Fast Randomized Algorithm for Hierarchical Clustering in Vehicular Ad-Hoc Networks

Efi Dror
Communication Systems Engineering
Ben-Gurion University of the Negev
efraima@bgu.ac.il

Chen Avin
Communication Systems Engineering
Ben-Gurion University of the Negev
avin@cse.bgu.ac.il

Zvi Lotker
Communication Systems Engineering
Ben-Gurion University of the Negev
zvilo@cse.bgu.ac.il

***Abstract*—Vehicular Ad-Hoc Networks (VANETs) offer communication between vehicles and infrastructure. Warning messages, among others, can be used to alert drivers, and thus improve road safety. To adapt to the unique nature of VANETs, which demands the delivery of time sensitive messages to nearby vehicles, fast topology control and scheduling algorithms are required. A clustering approach, which was initially offered for Mobile Ad-Hoc Networks (MANETs), can be adapted to VANETs to solve this problem. In this paper we present Hierarchical Clustering Algorithm (HCA), a fast randomized clustering and scheduling algorithm. HCA creates hierarchical clusters with a diameter of at most four hops. Additionally, the algorithm handles channel access and schedules transmissions within the cluster to ensure reliable communication. Unlike other clustering algorithms for VANETs, HCA does not rely on localization systems which contributes to its robustness. The running time of the algorithm was analyzed analytically and HCA was evaluated by simulation. We compared our algorithm with 2-ConID, a clustering algorithm for MANETs, under several mobility scenarios. The simulation results confirm that the algorithm behaves well under realistic vehicle mobility patterns and outperforms 2-ConID in terms of cluster stability.**

I. Introduction

Vehicular Ad-Hoc Networks (VANETs) serve as the basis for intelligent transportation systems (ITS) by utilizing Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communication. Dedicated hardware is planned to be incorporated into vehicles and some roadside units, while a wireless medium will be utilized to enable V2V and V2I communication.

Such systems can serve several purposes [1]: First, safety and warning messages could be used to alert drivers as to dangerous and unpredicted situations, and thus suppress the number of car accidents or reduce their severity. In addition, such systems can improve road utilization by managing traffic flows. Traffic updates can be delivered in real time to allow time saving and lower fuel consumption. Finally, commercial and entertainment services can be distributed via these systems. For example, advertisement can be sent to users based on their location and Internet access can be provided for passenger comfort.

Therefore, research of VANETs has been receiving increasing interest in the last couple of years, both on the algorithmic aspects [2]–[6] as well as standardization efforts such as the IEEE 802.11p and IEEE 1609 standards (named *WAVE* - Wireless Access in Vehicular Environments). 802.11p handles the MAC and PHY layers for each individual channel, while IEEE 1609 standards deal with upper layer protocols and multi-channel operation [7], [8].

VANETs have some unique characteristics and requirements [9]. On one hand, safety applications require extremely low message delay in order for them to be effective. On the other hand, unlike some MANETs (Mobile-Ad-Hoc Networks), power and computational abilities are sufficient since the devices are carried by vehicles. Another key characteristic of V2V communication is the high relative velocity between the vehicles engaged in such communication, which results in short term sessions. Consequently, solutions based on the current packet radio communication protocols cannot guarantee the required quality of service (QoS). In addition, safety messages are only required to be transmitted in a small radius in order for them to be effective. Combining these two key requirements leads to the basic approach - clustering vehicles into groups [2]–[4] to ensure high QoS and fast propagation of messages in a limited area.

The clustering approach was initially offered for MANETs [10]–[15]. These works include k-hop clusters where the number of hops between any of the nodes in the cluster is at most k [13]–[15]. More recent works adopted the clustering approach to VANETs [2]–[6]. However, these are all based on localization systems such as GPS to improve their performance, which in some situations leads to an undesirable dependency on these systems. We elaborate more on this in the related work section.

Following the unique nature of VANETs expressed in short term sessions, a fast clustering algorithm is required. In this paper we offer a fast randomized clustering and scheduling algorithm, HCA, to allow a quick network setup. Motivated by an industry-based requirement [16] our goals were to create 4-hop clusters (where each node in the cluster is at most 2 hops from a *cluster head*) as fast as possible without the use of GPS. Our basic motivation was that 4-hops are sufficiently "local" for safety and other messages types, and that the initial setup time is more critical than the quality of the clusters. The initial setup, in turn, can be followed by a maintenance phase that improves the clusters. Therefore, the suggested algorithm

This work was supported by the office of the Chief Scientist of the Ministry of Industry, Trade & Labor, Israel. Grant No. 41902.

978-1-4577-0899-2/11/$26.00 ©2011 IEEE

creates hierarchical clusters in which the maximal distance between a ClusterHead (CH) vehicle and any other vehicle in the cluster is two hops. Additionally, unlike other k clustering algorithms such as [13]–[15], the algorithm does not assume any lower layer connectivity, and the algorithm handles the channel access method and transmission scheduling within the cluster to avoid collisions. In order to evaluate the proposed algorithm prior to deployment and to examine its compatibility to vehicular networks, a dedicated simulator for VANETs was developed. The suggested algorithm was simulated in several scenarios and was compared with K-ConID algorithm [15] in which the number of the formed clusters and stability were measured. Simulations show that the algorithm performs better under realistic mobility patterns rather than under a random mobility pattern, which indicates that the algorithm suits VANETs. In addition, even though more clusters are formed by the HCA algorithm than by K-ConID, it forms more stable clusters. Furthermore, the algorithm's cluster formation process was analyzed analytically showing that the algorithm creates clusters within $O(m \log\log m)$ time slots with m denoting the maximal cluster size in the network (of size n).

