selecting the frame size, v takes the first time-slot (*Dual Slot*) to itself, and sends a token with the selected number of slots in the cluster allowing each node to select the correct time-slot.

C. Cluster Merging

Clusters will be merged only if one or both of the following conditions are met:

- **Condition 1 (Fig. 3a):** Clusters which are connected to each other on channel B come within range of their edge node's channel A transmission.
- **Condition 2 (Fig. 3b):** Two nodes which are partners on channel B (nodes r and s in Fig. 3) see a common third node (node t in Fig. 3) on channel B. This means that their channel B edge is completely covered by another channel B edge.

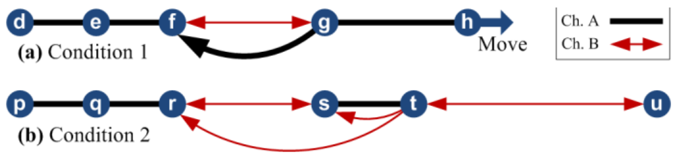

Fig. 3 – Cluster Merging Conditions

Detection of Condition 2 is trivial. For detecting Condition 1, the edge nodes of neighboring clusters each use their one available slot. Recall that in the dual-slots MAC, each node has two slots - one for each direction. The edge nodes of a cluster therefore have one of their two slots vacant. During the vacant slot, each edge node waits a random backoff and transmits a probe packet using its currently set TX power level. During the backoff, the node listens to the media. An edge node which receives a probe packet, successfully detects condition 1.

For clarity of exposition, we begin by describing the cluster merging process of two clusters. This process will be extended in subsection G to support a cascading case where several clusters merge and split simultaneously.

Upon detecting one of the merging conditions, the detecting edge node informs its partner of the detection and initiates the cluster merging process which includes decision on the new frame size, and notification of the nodes in both merged clusters about this size and about the new time-slots they should use for transmitting data. The frame size of the merged cluster is the sum of the frame sizes of both original clusters. Each edge node publishes this new size to its own cluster using an UPDATE message which includes: the new frame size, a slot offset parameter and the merging time.

To continue our explanation, let us assume that the traffic on our highway goes from left to right, then the merging clusters will be designated from left to right as $C1$ and $C2$. In reference to Fig. 3, cluster $C1$ includes the head node f and cluster $C2$ includes the tail node g. As mentioned above, the UPDATE messages contain a *Slot Offset* parameter. This parameter denotes the offset nodes will use when selecting their odd slot. The even slot can then be calculated using the odd slot and the frame size. For $C1$ the *Slot Offset* is 0 indicating that nodes in this cluster keep their odd slot indices. For $C2$ the *Slot Offset* is the original frame size of $C1$ indicating that all nodes in $C2$ should use odd slots following those of nodes in $C1$.

Finally, the *Merging Time* included in the UPDATE message is the start time of the next frame in the new cluster. This is decided according to the time required for all nodes in both $C1$ and $C2$ to receive the UPDATE message, and for newer update messages to return. i.e., the merging time in dual-slots is twice the number of nodes in the cluster. Before the *Merging Time*, all nodes continue using their original cluster configuration, and after it, all nodes will necessarily switch to the new configuration. The *Merging Time* includes time for newer UPDATE messages to return in order to allow cascading merges as described in subsection G below.

D. Cluster Splitting

Since cluster splitting produces smaller clusters with smaller frame sizes, the cluster splitting process runs all the time in an attempt to optimize the clusters. Each time the Interference Neighborhood of an arbitrary node v changes, node v finds the node $m \in IN_v$ with the largest L_m of all of the nodes in IN_v, and sends a SELECT message to it. Upon receiving a SELECT message, node m decides whether or not to switch strategy to *Near*. The decision is made by checking the cluster merging conditions to avoid unstable states of a cluster splitting and merging loop. Should a strategy change be decided upon, node m will transmit an UPDATE message informing the other nodes in the cluster of the change.

