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 timely sensitive messages to near by vehicles, fast topology control and scheduling algorithms are required. A clustering approach, which was initially offered for Mobile Ad-Hoc Networks (MANETs), can be adopted 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 at most four hops diameter. Additionally, the algorithm handles channel access and schedule 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 to 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.**

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 are 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 on VANETs has been receiving increasing interest in the last couple of years, both on the algorithmic aspects [2]–[6] as well as standardization efforts like 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-hops clusters where the number of hops between any of the nodes in the cluster is at most k [13]–[15]. More recent work adopted the clustering approach to VANETs. Examples include [2]–[6], however, they are all based on a localization system such as GPS to improve their performance. However, this leads to undesirable dependence upon these systems in some situations. 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-hops 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 quality of the clusters. The initial setup, in turn, can be followed by a maintenance phase that improves the clusters. Therefore, the suggested algorithm 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 to 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 suggested algorithm and the running time is then analyzed analytically 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 and the paper concludes with the conclusions and future work section.

II. Hierarchical Clustering For VANETs

The suggested algorithm is a distributed randomized two hops clustering algorithm (i.e., the maximal number of hops between any node to a ClusterHead is two). Our basic approach is to create a randomize 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 randomize 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, transmissions are only allowed on assigned slots by the ClusterHead. The algorithm is comprised of 4 phases. The first 3 phases referring to a static scenario in which the clusters are formed, while the fourth phase handles topology changes caused by mobility changes. 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 differ from the algorithms that will be presented in section V in two key aspects. First it does not require the knowledge of nodes' location. Such feature enhances the algorithm's robustness since it does not rely on localization systems (e.g., GPS), which sometimes is preferable. Secondly, it handles the channel access and does not assume any lower layer connectivity.
To ease the formation of clusters, in which the maximal distance from a ClusterHead to any other node in the clusters is two, 3 hierarchies are defined. Each hierarchy is characterized by 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

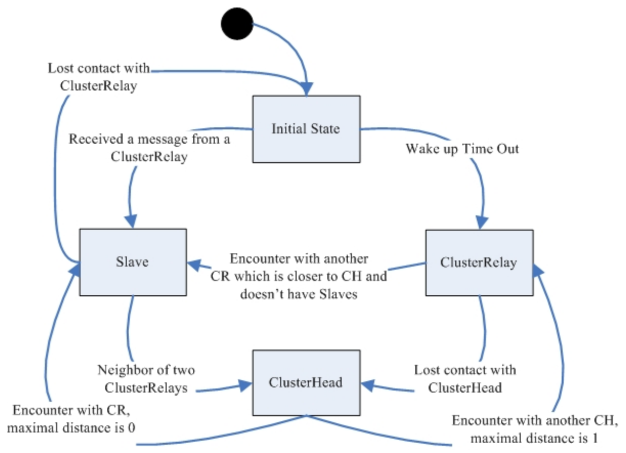
Lost contact with ClusterRelay
Initial State
Received a message from a ClusterRelay
Wake up Time Out
Slave
Encounter with another CR which is closer to CH and doesn't have Slaves
ClusterRelay
Neighbor of two ClusterRelays
ClusterHead
Lost contact with ClusterHead
Encounter with CR, maximal distance is 0
Encounter with another CH, maximal distance is 1

Fig. 1. HCA state machine of role role transitions

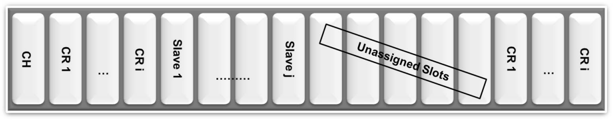

Fig. 2. SYNC message slots scheme. First a slot is assigned for the ClusterHead followed by slots for each ClusterRelay. Slave nodes are assigned next, with m empty slots. The slots schemes completes with slots for ACKs from the ClusterRelays

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 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 the HCA's execution: SYNC messages, are messages that are created by ClusterHeads, which are used to assign slots to cluster member nodes. These messages are sent from ClusterHeads to Slave nodes and are forwarded by ClusterRelays. This is required in order to reduce the collisions that are created due to simultaneous transmissions. Fig. 2 shows a possible assignment of slots as carried by a SYNC message. 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.