The rest of this paper is organized as follows: Section II presents the proposed algorithm followed by the analytical runtime analysis in section III. Section IV presents the simulator which we had developed, the scenarios that were used to evaluate the algorithm, and the results that were obtained. We discuss related work on clustering algorithms for MANETs and VANETs in section V. The paper concludes with the conclusions and future work section.

II. Hierarchical Clustering For VANETs

The suggested algorithm is a distributed randomized two hop clustering algorithm (i.e., the maximal number of hops between any node to a ClusterHead is two). Our basic approach is to create a randomized dominating set in G^2 as detailed bellow.

A. Problem Definitions

First we will introduce some notions and definitions which will be used throughout this paper:

- G^2 of a graph $G = (V, E)$ is obtained by adding an edge $(u, v) \in E$ if node u is at most 2 hops away from v in G
- Dominating Set - A dominating set (DS) of a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that each node in $V \setminus V'$ is adjacent to some node in V'.
- ClusterHead - A ClusterHead is a node that belongs to the Dominating Set. It manages and handles the channel access method.
- Cluster - Each node has a single ClusterHead. A cluster is a subset of nodes with the same ClusterHead.
- ch_v and $slot_v$ are the cluster head of v and the time slot in which v transmit, respectively, for $v \in V$.

Next, we formulate the problem of creating clusters in which each node is at most 2 hops from a ClusterHead and scheduling them (i.e., nodes that belong to the same cluster have different transmission time slots) as G^2 dominating set problem named G^2DS.

Definition: The G^2DS Problem:

- Instance: Undirected graph $G = (V, E)$
- Solution:
  1) A Dominating Set of G^2, i.e., a subset $V' \subseteq V$ such that each vertex is either in V' or has a maximal two hops path to at least one of the vertices in V'. The nodes within the dominating set will be referred as ClusterHeads.
  2) Each node $u \in V$ has a single ClusterHead - $ch_u \in V'$
  3) Each node is allocated a unique transmission slot in its cluster i.e., $\forall v, u \in V$ such that if $ch_v = ch_u$ than $slot_v \neq slot_u$

The solution is measured in two aspects: the size of the Dominating Set and the time required to form clusters and assign a unique time slot for each of the nodes. There is obviously a tradeoff between the two, for example a solution can be a set that is composed of all the nodes in the network, which can be obtained immediately. However it is not an acceptable result and we would like to minimize the size of the dominating set. We take a randomized approach similar to [17], which allows us to benefit both from small dominating set and fast convergence.

Next we present our solution for the G^2DS problem:

B. Hierarchical Clustering Algorithm (HCA)

Next we present a distributed randomized solution to the G^2DS problem: Hierarchical Clustering Algorithm (HCA). The algorithm was influenced by work made by [17].

The HCA forms TDMA like synchronized clusters. In order to reduce the number of collisions by simultaneous transmissions in the same cluster, transmissions are only allowed on assigned slots by the ClusterHead. The algorithm is comprised of 4 phases. The first 3 phases refer to a static scenario in which the clusters are formed, while the fourth phase handles topology changes caused by mobility. The nodes are not required to travel according to a particular mobility pattern, and are allowed to move freely. Therefore, the algorithm can be used in VANETs or in any other network that consists of mobile entities (such as networks used for military or search and rescue purposes). It is worth mentioning that the suggested algorithm differs from the algorithms that will be presented in section V in two key aspects. First it handles the channel access and does not assume any lower layer connectivity. Secondly, it does not require the knowledge of the nodes' location. This feature enhances the algorithm's robustness since it does not rely on localization systems (e.g., GPS), which sometimes is preferable. We overcome the lack of information regarding the nodes' location by inferring connectivity from sent messages.

To ease the formation of clusters, in which the maximal distance from a ClusterHead to any other node in its cluster is

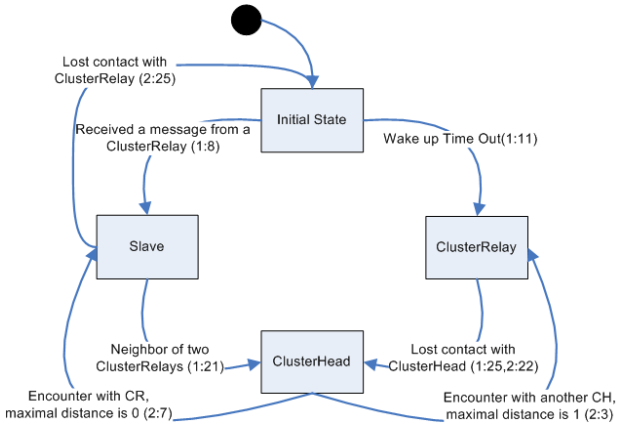

Fig. 1. HCA state machine of role transitions. Numbers denote the algorithm's index and relevant line in the associated to the transition.