E. Joining an Existing Network

During startup our nodes listen to the media, and upon detecting an existing cluster may attempt to join it. The decision on joining an existing cluster is described in section F below. For now, let us assume that our new node v has decided to join an existing cluster. In order to do that, v chooses the nearest cluster member and uses channel B to send a join request JOIN_REQ message to it. A node u which receives a join request sends an UPDATE message to its cluster adding a new slot to the MAC frame, and replies v with a JOIN_RSP message telling it its new cluster parameters (frame size and allocated slot indices).

If v is located between two clusters, its joining to one of them may activate *Merging Condition 1* and yield the merging of the clusters. In order to improve performance, we identify this situation and use a single UPDATE to both merge the clusters, and add a slot for v. The detection is done by letting v participate in *Merging Condition 1 Detection* (as described in subsection C) on behalf of node u.

F. Full Network Initialization

Upon startup an arbitrary node v begins to listen to the media. If v receives beacons only from cluster-member nodes, v will initiate the cluster joining procedure. If new nodes are detected during the cluster joining process, v will not reply to their beacons until it becomes a cluster member. Another case is that node v receives beacons only from non-cluster-member nodes. In this case v will initiate the cluster construction procedure. If during cluster construction, node v receives a beacon from a cluster-member node, it will abandon the construction process and will join the detected cluster. The last possibility is for node v to receive beacons from both cluster members and non-cluster-member nodes. In this case v will switch to the *Near* strategy and act according to the nodes closer to itself.

G. *Cascading Cluster Operations*

In this section we handle the case of several simultaneous clusters splitting, merging and joining operations. Notice that all of these operations use the same simple UPDATE message, and that this message is disseminated in both directions within the affected clusters. This means that if multiple events occur in different places in a cluster, the UPDATE messages of these events are bound to meet at some node, and no more than two UPDATEs will meet at any given time.

Let us now observe the case of simultaneous merging and splitting. This process is identical to that of simultaneous multiple merging, except for the calculation of the new frame size at the node where two UPDATE messages meet. If one of the received update messages increases the frame size while the other one decreases it, then one of the updates resulted from cluster merging and the other from cluster splitting. The new frame size FS_{new} will then be calculated using (2) where FS_{max} is the larger of the receive frame sizes, FS_{min} is the smaller of the receive frame sizes, and FS_c is the current frame size.

$$FS_{new} = FS_{max} - (FS_c - FS_{min}) \qquad (2)$$

VII. Cluster Size and Interference Bounds

Being based on the property that each node must route all of its messages through its *Friends*, our algorithm as described so far has a weakness allowing creation clusters with either $O(n)$ nodes, $O(n)$ slots, or both. This occurs in chains where distances between nodes are constantly increasing. According to our first cluster merging rule, such chains will not be broken into clusters. Depending on the increase in distance in each hop, several situations may occur. If the distance increases exponentially, the first node in the chain will hear all of the following nodes. The interference neighborhood of all nodes will then be equal to the size of the cluster. This situation is designated as an *Exponential Chain*. If the increase in distance will be such that each node is heard by two nodes behind it, the interference neighborhood size will be constant, but the length of the chain will be limited only by the max TX radius of the nodes. We name this case a *Fibonacci Chain*. The next case is where each node is heard by a single node behind it, but the distances along the chain still increase. This case gives a constant interference neighborhood of 5 nodes, but can yield an infinite chain of nodes. We name this case as *Simple Chain*. It is possible to expand the above *Fibonacci* case to cases where each node is heard by k nodes behind it. We name these cases *k-Fibonacci* chains. Such chains are constructed of two parts: a *Simple Chain* of nodes (*Simple Part*) which are closely packed together and a *Fibonacci Part* where distances increase in such a way that the TX ranges of the nodes reach into the *Simple Part*. The *Simple Part* in this case will contain $O(k)$ nodes. Working in our favor is the fact that Exponential chains, Fibonacci chains and the *Fibonacci Part* of the *k-Fibonacci Chain* quickly increase their required