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

Algorithm 1 Hierarchical Clustering

```
1: procedure INITIAL PHASE
2:     $role_v \leftarrow \emptyset$
3:     $cr_v \leftarrow \emptyset$
4:     $ch_v \leftarrow \emptyset$
5:     Select random slot from $2m$ slots for contention period
6:     go to *ClusterRelay Selection*
7: procedure CLUSTERRELAY SELECTION
   *On reception of* SYNC *from $u$ such that $role_u = CR$*
8:     $role_v \leftarrow Slave$
9:     $cr_v \leftarrow u$
10:    go to *ClusterHead Selection*
   *On Timeout*
11:    $role_v \leftarrow CR$
12:    $cr_v \leftarrow v$
13:    Send SYNC messages periodically
14:    go to *ClusterHead Selection*
15: procedure CLUSTERHEAD SELECTION
16:    if $role_v = Slave$ and received SYNC messages from two CRs then
17:        Select random slot of $2m$ slots for contention period
18:        if Received SYNC message from another $u$ such that $role_u = CH$ then
19:            Cancel contention period
20:        else
21:            $role_v \leftarrow CH$
22:            $ch_v \leftarrow v$
23:            Periodically send SYNC message with slots for all the known nodes with $2m$ slots unassigned
24:        go to *Synchronization And Cluster Formation*
   *On Timeout*
25:    if $role_v = CR$ then
26:        $role_v \leftarrow CH$
27:        $ch_v \leftarrow v$
28:    go to *Synchronization And Cluster Formation*
29: procedure SYNCHRONIZATION AND CLUSTER FORMATION
   *On* SYNC *reception from $u$*
30:    if $role_v = CR$ then
31:        if $role_u = CH$ then
32:            Broadcast SYNC from $u$ on the assigned slot
33:            $ch_v \leftarrow u$
   *On reception of* ACK *from $u$ such that $role_u = Slave$*
34:        Store the ACK and send ACK behalf of all nodes in the end of the round
35:    if $role_v = Slave$ then
36:        if SYNC contains assigned slot for $v$ then
37:            Reply an ACK on the assigned slot
38:            $ch_v \leftarrow ch_u$
39:        else
40:            Select a random unassigned slot from the SYNC message and send ACK
```

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 and $role_v$ which refers to the node's ClusterHead, ClusterRelay and role 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 had 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 had heard `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, before they announce themselves as ClusterHeads and send a message declaring themselves as ClusterHead. If so, other ClusterHead nominees, if exist, will drop their nomination and remain Slave nodes.

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 with each ClusterRelay assigned two unique slots, to be used for transmissions - 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 `SYNC` message from its ClusterHead, sends an `ACK` message in a slot as 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 is not yet 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 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. The process continuous until all nodes obtains 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. 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. 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.
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 its nodes. In case both ClusterRelays control clusters with maximal distance to the ClusterHead of one, some arbitrary rule (such as number of members, id, etc.) can be used to decide the identity of the ClusterHead that needs to abandon its role. 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 $u$
 1: if $role_v = CH$ then
 2:     if $role_u = CH$ and $d_{max} < 2$ then
 3:         $role_v \leftarrow CR$
 4:         $ch_v \leftarrow ch_u$
 5:         $cr_v \leftarrow v$
 6:     if $role_u = CR$, $ch_u \neq v$ and $d_{max} < 1$ then
 7:         $role_v \leftarrow Slave$
 8:         $ch_v \leftarrow ch_u$
 9:         $cr_v \leftarrow u$
10: if $role_v = CR$ then
11:     Forward SYNC
12: if $role_v = Slave$ then
13:     if SYNC contains assigned slot for $v$ then
14:         Reply an ACK on the assigned slot
15:         $ch_v \leftarrow ch_u$
16:     else
17:         Select a random slot of the unassigned slots and send
    ACK

    On Aging Timeout
18: if $role_v = CR$ then
19:     if $v$ is neighbor of $u$ s.t $role_u = CH$ then
20:         $ch_v \leftarrow u$
21:     else
22:         $role_v \leftarrow CH$
23:         $ch_v \leftarrow v$
24: if $role_v = Slave$ then
25:     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 an analysis of HCA's first three phases run time 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; 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 appendix.