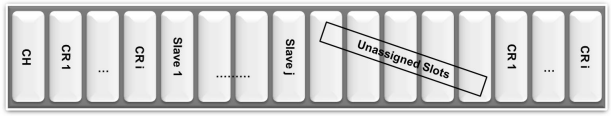

Fig. 2. SYNC message slots scheme. First a slot is assigned for the ClusterHead. Next, slots are allocated for each ClusterRelay, followed by slots which are randomly assigned for Slave nodes. Additional m empty slots are reserved for new members. The slots schemes completes with slots which are designated for ACKs sent by ClusterRelays.

two, 3 hierarchies are defined. Each hierarchy is characterized by a role carried out by the nodes. The *Slave* role is executed by regular nodes that are at most two hops away from a ClusterHead. Such nodes are at the lowest hierarchy of the cluster. The *ClusterRelay* (CR) role refers to nodes that relay messages from the ClusterHead to the Slaves, and help extend the reach of nodes within the cluster. ClusterRelay nodes are used to forward the control messages in both directions: from ClusterHeads to Slaves and vice versa. The *ClusterHead* role refers to nodes that manage and synchronize the shared channel access for all other nodes in the formed cluster (a CH is a member of the Dominating Set of G^2 as defined in subsection II-A). Such nodes are in the highest hierarchy of the algorithm. The state machine of possible roles and transitions among them is presented in Fig. 1. The pseudo code of HCA for a node v can be seen in Algorithm 1.

Some control messages are required for HCA's execution: SYNC messages, are messages that are created by ClusterHeads and used to assign slots to cluster member nodes. These messages are sent from ClusterHeads to Slave nodes and are forwarded by ClusterRelays. This type of scheduling is required in order to reduce the collisions caused by simultaneous transmissions in the same cluster. Fig. 2 shows a possible assignment of slots as carried by a SYNC message and includes the suggested scheduling scheme. ACK messages are messages required to acknowledge reception of SYNC messages. They are sent from Slave nodes to the ClusterHeads. Additionally, we define a round as the time period between two successive SYNC messages sent by the same ClusterHead.

Algorithm 1 Hierarchical Clustering

procedure INITIAL PHASE
  $role_v \leftarrow \emptyset$
  $cr_v \leftarrow \emptyset$
  $ch_v \leftarrow \emptyset$
  Select random slot from $2m$ slots for contention period
  go to *ClusterRelay Selection*
procedure CLUSTERRELAY SELECTION
On reception of SYNC *from* u *such that* $role_u = CR$
  $role_v \leftarrow Slave$
  $cr_v \leftarrow u$
  go to *ClusterHead Selection*
On Timeout
  $role_v \leftarrow CR$
  $cr_v \leftarrow v$
  Send SYNC messages periodically
  go to *ClusterHead Selection*
procedure CLUSTERHEAD SELECTION
  if $role_v = Slave$ and received SYNC messages from two CRs **then**
    Select random slot of $2m$ slots for contention period
    if Received SYNC message from another u such that $role_u = CH$ **then**
      Cancel contention period
    else
      $role_v \leftarrow CH$
      $ch_v \leftarrow v$
    go to *Synchronization And Cluster Formation*
On Timeout ▷ Didn't receive message from other CH
  if $role_v = CR$ **then**
    $role_v \leftarrow CH$
    $ch_v \leftarrow v$
  go to *Synchronization and Cluster Formation*
procedure SYNCHRONIZATION AND CLUSTER FORMATION
  Periodically send SYNC message with slots for all the known nodes with m unassigned slots
On SYNC *reception from* u
  if $role_v = CR$ **then**
    if $role_u = CH$ **then**
      Broadcast SYNC from u on the assigned slot
      $ch_v \leftarrow u$
On reception of ACK *from* u *such that* $role_u = Slave$
    Store the ACK and send ACK behalf of all nodes in the end of the round
  if $role_v = Slave$ **then**
    if SYNC contains assigned slot for v **then**
      Reply an ACK on the assigned slot
      $ch_v \leftarrow ch_u$
    else
      Select a random unassigned slot from the SYNC message and send ACK

The execution of the proposed algorithm can be divided into the following four phases, which are executed asynchronously:

(1) ClusterRelays Selection
(2) ClusterHead Selection
(3) Synchronization and Cluster Formation
(4) Cluster Maintenance

The first three phases refer to the initial phase of the algorithm in which the G^2DS is created, while the fourth phase handles topology changes and maintenance tasks. Each node stores the following parameters: ch_v, cr_v, $role_v$ and $N(v)$ which refers

to the node's ClusterHead, ClusterRelay, role and neighbors respectively. Additionally, each node has a parameter d_{max} which denotes its current maximal distance in hops within the cluster.

The *ClusterRelays Selection* phase is used to select ClusterRelays that help with the cluster formation. Initially, each node draws a random slot of $2m$ slots which would end its listening period. If a node has not received a message before its listening period elapses, it will announce itself as ClusterRelay.

In the *ClusterHead Selection* phase, a relatively central ClusterHead is selected. A Slave node which has heard a SYNC message from two ClusterRelays, will announce itself as nominated to become a ClusterHead. Such nodes set up a random period of time, drawn uniformly from $2m$ slots, in which they listen for incoming messages from other ClusterHead nominees. At the end of this period, nominees will send a message declaring themselves as ClusterHeads. The remaining nominees, if any, will drop their nomination and remain Slave nodes. In case there is only one ClusterRelay, and neither of the Slaves is in range of two ClusterRelays, the ClusterRelay will become a ClusterHead by itself, hence forming a one hop cluster.