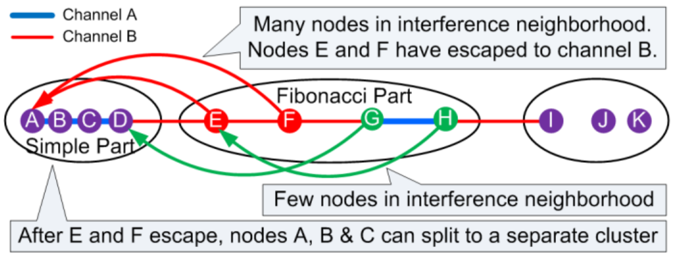
Channel A
Channel B
Many nodes in interference neighborhood. Nodes E and F have escaped to channel B.
Fibonacci Part
A
B
C
D
E
F
G
H
I
J
K
Simple Part
Few nodes in interference neighborhood
After E and F escape, nodes A, B & C can split to a separate cluster

Fig. 4 – Escaping Nodes Illustration

transmission radius, so the number of nodes in such chains is logarithmic in the maximum transmission radius, which according to [22] we can safely assume will not be greater than $R = 2km$. We therefore need to handle only two edge cases: The *Simple Chain* which has $O(1)$ *Interference Neighborhood* and $O(n)$ nodes, and the *k-Fibonacci* chain which has $O(k)$ *Interference Neighborhood* and $O(k) + Log(R)$ nodes. The *Exponential Chain* only has $Log(R)$ nodes, and we assume this value is acceptable for the expected transmission radius as stated above.

Let us begin by handling the *Simple Chain* case. Simple chains cannot be split by making two neighboring nodes communicate on a different channel, however splitting is possible if two adjacent edges are converted to the second channel. A mechanism to do this can be easily added. If a node detects that it is a part of a long chain (using the *counting token* sent as part of the clustering process), and its predecessor and successor nodes do not hear each other, the node will switch to communicating with both of them on channel B. This solution limits the number of nodes in the cluster, and since *Simple Chains* have a small *Interference Neighborhood*, their case can be considered closed.

The *k-Fibonacci* chains present a more serious challenge. Not only can the cluster size be large, but the *Interference Neighborhood* can be large as well. We should notice that the number of nodes in the *Fibonacci Part* of the chain is very limited, and if all of these nodes will work on a separate channel than the nodes in the *Simple Part*, our problem will be solved.

In order to implement the above solution we introduce a new node strategy: the *Escape Strategy*. Nodes with interference neighborhood size greater than 11 (which is actually Log(R) where R is the maximum TX radius of *2km*), will panic and switch to the *Escape* strategy. Nodes in this strategy use channel B to communicate with both of their *Friends*, and are not subjected to the cluster merging conditions. These nodes will continue using channel B until the network conditions would allow safe return to channel A. Nodes using the *Escape* strategy remain part of the cluster nearest to them. They are not affected by the cluster merging rules and cannot be used to split clusters. Figure 4 illustrates a possible use-case of the *Escape Strategy*.

It remains to describe the manner in which escaping nodes will return to using channel A. Consider the situation in Fig. 4. After nodes E and F have escaped, the forward slot of node D remains unused. Node F can use this slot to send probe messages on channel A. The other nodes will respond to the probes using regular intra-cluster routing and node F will be able to detect the size of its interference neighborhood on channel A. Once node F decides it is safe to return, node E will take over the vacant slot and begin detecting channel conditions. As a general rule, the escaping node farthest from the *simple part* of the *k-Fibonacci* chain is the first to return.