Theorem 1 (correctness): When 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 a ClusterRelay by itself. Each ClusterRelay is within one hop from a ClusterHead or 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, which is the node that a `SYNC` message was received from it initially. Each ClusterRelay selects the first ClusterHead that it receive `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 one and only 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 utilities 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 [18]. 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 be 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 [19]. The authors present a simple distributed edge coloring algorithm and its running time analysis. The algorithm starts when each edge is uncolored while the algorithm terminates when all edges are colored, and there are no two adjacent edges colored with 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 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 less than $o(m \log\log m)$ 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 a dynamic scenarios. The section concludes with the simulation results comparing HCA to 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 it 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 [20] is used as the main simulation development environment with the MiXiM framework [21] 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 [22] using a patch developed by [23] to allow bidirectional coupling of road traffic and communication simulators.
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 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 [23]. 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

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

TABLE I
SIMULATION PARAMETERS

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

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 sub-urban of Tel-Aviv metropolitan area, sized $1300 \times 3200m^2$ was extracted from Open-Street-Map project [24] and adapted into 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 the stability of them 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) [25] in which vehicles select random location and speed and change their location according to it. And iii) TRACI - a traffic simulator generated traces model (as described in subsection IV-A). This model simulates a more realistic scenario in which vehicle are obligated to move along roads according to the extracted map and road regulations. In this scenario four different types of vehicles (each differ in its 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. Since 2-ConID does not define channel access method like HCA does, we used MiXiM's 802.11 implementation as a channel access scheme. 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 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 seen 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. The TRACI scenario outperforms the static scenario and it 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 simulated area contains roads, less clusters are required. Second, 2-ConID forms fewer clusters specially in the RWP scenario. This can be explained by the fact that the 2-ConID forces cluster merging when every two ClusterHeads come near each other, while HCA tries to minimize cluster switching as can be seen in the *Cluster Stability* subsection.

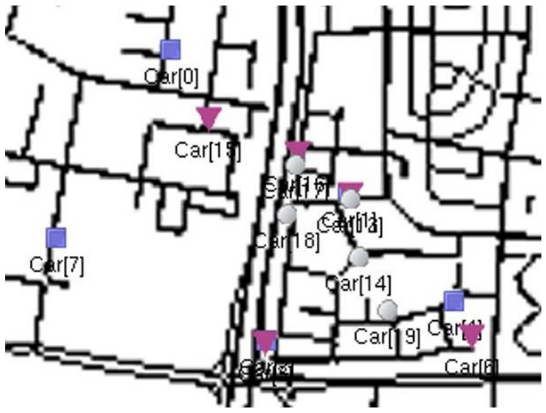

Fig. 3. A simulation screen-shot with the used region to evaluate dynamic scenario (a map of a region of Tel-Aviv). ClusterHeads are denoted by rectangles, ClusterRelay are denoted by triangles and slaves as circles

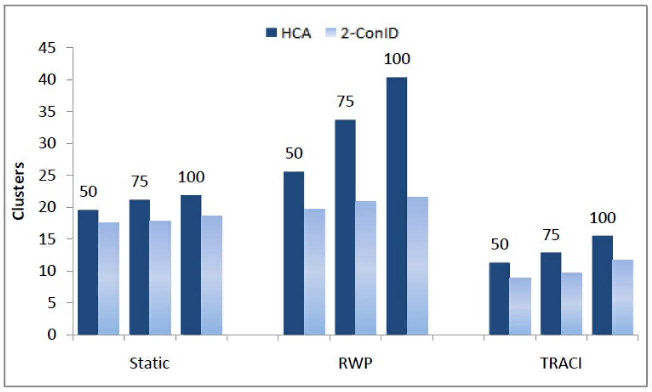

Fig. 4. Average number of clusters formed in HCA and 2-ConID in different scenarios. Numbers denote the number of vehicles.