The *Synchronization and Cluster Formation* phase is used for cluster formation and topology discovery by the ClusterHead and other nodes. The elected ClusterHead is not familiar with surrounding nodes. Therefore it sends SYNC messages to the ClusterRelays such that each ClusterRelay is assigned two unique slots - one for relaying the SYNC and the other for ACK transmission at the end of the round. ClusterRelays rebroadcast those messages to their neighbors. A Slave node that receives a SYNC message from its ClusterHead, sends an ACK message in the time slot which was assigned in the SYNC message. Additionally, SYNC messages contain m unassigned slots for new nodes to randomly select a slot and join the cluster. If the node has not yet been assigned a slot, meaning it is not a member of the cluster yet, it will select an unassigned slot randomly and send an ACK to inform of its joining. In each round, the ClusterRelay adds the identifiers of Slave nodes, as extracted from ACK messages, and stores them in its neighbors list. At the end of each round, an accumulated ACK is sent on behalf of all Slave nodes to the ClusterHead, in order to allow learning of the exact topology and assigning slots for the following round. Since nodes select a random slot for their first ACK, some collisions may occur. In case of such a collision, all colliding nodes will not receive a slot, and therefore they will contend in the next round. An analytical analysis (described in section III) guarantees a bounded limit for this process. The process continues until all nodes obtain their slot and reply with an ACK.

Finally, the *Cluster Maintenance* phase (Algorithm 2) is used to maintain up to date clusters and reduce the number of redundant clusters by merging relatively small clusters. Furthermore, it is required for forming new clusters due to topological changes. This phase is executed in parallel to phase 3 of the HCA algorithm.

In order to prevent the propagation of outdated information in the network, each node executes an aging process to eliminate irrelevant entries from the neighbors list. A Slave node that did not receive SYNC message from its ClusterRelay during an aging period will erase its ClusterRelay identifier and look for neighboring ClusterRelays within the same cluster. As a last resort, the Slave node will start its random listening period in the Initial State in order to join a different cluster. All timeout periods were chosen after an extensive simulation study.

In order to prevent an unlimited drift of ClusterHeads and the creation of redundant clusters, some rules for clusters merging are defined. First, a ClusterHead that was abandoned by all of its members, and encounters a ClusterRelay associated with another cluster, will be merged into the other cluster. Second, if a ClusterHead of a cluster with maximal distance within the cluster of one hop encounters another ClusterHead, the first ClusterHead will change its role to ClusterRelay and switch clusters together with all of its nodes. Additionally, a ClusterRelay that did not receive a SYNC message from its ClusterHead will try to join another cluster. If this did not succeed it will become a ClusterHead by itself. In addition, Slave node will send their ACK messages to the nearest ClusterRelay within the same cluster to allow smooth transition of Slaves to other ClusterRelays in the same cluster, according to topology changes caused by mobility.

Algorithm 2 Cluster Maintenance Phase

On Reception of SYNC *from* u
if $role_v = CH$ **then**
  if $role_u = CH$ and $d_{max} < 2$ **then**
    $role_v \leftarrow CR$
    $ch_v \leftarrow ch_u$
    $cr_v \leftarrow v$
  if $role_u = CR$, $ch_u \neq v$ and $d_{max} < 1$ **then**
    $role_v \leftarrow Slave$
    $ch_v \leftarrow ch_u$
    $cr_v \leftarrow u$
if $role_v = CR$ **then**
  Forward SYNC
if $role_v = Slave$ **then**
  if SYNC contains assigned slot for v **then**
    Reply an ACK on the assigned slot
    $ch_v \leftarrow ch_u$
  else
    Select a random slot of the unassigned slots and send ACK

On Aging Timeout
if $role_v = CR$ **then**
  if v is neighbor of u s.t $role_u = CH$ **then**
    $ch_v \leftarrow u$
  else
    $role_v \leftarrow CH$
    $ch_v \leftarrow v$
if $role_v = Slave$ **then**
  $cr_v \leftarrow \emptyset$
  if $\exists u \in N(v)$ s.t $role_u = CR$ and $ch_u = ch_v$ **then**
    $cr_v \leftarrow u$
  else
    Start over HCA

The main feature of this algorithm is its quick clusters setup.

This allows quick node synchronization by the ClusterHeads in order to maintain reliable communication. Next we present a run time analysis of HCA's first three phases, and a proof that it solves the G^2DS problem.

III. Analytical Results - Correctness and Run-time Analysis

This section will describe the key results of the analysis of HCA's first three phases (ClusterRelay selection; ClusterHead selection; and Synchronization and cluster formation). These three phases are assumed to take place while the nodes are static. Additionally, we'll assume that each node in the network is represented by a unique identifier, and that each node in G^2 has a maximal degree of m, i.e., the maximal size of a cluster in G is m. For the detailed proofs we refer the reader to the full version [18].

Theorem 1 (correctness): When the Hierarchical Clustering Algorithm terminates, the selected ClusterHeads form a Dominating Set on G^2, each node has a single ClusterHead and each node has a unique slot for transmission within the cluster.

Proof: First, one can see that each Slave node is within one hop from a ClusterRelay or is a ClusterRelay by itself. Each ClusterRelay is within one hop from a ClusterHead or is a ClusterHead by itself, therefore, each Slave node is within two hops from a ClusterHead, which is compliant with the G^2DS definition. In addition, each node has only one ClusterRelay, the node from which it initially received a `SYNC` message. Each ClusterRelay selects the first ClusterHead that it receives `SYNC` messages, hence each ClusterRelay has only one ClusterHead. Since ClusterRelays relay their selected ClusterHead identifier to their Slaves, it can be concluded that each node has only one ClusterHead. Additionally, each ClusterHead assigns slots for nodes. A node that has failed in selecting a unique slot, will try in the next round. Therefore we can conclude that each node will eventually be assigned a unique slot. ■

Next we address the running time of HCA:

Theorem 2 (time complexity): HCA's first three phases will terminate with high probability[1] in $O(m \log\log m)$ steps where $m \to \infty$ denotes the maximal size of a cluster.