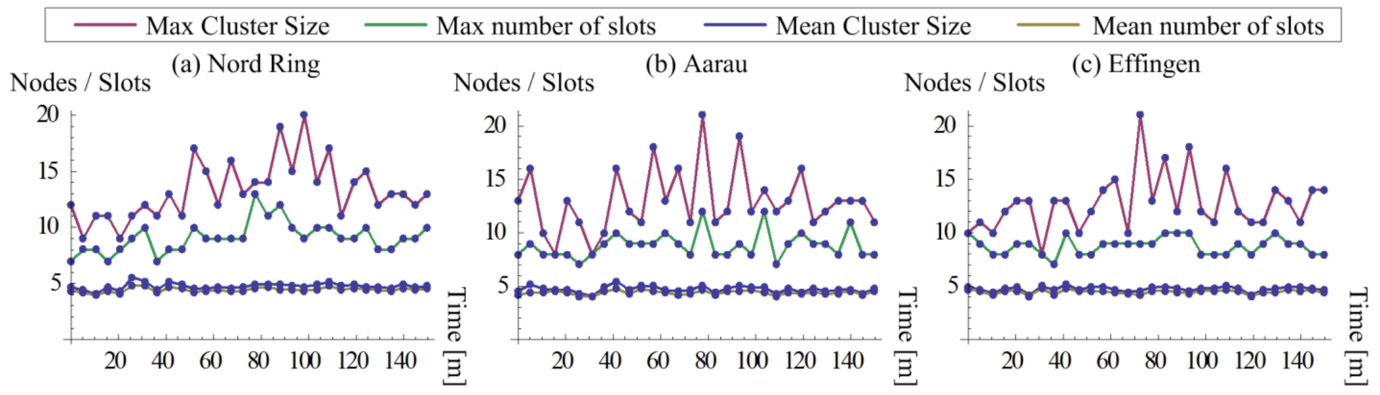

Fig. 5 – Simulation Results

VIII. Simulations

In order to determine the behavior of GIM on realistic traffic, we used traffic traces created by Naumov et al. [8] and made publically available on their website. From these traces, we have selected three highways near Zurich in Switzerland, and for each highway simulations were performed using a specialized simulator developed using *Mathematica*. Due to the Stationarity property of our clusters, movement of vehicles was simulated simply by taking snapshots of the highway Every 5 minutes. Figure 5 shows the simulation results on the three highways described above. It can be seen that for all vehicle densities the average cluster size and average number of dual-slots in each cluster are near the values expected form a simple chain. The maximum values on the contrary tend to be quite high with up to 13 slots per cluster and 20 vehicles per cluster. These can be explained by the existence of Fibonacci chains described above and by the difference of the highway model from reality – real highways usually have several lanes which allow packing more vehicles on a short distance when translated to the highway model.

IX. Conclusions

We have studied the problem of interference minimization for safety applications in Vehicular Ad-Hoc Networks using the highway model. We have created a new multi-hop clustering approach which is fully distributed, uses local decisions, and does not require a cluster-head selection. Our GIM scheme allows contention-less channel access for both collision avoidance and broadcasting applications. The interference measurements in our work use the *Neighborhood Interference Model* which is more demanding (interference wise) than the *Receiver Centric Model* used in many other works [7,19,20]; however, similarly those same works, we assumed a simple transmission-range based model for determining if nodes correctly receive each other. A direction for future work could be extending our scheme with more advanced RF models which take SINR into account.

References

[1] Jeremy J. Blum, Azim Eskandarian, and Lance J. Hoffman, "Challenges of Intervehicle Ad Hoc Networks," IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 4, December 2004.

[2] Elizabeth M. Royer and Chai-Keong Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," IEEE Personal Communications, 1999.

[3] Kevin C. Lee, Uichin Lee, and Mario Gerla, "Survey of Routing Protocols in Vehicular Ad Hoc Networks," Advances in Vehicular Ad-Hoc Networks: Developments and Challenges, IGI Global, October 2009.

[4] Antonios Skordylis and Niki Trigoni, "Delay-bounded Routing in Vehicular Ad-hoc Networks," in MobiHoc'08, 2008.