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 extend membership period, therefore, creating more stable clusters. This result is repeated in all

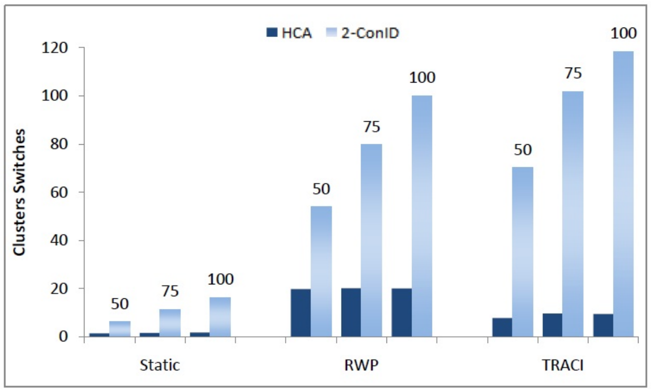

Fig. 5. The average number of cluster switches. The nodes in the Static scenario, in HCA algorithm, suffer from cluster switching due to inter-cluster interferences in border nodes.

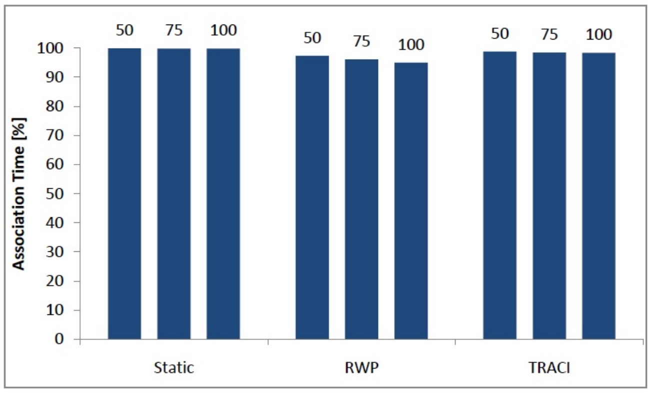

Fig. 6. The average percentage of time that each node was associated to a cluster in the HCA algorithm.

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 areas between two or more clusters. These nodes suffer from collisions caused by unsynchronized clusters. Comparing these results to the 2-ConID algorithm's results, show HCA's supremacy in terms of cluster stability. This can be explained by the fact that HCA does not force clusters merging when it can lead to disconnection of members, such as in the 2-ConID algorithm. 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 the with the 2-ConID requirements.

Fig. 6 presents the average percentage of time that each node executing HCA was associated to a cluster. Disconnection periods are caused by the topology changes and inter-cluster interferences. As a result, it can be seen that the RWP model causes the clusters to be less stable than other scenarios. It is worth mentioning that in the static scenario the nodes were not associated with a cluster 100 percent of time. This can be explained by inter cluster interferences which result in collisions and packets loss. This causes nodes to disconnect from their clusters, therefore decreasing the percentage of time nodes are connected. This scenario was not compared to the 2-ConID algorithm because nodes that were disconnected, in the 2-ConID algorithm will immediately join a neighboring cluster or form a new one, therefore such node can be seen as constantly cluster members.

V. Related Work

Mobility in VANETs has to be modeled carefully for accurate simulation results. In [23], the authors present different approaches for simulating the nodes' mobility in 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 a 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 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 surrounding nodes' speeds and locations. Hence, each control message exchanged between nodes include these parameters. Messages are sent in 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 to allow introducing QoS to VANETs in order to allow the delivery of timely 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 to well known algorithm K-ConID in terms of number of clusters and clusters' stability. 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 due to message collisions. Future work will try to resolve these inter-cluster interferences. In addition, future work will try to increase association time by maintaining information regarding neighboring clusters, enabling faster cluster switching.