The analysis utilizes the following lemmas which analyze the running time of each of the first three phases.

Lemma 1: The first phase of the HCA (selection of ClusterRelays) requires $O(\log_{\frac{4}{3}} m)$ slots with high probability. This phase terminates when all nodes in the network are ClusterRelays or within one hop from a ClusterRelay.

This is based on an analysis of the Balls and Bins model using Poisson approximation as presented in [19]. We defined the bins to be transmission slots and balls to be transmissions. Then we analyze the required number of slots for at least one successful transmission when m nodes contend for $2m$ slots, to achieve high probability. In the next phase a similar approach was used to find the number of slots required when an unknown number of nodes (can vary between 0 and $m-2$ nodes) contend for a slot among $2m$ available slots.

Lemma 2: The second phase of the HCA (selection of a ClusterHead) requires $O(m)$ slots with high probability. This phase terminates when all nodes in the cluster are within two hops from a ClusterRelay.

The third lemma is based on work presented by [20]. The authors present a simple distributed edge coloring algorithm and its running time analysis. The algorithm starts with each edge uncolored and terminates with all edges colored, and no two adjacent edges having the same color. At each stage, each edge selects independently with uniform distribution a color of its current colors palette. Followed by each stage, messages are exchanged by edges, and colors of adjacent edges are removed from the color palette. If conflicts arise, the edges will contend in the next round for another color. By referring to each node as an edge, and the slots pool as color pallets, we obtain the next Lemma:

Lemma 3: The third phase of the HCA (slots assignment and topology learning) requires $O(m \log\log m)$ slots with high probability. It terminates when all the nodes in the cluster are assigned a unique slot.

Taking the maximum time of the three lemmas we get the result of Theorem 2.

Next we present a simulation study of all four phases of HCA in different scenarios and comparison to K-ConID algorithm with $K = 2$.

IV. Evaluation

In this section we present a simulator which is dedicated for VANETs, and the scenarios that were used to evaluate the presented Hierarchical Clustering Algorithm in dynamic scenarios. The section concludes with the simulation results comparing HCA with K-ConID [15] algorithm with $K = 2$. K-ConID algorithm selects the nodes with the highest degree among their K neighbors to be ClusterHeads. The authors of K-ConID assumed that each node knows its K hops neighbors. This assumptions does not comply with realistic scenarios, therefore the algorithm was modified to allow learning of neighboring nodes by exchanging beacons.

A. Vehicular Ad - Hoc Network Dedicated Simulator

The Vehicular Ad - Hoc Network Dedicated Simulator was developed in order to evaluate VANET related algorithms and protocols. It combines both a traffic modeling simulator and a communication modeling simulator. The simulators are mutually influenced since road traffic can be changed according to traffic updates, and communication depends on the locations of the vehicles. OMNeT++ simulator [21] is used as the main simulation development environment with the MiXiM framework [22] used as the basis for developing new models, to suit the requirements for a VANETs oriented simulator. In addition, OMNeT++ was coupled with a road traffic simulator, SUMO [23], using a patch developed by [24] to allow bidirectional coupling of road traffic and communication simulators.

[1]Event $\mathcal{E}_m$ occurs with high probability if probability $\mathbb{P}[\mathcal{E}_m]$ is such that $\lim_{m\to\infty} \mathbb{P}[\mathcal{E}_m] = 1$

The simulator provides a tool for Vehicular Ad-Hoc Networks research; hence the main focus is on three key aspects. First, providing an accurate channel modeling according to the studied scenario and the topographic environment. Thus, several channel propagation characteristics were combined into one scenario according to the topographic environment yielding different channel models for each topographic area (urban, suburban, open space). Secondly, we focused on developing various mobility patterns based on [24]. Moreover, it includes the ability to characterize different mobility features (such as different maximal speeds) for each topographic area. Finally, the simulator must include a platform for evaluating algorithms and protocols prior to deployment on their dedicated hardware. We require that the evaluated algorithm's code will not have to be modified to allow execution on a dedicated hardware for VANETs. Therefore, the simulator must comply with the services provided by the hardware.

B. Scenarios and Results

HCA was evaluated using the VANETs simulator described in subsection IV-A. A suburb of Tel-Aviv metropolitan area, sized $1300 \times 3200m^2$, was extracted from Open-Street-Map project [25] and adapted into the SUMO traffic simulator (see Fig. 3 for screen-shot). Different mobility patterns were considered for the purpose of comparing the impact of the mobility pattern on the number of clusters and their stability, as formed by HCA and K-ConID with K set to 2. Three different mobility patterns were compared: i) Static scenario, in which nodes were forbidden to move; ii) Random Way Point (RWP) [26] in which vehicles select a random location and speed and change their location accordingly; And iii) TRACI - a traffic simulator generated traces model (as described in subsection IV-A). This model simulates a more realistic scenario in which vehicles are obligated to move along roads according to the extracted map and road regulations. In this scenario four different types of vehicles (differing in their maximal speed, acceleration and de-acceleration) selected one of six different source points in which they entered with Poisson rate. Each vehicle selected a random destination from a set of six possible destinations and drove towards that point according to the possible routes and traffic lights as simulated by the traffic simulator (SUMO). All three scenarios were based on a network consisting of 50, 75 and 100 vehicles, executing HCA and 2-ConID for 300 seconds of simulation time, in order to allow significant number of cluster creations and tear downs in highly dynamic scenarios. The maximal speed varied between 15m/s and 30 m/s. Since 2-ConID does not define channel access method like HCA does, we used MiXiM's 802.11 implementation as a channel access scheme. It is worth mentioning that 2-ConID does not handle scheduling, and therefore only clustering aspects (the number of clusters and cluster stability) were compared. Table I summaries key parameters used in the simulation study. Next we present the results that were obtained for the two studied measures.