[5] Luciano Bononi and Marco Di Felice, "A Cross Layered MAC and Clustering Scheme for Efficient Broadcast in," in MASS'2007, 2007, pp. 1-8.

[6] Jason J. Haas and Yih-Chun Hu, "Communication Requirements for Crash Avoidance," in VANET, Chicago, Illinois, USA, 2010.

[7] P. von Rickenbach, S. Schmid, R. Wattenhofer, and A. Zollinger, "A robust interference model for wireless ad-hoc networks," in Parallel and Distributed Processing Symposium, 2005, p. 8.

[8] Valery Naumov, Rainer Baumann, and Thomas Gross, "An Evaluation of InterVehicle Ad Hoc Networks Based on Realistic Vehicular Traces," in MobiHoc '06, 2006, pp. 108-119.

[9] Ray K. Lam and P. R. Kumar, "Dynamic Channel Reservation to Enhance Channel Access by Exploiting Structure of Vehicular Networks," in VTC, 2010.

[10] Ray K. Lam and P. R. Kumar, "Dynamic Channel Partition and Reservation for Structured Channel Access in Vehicular Networks," in VANET, Chicago, Illinois, USA, 2010.

[11] Jens Mittag, Florian Thomas, Jérôme Härri, and Hannes Hartenstein, "A Comparison of Single- and Multi-hop Beaconing in VANETs," in VANET, Beijing, China, 2009.

[12] Giovanni Resta, Paolo Santi, and Janos Simon, "Analysis of MultiHop Emergency Message Propagation in Vehicular Ad Hoc Networks," in MobiHoc, Monteeal, Quebec, Canada, 2007.

[13] Maen M. Artimy, William Robertson, and William J. Phillips, "Assignment of Dynamic Transmission Range Based on Estimation of Vehicle Density," in ACM VANET, 2005, pp. 40-48.

[14] M. Moreno, P. Santi, and H. Hartenstein, "Fair sharing of bandwidth in VANETs," in ACM VANET, 2005, pp. 49-58.

[15] M. Moreno, P. Santi, and H. Hartenstein, "Distributed Fair Transmit Power Adjustment For Vehicular AdHoc Networks," in IEEE SECON, 2006, pp. 479-488.

[16] C. Chigan and J. Li, "A delay-bounded dynamic interactive power control algorithm for vanets," in IEEE International Conference on Communications, 2007, pp. 5849-5855.

[17] Jens Mittag, Felix Schmidt-Eisenlohr, and Moritz Killat, "Analysis and Design of Effective and Low-Overhead Transmission Power Control for VANETs," in VANET, San Francisco, California, USA, 2008.

[18] Yair Allouche and Michael Segal, "A cluster-based beaconing approach in VANETs: Near optimal topology via proximity information", ACM Mobile Networks and Applications, 18(6), pp. 766-787, 2013.

[19] Majid Khabbazian, Stephane Durocher, and Alireza Haghnegahdar, "Bounding interference in wireless ad hoc networks with nodes in random position," LNCS'2012, vol. 7355, 2012.

[20] Haisheng Tan, Tiancheng Lou, Francis C. M. Lau, Yuexuan Wang, and Shiteng Chen, "Minimizing interference for the highway model in wireless ad-hoc and sensor networks," LNCS'2011, vol. 6543, pp. 520-532, 2011.

[21] Evangelos Kranakis, Danny Krizanc, Pat Morin, Lata Narayanan, and Ladislav Stachox, "A Tight Bound on the Maximum Interference of Random Sensors in the Highway Model," July 2010.

[22] Lusheng Miao, Karim Djouani, Barend J. van Wyk, and Yskandar Hamam, "A Survey of IEEE 802.11p MAC Protocol," Journal of Selected Areas in Telecommunications (JSAT), vol. 2011, no. 9, pp. 1-7, Sep. 2011.