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, 2007. ITSC 2007. IEEE.* IEEE, 2007, pp. 343–348.

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

[4] C. Shea, B. Hassanabadi, and S. Valaee, "Mobility-based clustering in VANETs using affinity propagation," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE.* 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), 2010 IEEE International Conference on.* IEEE, 2010, pp. 1–5.

[6] P. Fan, "Improving broadcasting performance by clustering with stability for inter-vehicle communication," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th.* 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. VTC Spring 2008. IEEE.* 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," *icdcsw*, p. 0413, 2001.

[13] A. Amis, R. Prakash, T. Vuong, and D. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2002, 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, 2004. WCNC. 2004 IEEE*, 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 *hicss*. Published by the IEEE Computer Society, 2002, p. 188c.

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

[19] 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.

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

[21] 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 (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–8.

[22] D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, J. Ringel, C. Rssel, and P. Wagner, "The" Simulation of Urban MObility" package: An open source traffic simulation," in *2003 European Simulation and Modelling Conference*, 2003.

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

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

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

Appendix

Theorem 3: The algorithm outputs a Dominating Set on 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'.

Proof: A ClusterHead will be defined as a member of the Dominating Set. The proof will show that each node is within two hops from a ClusterHead. First, one can see that each Slave node is within one hop from a ClusterRelay or a ClusterRelay by itself. Each ClusterRelay is within one hop from a ClusterHead or a ClusterHead by itself if it didn't receive a message from a ClusterHead, hence each Slave node is within two hops from a ClusterHead, which is compliant with the G^2DS definition. This concludes the proof of the first algorithm output. ■

Theorem 4: Each node $u \in V$ has a unique ClusterHead - $ch_u \in V'$

Proof: First, one can see that each node has only one ClusterRelay, which is the node that a message was received from it initially. If such message wasn't received the node would become a ClusterRelay. The first ClusterHead that its listening period will elapse, will announce it to its ClusterRelays. If two ClusterHeads will transmit simultaneously a collision would occur, which would result in other slave node declaring itself as ClusterHead. Each ClusterRelay selects the first ClusterHead, hence each ClusterRelay has only one ClusterHead. If a ClusterRelay didn't receive any message from a ClusterHead it will select itself as ClusterHead. The ClusterRelays relay their selected ClusterHead identifier to their Slaves so it can be concluded that each node has one and only ClusterHead. This concludes the proof of the algorithms second output. ■

Theorem 5: The algorithm will terminate when all nodes are assigned a unique slot for transmission. The termination will occur in $O(m \log \log m)$ with high probability (w.h.p) i.e, $\left(1-\frac{1}{m}\right)$, with m denoting the number of maximal nodes in a cluster.

In order to prove theorem 5 we'll analyze the running time of each of the algorithms phases while executed in a single cluster and their failure probability. The Balls and Bins model will be used to calculate the probability of failure in the first two phases. Failure probability is referred to the probability that neither of the transmitted nodes had succeeded in transmitting in a unique slot during the first t slots. The Balls and Bins model referrers to a model in which m balls are thrown with uniform distribution into n unique bins. During the algorithms analyses each slot will be referred as a bin and a transmission would be referred as a ball. In order to simplify calculations, we would like to assume that each bin is an independent variable, which would enable us to find an upper bound for the exact case probability. Using Poisson approximation for relatively rare events can simplify calculation and an upper bound can be derived for the failure probability. In order to assume so, the authors of [18, Corollary 5.9] found an upper bound to the probability of an event in the Balls and Bins $\mathcal{E}_{BB}$ yielding:

$$\mathbb{P}\left[\mathcal{E}_{BB}\right] \leq e\sqrt{m} \cdot \mathbb{P}\left[\mathcal{E}_{P}\right] \tag{1}$$