1) Number of Clusters: One of the goals of the HCA was to minimize the number of formed clusters. Fig. 4 presents

TABLE I
SIMULATION PARAMETERS

Number of Vehicles	50, 75, 100	Transmission Rate	2.5Mbps
Playground Size	$1300 \times 3200m^2$	Slot Length	0.005sec
Path Loss Coefficient	3.5	Maximal Speed	15-30m/s
Carrier Frequency	5.85GHZ	Repetitions	20
Simulation Time	300sec	Transmission Range	200m

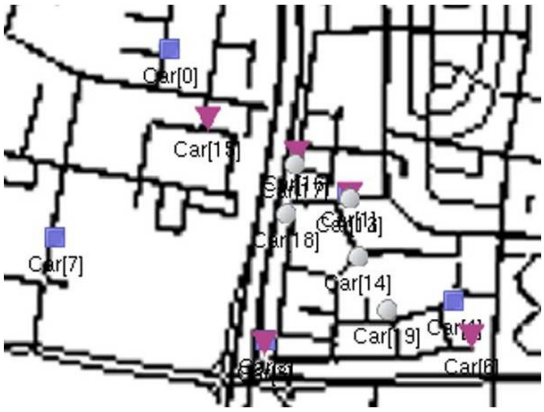

Fig. 3. A screen-shot of the simulation with the region which is used (a suburb of Tel-Aviv). ClusterHeads, ClusterRelays, and Slaves are denoted by rectangles, triangles, and circles respectively.

the average number of clusters (per time unit) formed in each scenario for different vehicular densities. We measure the number of ClusterHeads at each time step and calculate the mean number of clusters. First, it can be observed that for both algorithms, the mobility pattern indeed influences the number of clusters. It is clear that in the RWP scenario more clusters are created than in the TRACI scenario, and that the TRACI scenario outperforms the static scenario. This can be explained by the fact that the TRACI scenario imitates real life traffic, which constrains vehicles to move along roads. Due to the fact that vehicles usually move in groups, and perhaps not all of the simulated area contains roads, fewer clusters are required. Second, 2-ConID forms fewer clusters especially in the RWP scenario. This can be explained by the fact that the 2-ConID forces clusters to merge when every two ClusterHeads approach each other, while HCA tries to minimize cluster switching as can be seen in the *Cluster Stability* subsection.

2) Cluster Stability: Cluster stability was measured in terms of the average number of cluster switches throughout the simulation and the percentage of time in which nodes were members of a cluster, denoted as association time.

HCA outperforms 2-ConID in terms of cluster switches as can be seen in Fig. 5, which describes the average number of cluster switches per node in both algorithms. From it we conclude that a realistic mobility model (as simulated in the TRACI scenario) indeed minimizes the number of cluster switching and extends the membership period, therefore creating more stable clusters. This result is repeated in all three different vehicular densities. Static nodes executing HCA suffer from a noticeable amount of cluster switching which can be explained by nodes that are located in border

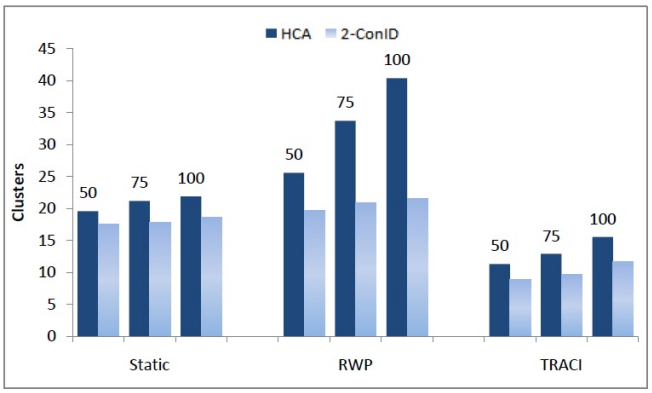

Fig. 4. The average number of clusters formed in HCA and 2-ConID in different scenarios with maximal speed set to 20m/s. Similar behavior was exhibited in different speeds. The numbers denote the number of vehicles.