with $\mathcal{E}_P$ denoting the event in the Poisson approximation case.
Let x be a Poisson distributed random variable with parameter $\lambda = \frac{m}{n}$, which denotes the number of balls in each bin and the expected number of balls in all the bins is n. p_1 denotes the probability that a bin contains exactly one ball:

$$p_1 = e^{-\lambda}\lambda \tag{2}$$

Since $x_1, x_2 \ldots x_n$ are independent Poisson variables the probability that neither of the first t bins contain one ball is given by p_t:

$$p_t = (1 - e^{-\lambda}\lambda)^t \tag{3}$$

Claim 1: Let z be a random variable which denote the index of the first slot with exactly one ball, the distribution of z is given according to Geometric distribution:

$$\mathbb{P}\left[z = i\right] = \begin{cases} (1 - e^{-\lambda}\lambda)^{i-1}(e^{-\lambda}\lambda) & i \geq 1 \\ 0 & otherwise \end{cases} \tag{4}$$

Thus the expected number of required slots before a successful slot is given by

$$\mathbb{E}\left[z\right] = \frac{1}{p_1} = \frac{e^{\lambda}}{\lambda} \tag{5}$$

Lemma 4: The first phase of the Leader Election Algorithm (selection of ClusterRelays) requires $O(\log_{\frac{4}{3}} m)$ slots w.h.p. This phase terminates when all nodes in the network are ClusterRelays or within one hop from a ClusterRelay.

Proof: To prove the lemma we will bound the the probability of failure $\mathbb{P}\left[failure\right]$. Failure occurs when all the contending nodes collide during the first $t = c_1 \log_{\frac{4}{3}} m$ slots, i.e non of the first t slots has only one transmission, with t denoting the time required for this phase in slots and $c_1 \geq 1.5$ a constant. Setting n as $2m$ yields $\lambda = 1/2$, and t in equation 3 yields:

$$p_{c_1 \log_{\frac{4}{3}} m} = \left(1 - \frac{1}{2\sqrt{e}}\right)^{c_1 \log_{\frac{4}{3}} m} \leq \left(\frac{3}{4}\right)^{c_1 \log_{\frac{4}{3}} m} = \left(\frac{1}{m}\right)^{c_1} \tag{6}$$

Next $\mathbb{P}\left[failure\right]$ the probability that the first phase would fail will be found by finding an upper bound to the exact case (Balls and Bins). Using equation 1 yields:

$$\mathbb{P}\left[failure\right] \leq e\sqrt{m}\cdot p_{c_1 \log_{\frac{4}{3}} m} = e\sqrt{m}\Big(\frac{1}{m}\Big)^{c_1} = e\Big(\frac{1}{m}\Big)^{c_2} \tag{7}$$

This means that at least one node will successfully select a unique slot among the first $c_1 \log_{\frac{4}{3}} m$ slots and declare itself as ClusterRelay w.h.p. All its direct neighbors will receive its message and terminate the execution of this phase. This implies that the first phase will terminate in $c_1 \log_{\frac{4}{3}} m$ slots, while the expected number of required slots is given by equation 5:

$$\mathbb{E}\left[\mathrm{F}irstSlot\right] = \frac{e^{\lambda}}{\lambda} = 2\cdot\sqrt{e} \tag{8}$$

■

Lemma 5: The second phase of the Leader Election Algorithm (selection of a ClusterHead) requires $O(m)$ slots w.h.p. This phase terminates when all nodes in the cluster are within two hops from a ClusterRelay.

Proof: In order to prove the lemma we will bound the probability of failure $\mathbb{P}\left[failure\right]$. During the second phase Slave nodes that heard two ClusterRelays (will be referred as nominees) will propose themselves to act as ClusterHead. This is done by selecting a slot among the first $O(m)$ slots. Since there are only $O(m)$ slots to select from, failure will occur only if all the nominees will collide during their proposal, i.e, non of the other nodes in the cluster can receive their proposal. Since the number of nominees k is unknown (it varies from 1 to $m-2$ nodes) the analyses will be divided into two cases: **1)** $\log m \leq k \leq m-2$ and **2)** $1 \leq k < \log m$.
For values of k that are larger than $\log m$ a similar approach to the one used in lemma 4 will be used. The analyses will consider the worst case (highest probability of failure) i.e,. $k = \log m$ hence $\lambda = \frac{\log m}{m}$. Using equation 3 with $t = c_1 \cdot m$ with $c_1 \geq 1.5$ denoting a constant, (since a failure would occur if the first $O(m)$ slots will not contain a single nodes transmission) yields:

$$p_{c_1\cdot m} = \Big(1 - \frac{1}{e^{\lambda}\cdot\frac{1}{\lambda}}\Big)^{c_1\cdot m} = \Big(1 - \frac{1}{e^{\lambda}\cdot\frac{1}{\lambda}}\Big)^{c_1\cdot m\cdot\frac{1}{e^{\lambda}\cdot\frac{1}{\lambda}}\cdot e^{\lambda}\cdot\frac{1}{\lambda}} \leq \Big(\frac{1}{e}\Big)^{c_1\cdot m\cdot e^{-\lambda}\cdot\lambda} = \Big(\frac{1}{e}\Big)^{c_1\cdot m\cdot e^{-\frac{\log m}{m}}\cdot\frac{\log m}{m}} \tag{9}$$

Hence

$$p_{c_1\cdot m} \leq \Big(\frac{1}{m}\Big)^{c_1\cdot e^{-\frac{\log m}{m}}} \leq \Big(\frac{1}{m}\Big)^{c_1} \tag{10}$$

Finding an upper bound to the exact case $\mathbb{P}\left[failure\right]$ yields:

$$\mathbb{P}\left[failure\right] \leq e\sqrt{m}\cdot p_{c_1\cdot m} \leq \Big(\frac{1}{m}\Big)^{c_2} \tag{11}$$

For small values of k i.e, $1 < k \leq \log m$ a different technique would be used. We'll look from the nodes perspective in the worst case i.e, the number of contending nodes is $\log m$.
We'll define 3 mutually exclusive events that relate to the number of transmissions in each slot and analyse their occurrence probability.

$$\begin{cases} e_1 & \text{At least one slot with at least 3 transmissions} \\ e_2 & \text{All slots with transmissions contain exactly 2 transmissions} \\ e_3 & \text{At least one slot with 1 transmission given that the maximal load is at most 2} \end{cases}$$

The event of failure F (There is not a single slot with a single transmission) can be expressed by $F \cap e_3 = \emptyset$ therefore $F \subseteq e_1 \cup e_2$ and it's obvious that $\mathbb{P}\left[failure\right] = \Pr(F) \leq \Pr(e_1) + \Pr(e_2)$. Therefore $\Pr(e_1)$ and $\Pr(e_2)$ will be presented. $\Pr(e_1)$ is first introduced: The probability that 3 nodes select the same slot is given by $m\cdot\frac{1}{m^3}$. The number of distinct sets that contain 3 nodes that choose from $\log m$ slots is $\binom{\log m}{3}$. Therefore:

$$\Pr(e_1) \leq \Big(\frac{1}{m^2}\Big)\binom{\log m}{3} \leq O\Big(\frac{\log^3 m}{m^2}\Big) \tag{12}$$

Next $\Pr(e_2)$ will be found. First we'll define Pair Set; a Pair Set S is a set of unordered pairs of numbers. Given S, the probability that the set will be ordered in some order is given by:

$$P_s = 1\cdot\frac{1}{m}\cdot\Big(\frac{m-1}{m}\cdot\frac{1}{m}\Big)\cdot\Big(\frac{m-2}{m}\cdot\frac{1}{m}\Big)\ldots\Big(\frac{m-\log m}{m}\cdot\frac{1}{m}\Big) \tag{13}$$