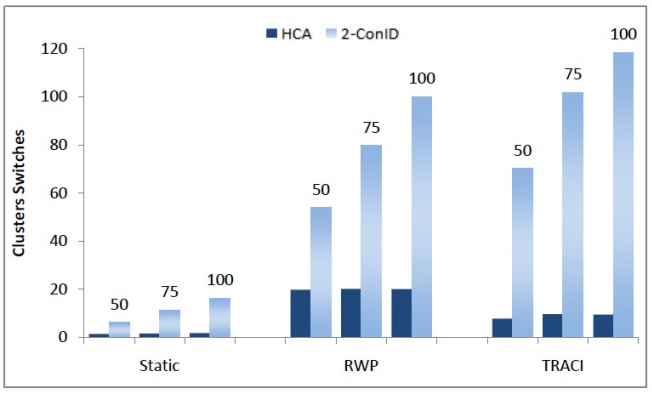

Fig. 5. The average number of cluster switches per node during the simulation time (300 sec), with vehicles' maximal speed set to 20m/s. Similar behavior was exhibited in different speeds. Nodes in the Static scenario suffer from cluster switching due to inter-cluster interferences in border nodes.

areas between two or more clusters. These nodes suffer from collisions caused by adjacent clusters. Comparing these results to the 2-ConID algorithm's results shows HCA's supremacy in terms of cluster stability. This can be explained by the fact that as opposed to the 2-ConID algorithm, HCA does not force clusters merging when it can lead to disconnection of members. It is worth mentioning that the 2-ConID algorithm suffers from cluster switching even in the static scenario. This is explained by the fact that nodes are not familiar with their neighborhood at the beginning of the algorithm's execution. Along with topology study, changes are made to comply with the 2-ConID requirements.

An additional dimension of stability was measured in terms of the percentage of time in which nodes were connected to a ClusterHead. Since 2-ConID forces nodes to become ClusterHeads when they are disconnected, this measure cannot be compared to HCA. HCA tries to minimize the number of 1 node clusters, and thus disconnected nodes might extend their contention period before declaring themselves as ClusterHeads. This results in disconnection periods. The study of the disconnection periods of HCA yields that nodes are connected to a ClusterHead in about $95\% - 97\%$ percent of the time in the RWP scenario and 98% percent of the time in the TRACI scenario. This emphasizes the fact that HCA forms stable clusters.

V. Related Work

Mobility in VANETs has to be modeled carefully for accurate simulation results. In [24], the authors present different approaches for simulating the nodes' mobility in a VANET simulation. The authors show that realistic mobility patterns are favorable for simulating VANETs.

Clustering is a well known method used in Ad Hoc networks. Some clustering algorithms [11], [12] use different metrics (e.g., id, mobility) for the cluster formation process in order to form stable clusters. All nodes send "Hello" messages with their metric and each of them selects the node with the lowest/highest value among its neighbors as their ClusterHead. Several works deal with extending the span of the clusters created. In [13] the authors present a heuristic for forming clusters with d as the maximal number of hops between any cluster member to its ClusterHead.

MobDHop is presented in [14]. It is a mobility based clustering algorithm with a varying diameter. The diameter varies according to the group mobility pattern detected by the algorithm as extracted from RSSI measurements. Stability metric is calculated according to the variation in the received power of exchanged messages. Following the creation of one hop clusters, nodes between two clusters initiate a merger between clusters with similar mobility patterns.

The authors of [15] modify the Lowest Id Algorithm [11] and different degree based algorithms to create k-hop clusters named K-ConID, i.e., the maximal number of hops between each cluster member to its ClusterHead is k. Nodes flood their identifiers, which consist of number of k-degree neighbors and id, to k hops away and select the larger identifier as their ClusterHead, with ties broken according to the lowest id.

During the last couple of years, several researches were held on clustering algorithms in VANETs scenarios. Such works use exact location knowledge by using devices such as GPS receivers. In [4] an algorithm that forms clusters with low relative velocity between cluster members is presented. The algorithm utilizes a GPS receiver and knowledge of future location to form stable clusters. The algorithm requires message exchange between each vehicle and its one hop neighbors, to allow distributed ClusterHead selection. Some of the works that deal with Vehicular Ad-Hoc Networks use the clustering algorithm to improve network stability by introducing new MAC layer algorithms and protocols. In [2] a clustering scheme that enables channel access for VANETS is presented. The authors present a substitution to the 802.11 MAC layer in VANETs. The algorithm calculates a weighted factor for the node's compatibility to act as a ClusterHead. The weighted factor requires the knowledge of the surrounding nodes' speeds and locations. Hence, each control message exchanged between nodes includes these parameters. Messages are sent using a TDMA scheme, with the ClusterHead scheduling all

nodes in the cluster. In addition, 10% of the slots are reserved for new nodes to join the cluster. Collisions at border nodes are resolved by ClusterHeads exchanging their local schemes and solving conflicts for neighboring nodes. However, this solution leads to some scaling issues.

In [3] a clustering scheme that forms clusters and enables intra-cluster communication using TDMA and inter-cluster communication using 802.11 MAC is presented. The suggested scheme requires the use of two transceivers, with one used for delay sensitive communication within the cluster, while the other is used for inter-cluster data transfer.

VI. Conclusions and Future Work

In this work we presented an algorithm for Hierarchical Clustering for Vehicular Ad-Hoc Networks. Such an algorithm is required for introducing QoS to VANETs in order to allow the delivery of time sensitive messages such as warning and safety messages. We showed that the suggested algorithm, HCA, meets the requirements of the G^2DS problem as defined in subsection II-A. Additionally, we demonstrated that HCA requires $O(m \log\log m)$ time slots with m denoting the maximal size of the cluster in order to form the initial clusters. The algorithm differs from other clustering algorithms for VANETs due to the fact that it is capable of creating clusters with a larger span than one hop from the ClusterHead. In addition, it does not require the knowledge of the vehicles' locations which contributes to its robustness.