This is obtained by the probability that each odd indexed node selects a slot that wasn't already selected and that the following node selects the same slot as the previous one. Next the number of distinct Pair Sets will be found. There are $\log m!$ permutations of $\log m$ nodes and since the order between the pairs themselves is not important this is divided by $\frac{\log m}{2}!$. The number of permutations among two nodes given $\frac{\log m}{2}$ pairs is $2^{\frac{\log m}{2}}$. Therefore the number of distinct Pair Sets is:

$$\frac{\log m!}{2^{\frac{\log m}{2}} \cdot \left(\frac{\log m}{2}\right)!} \tag{14}$$

From equations 13 (probability per a Pair Set) and 14 (number of Pair Sets) an upper bound can be derived:

$$\Pr(e_2) = \sum_{s \in S} P_s \leq \left(\frac{1}{m}\right)^{2\log m} \cdot (m-1)(m-2)\ldots(m-\log m) \cdot \frac{\log m!}{2^{\frac{\log m}{2}} \cdot \left(\frac{\log m}{2}\right)!}$$

$$\Pr(e_2) \leq O\left(\left(\frac{1}{m}\right)^{2\log m} \cdot m^{\log m} \cdot \frac{\log m \cdot (\log m - 1)\ldots(\frac{\log m}{2}+1)}{2^{\frac{\log m}{2}}}\right)$$

$$\Pr(e_2) \leq O\left(\left(\frac{1}{m}\right)^{\log m} \cdot \frac{\log m^{\frac{\log m}{2}}}{2^{\frac{\log m}{2}}}\right) \leq O\left(\left(\frac{1}{m}\right)^{\log m} \cdot \left(\frac{\log m}{2}\right)^{\frac{\log m}{2}}\right)$$

$$\Pr(e_2) \leq O\left(\left(\frac{1}{m}\right)^{\log m} \cdot (m)^{\frac{\log m}{4}}\right) \leq O\left(\frac{1}{m}\right) \tag{15}$$

From equations 12 and 15:

$$\mathbb{P}\left[failure\right] \leq \Pr(e_1) + \Pr(e_2) \leq O\left(\frac{1}{m}\right) \tag{16}$$

It can be seen that the failure probability, $\mathbb{P}\left[failure\right]$ in scenarios that the number of nodes contending k, is smaller than $\log m$, is relatively small. Hence it can be shown w.h.p that there will be at least one slot with a single transmission.

Remark: In case that all nodes are within the hearing range of the first woken node (ClusterRelay), non of the neighboring nodes will select a slot (hence $k = 0$). In that case the first node would become a ClusterHead in $O(m)$ time slots. ■

Lemma 6: The third phase of the Leader Election algorithm (slots assignment and topology learning) requires less than $o(m \log\log m)$ slots with high probability. It terminates when all the nodes in the Cluster were assigned a unique slot.

Proof: The third phase analyses is based on work made by Grable and Panconesi [19]. The authors present a simple distributed edge colouring algorithm and its running time analysis. The algorithm starts when each edge is uncolored while the algorithm terminates when all edges are colored and there are no two adjacent edges colored with 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. The authors show that the probability to success by selecting a unique color increases by a double exponential rate as the algorithm advances. Hence the algorithm requires $O((1+c^{-1})\log\log m)$ rounds of communication, while the initial palette size is $\Omega(m^{c/\log\log m})$, with m denoting the number of nodes in a graph and c denoting a constant. The authors show that the algorithm fails with $o(1)$.

The Leader Election algorithms third phase sets a unique transmission slot for each of the nodes in the cluster. This can be presented as finding an edge colouring setup in a clique with nodes denoting the edges in the graph and slots denoting the colors palette. Therefore c will be set to $\log\log m$ (initial palette of $\Omega(m)$ colors) yielding a running time of $O(\log\log m)$ rounds of communication. In addition each round requires in the Leader Election algorithm $2m$ slots, since nodes select from a pool of $2m$ slots, summing up for a total of $O(m \log\log m)$ time slots. ■

Proof: (Theorem 5) Since each of all three phases of the Leader Election Algorithm terminate within $O(m \log\log m)$ slots, it can be concluded that the algorithms run-time is $O(m \log\log m)$ with high probability as $1 - \frac{1}{m}$. ■