A simulation study was used to compare the algorithm with a well known algorithm, K-ConID, in terms of the number of clusters and the clusters' stability. A simulation study showed that even though HCA forms few redundant clusters, the formed clusters are much more stable and robust to topology changes caused by vehicular movement. In addition, it can be seen that the mobility pattern influences the algorithm's behavior and has a great impact on the results obtained.

Nevertheless, HCA suffers from some difficulties in terms of inter cluster interferences which cause redundant cluster changes and message lose due to message collisions. Future work will try to resolve these inter-cluster interferences, perhaps by assigning different frequencies or codes, to allow more reliable communication. In addition, future work will utilize HCA to enable efficient inter cluster communication.

Acknowledgements

The authors would like to thank the anonymous reviewers and Arik Sapojnik, Naaman Sittsamer, Elad Gross, Eran Nahum and Oren Tzfati for their help and support in this project.

References

[1] K. Dar, M. Bakhouya, J. Gaber, M. Wack, and P. Lorenz, "Wireless communication technologies for ITS applications [Topics in Automotive Networking]," *Communications Magazine, IEEE*, vol. 48, no. 5, pp. 156–162, 2010.

[2] Y. Gunter, B. Wiegel, and H. Grossmann, "Cluster-based medium access scheme for vanets," in *Intelligent Transportation Systems Conference. ITSC 2007*. IEEE, 2007, pp. 343–348.

[3] H. Su and X. Zhang, "Clustering-based multichannel MAC protocols for QoS provisionings over vehicular ad hoc networks," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 6, pp. 3309–3323, 2007.

[4] C. Shea, B. Hassanabadi, and S. Valaee, "Mobility-based clustering in VANETs using affinity propagation," in *GLOBECOM*. IEEE, 2010, pp. 1–6.

[5] E. Souza, I. Nikolaidis, and P. Gburzynski, "A New Aggregate Local Mobility (ALM) Clustering Algorithm for VANETs," in *Communications (ICC), IEEE International Conference on*, 2010, pp. 1–5.

[6] P. Fan, "Improving broadcasting performance by clustering with stability for inter-vehicle communication," in *Vehicular Technology Conference, IEEE*, 2007, pp. 2491–2495.

[7] D. Jiang and L. Delgrossi, "Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments," in *Vehicular Technology Conference, 2008.* IEEE, 2008, pp. 2036–2040.

[8] R. Uzcategui and G. Acosta-Marum, "WAVE: a tutorial," *Communications Magazine, IEEE*, vol. 47, no. 5, pp. 126–133, 2009.

[9] J. Blum, A. Eskandarian, and L. Hoffman, "Challenges of intervehicle ad hoc networks," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 347–351, 2004.

[10] Z. Cai, M. Lu, and X. Wang, "Channel access-based self-organized clustering in ad hoc networks," *IEEE Transactions on Mobile Computing*, pp. 102–113, 2003.

[11] C. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 7, pp. 1265–1275, 2002.

[12] P. Basu, N. Khan, and T. Little, "A mobility based metric for clustering in mobile ad hoc networks," in *Distributed Computing Systems Workshop, 2001 International Conference on*. IEEE Computer Society, 2001, pp. 413–418.

[13] A. Amis, R. Prakash, T. Vuong, and D. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *INFOCOM*, vol. 1. IEEE, 2000, pp. 32–41.

[14] I. Er and W. Seah, "Mobility-based d-hop clustering algorithm for mobile ad hoc networks," in *Wireless Communications and Networking Conference.*, vol. 4. IEEE, 2004, pp. 2359–2364.

[15] G. Chen, F. Nocetti, J. Gonzalez, and I. Stojmenovic, "Connectivity-based k-hop clustering in wireless networks," in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. IEEE Computer Society, 2002, pp. 2450–2459.

[16] The Office of the Chief Scientist of the Ministry of Industry, Trade & Labor, Israel. Grant No. 41902., "Research and feasibility proof for dynamic, time limited, spontaneous wireless vehicular networks simulator." 2010.

[17] N. Alon, L. Babai, and A. Itai, "A fast and simple randomized parallel algorithm for the maximal independent set problem," *Journal of Algorithms*, vol. 7, no. 4, pp. 567–583, 1986.

[18] E. Dror, C. Avin, and Z. Lotker, "Fast randomized algorithm for hierarchical clustering in vehicular ad-hoc networks," BGU, Tech. Rep., 2011. [Online]. Available: www.bgu.ac.il/∼avin/papers/hca.pdf

[19] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge Univ Pr, 2005.

[20] D. Grable and A. Panconesi, "Nearly optimal distributed edge colouring in O (log log n) rounds," in *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1997, pp. 278–285.

[21] A. Varga, "OMNeT++ Web Site," http://www.omnetpp.org.

[22] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Haneveld, T. Parker, O. Visser, H. Lichte, and S. Valentin, "Simulating wireless and mobile networks in OMNeT++ the MiXiM vision," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, 2008, pp. 1–8.

[23] D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, J. Ringel, C. Rssel, and P. Wagner, "Simulation of Urban MObility package," in *European Simulation and Modelling Conference*, 2003.

[24] C. Sommer and F. Dressler, "Progressing toward realistic mobility models in VANET simulations," *Communications Magazine, IEEE*, vol. 46, no. 11, pp. 132–137, 2008.

[25] M. Haklay and P. Weber, "OpenStreetMap: user-generated street maps," *IEEE Pervasive Computing*, pp. 12–18, 2008.

[26] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile computing*, pp. 153–181, 1996.