Mobility-based Clustering in VANETs using Affinity Propagation

Christine Shea, Behnam Hassanabadi and Shahrokh Valaee
Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON, M5S 2E4,
Canada
{cshea,behnam,valaee}@comm.utoronto.ca

***Abstract*—The recent research in cluster-based MAC and routing schemes for Vehicle Ad Hoc Networks (VANETs) motivates the necessity for a stable VANET clustering algorithm. Due to the highly mobile nature of VANETs, mobility must play an integral role in cluster formation. We present a novel, mobility-based clustering scheme for Vehicle Ad hoc Networks, which utilizes the Affinity Propagation algorithm in a distributed manner. The proposed algorithm considers typical vehicular mobility during cluster formation to garner clusters with high stability. Simulation results confirm the superior performance of the proposed algorithm, when compared to other accepted mobility-based clustering techniques, in terms of average cluster head duration, average cluster member duration, and average rate of cluster head change.**

I. Introduction

Research in vehicular communications, specifically Vehicular Ad Hoc Networks (VANETs), is playing a vital role in the future safety and ease of our roads. VANETs will enhance driver safety and reduce traffic deaths and injuries by implementing collision avoidance and warning systems. In addition, VANETs can relieve traffic congestion by providing a driver with live routes that avoid road hazards and bottleneck areas. The vast sensor network that VANETs will create, is inciting countless other applications, and making VANETs a hot topic in ad hoc networking today.

The VANET scenario has various difficult challenges for communication, many of which can be addressed by a clustered network. VANETs have a highly-mobile environment with a rapidly changing network topology. By clustering the vehicles into groups of similar mobility, the relative mobility between communicating neighbor nodes can be reduced. Both delay-intolerant (e.g. safety messages) and delay-tolerant (e.g. road/weather information) data will need to be transmitted, necessitating Quality-of-Service (QoS) requirements. Clustering has been shown to effectively reduce data congestion [1], and can support QoS requirements [2]. In addition, traffic jams and high node density in urban areas will be inevitable. This leads to contention and the hidden terminal problem, which are the performance limiting factors in a dense network and can be effectively alleviated by a clustered topology.

This work was supported in part by AUTO21 NCE and MARK IV Industries - IVHS Division

There has been much research on cluster-based VANETs in the recent literature, which has been focused on developing cluster-based MAC protocols, as in [3 – 9]. and cluster-based routing protocols, as in [10] and [11]. In [4] and [9], the cluster head (CH) takes on a managerial role and facilitates intra-cluster communication by providing a TDMA schedule to its cluster members. In [9], adjacent clusters are assigned different CDMA codes to avoid interference between clusters. The work in [9] shows a substantial reduction in probability of message delivery failure, when compared to traditional 802.11 MAC.

The recent research discussing cluster-based MACs and routing schemes for VANETs, present many similar low-maintenance clustering algorithms. In these schemes nodes are required to transmit periodic HELLO beacons to indicate their present state of either Undecided, Cluster Head or Cluster Member. An undecided node will join the first CH that it hears a HELLO beacon from, and if the node does not hear from a CH within a given time period, it will become a CH itself. In these schemes, the first node to declare CH status wins.

Node mobility should play an integral part in cluster creation in order to achieve stability. In [4], mobility is addressed during cluster collision; when two cluster heads come within range, the winning CH will be the one with both lower relative mobility and closer proximity to its members. Alternatively, [8] addresses mobility by first classifying nodes into speed groups, such that nodes will only join a CH of similar velocity. The above clustering techniques are lacking in cluster stability, because they do not attempt to select a stable CH during initial cluster head election.

Although there is not a VANET clustering scheme focused on cluster stability, there are many mobility-based clustering techniques for ad hoc networks. A well-known mobility-based clustering technique is MOBIC [12], which is an extension of the Lowest-ID algorithm [13]. In Lowest-ID, each node is assigned a unique ID, and the node with the lowest ID in its two-hop neighborhood is elected to be the cluster head. In MOBIC, an aggregate local mobility metric is the basis for cluster formation instead of node ID. The node with the smallest variance of relative mobility to its neighbors is elected as the cluster head. The relative mobility for a certain node is estimated by comparing the received power of two

consecutive messages from each neighboring node. Cluster head re-election only occurs when two cluster heads move within range of one another for a certain contention interval. When a cluster member moves out of range of its cluster head, it joins any current cluster head in its neighborhood, or forms a new cluster.

The previous cluster-based VANET research motivates the need for a stable VANET clustering scheme. In this paper, we propose a distributed mobility-based clustering algorithm focused on cluster stability, where stability is defined by long cluster head duration, long cluster member duration, and low rate of cluster head change. We achieve this algorithm by utilizing a new data clustering technique called Affinity Propagation (AP) [14]. Our clustering scheme will form clusters with both minimum distance and minimum relative velocity between each cluster head and its members. We assume position information is provided by the vehicles GPS. We validate our proposed algorithm by comparing it to the mobility-based ad-hoc clustering scheme, MOBIC [12]. MOBIC is well-established and stability driven, thus providing a good benchmark for our algorithm's success.

The rest of this paper is organized as follows. Section II presents the Affinity Propagation algorithm. Section III proposes our distributed, mobility-based, VANET clustering algorithm. Section IV presents our simulation results, and finally this paper concludes with Section V.

II. Affinity Propagation

Thus far, we have discussed clustering from an ad hoc networking perspective, however clustering is also used in scientific data analysis, where it aims to process and detect patterns in data. Data clustering is a static, one-shot process that searches data for a set of centers, or *exemplars*, which best describe the data. In this context, clustering aims to minimize the distance between each data point and its assigned exemplar, where distance could be Euclidian distance, or any other application-specific function. A revolutionary new technique for data clustering is the Affinity Propagation (AP) algorithm [14], which has been shown to produce clusters in much less time, and with much less error than traditional techniques (such as K-means clustering [15]). Here, clustering error refers to the application-specific distance between each data point and its assigned exemplar. In Affinity Propagation, data points pass messages to one another, which describe the current affinity that one data point has for choosing another data point as its exemplar.

This algorithm takes an input function of similarities, $s(i,j)$, where $s(i,j)$ reflects how well suited data point j is to be the exemplar of data point i. Affinity Propagation aims to maximize the similarity $s(i,j)$ for every data point i and its chosen exemplar j, therefore an application requiring a minimization (e.g. Euclidean distance) should have a negative similarity function. Each node i also has a self-similarity, $s(i,i)$, which influences the number of exemplars that are identified. Individual data points that are initialized with a larger self-similarity are more likely to become exemplars. If all the data points are initialized with the same constant self-similarity, then all data points are equally likely to become exemplars. By increasing and decreasing this common self-similarity input, the number of clusters produced is increased and decreased respectively.

There are two types of messages passed in this technique. The *responsibility*, $r(i,j)$, is sent from i to candidate exemplar j and indicates how well suited j is to be i's exemplar, taking into account competing potential exemplars. The *availability*, $a(i,j)$, is sent from candidate exemplar j back to i, and indicates j's desire to be an exemplar for i based on supporting feedback from other data points. The *self-responsibility*, $r(i,i)$ and *self-availability*, $a(i,i)$, both reflect accumulated evidence that i is an exemplar.

The update formulas for responsibility and availability are stated below:

$$r(i,j) \leftarrow s(i,j) - \max_{j' \, s.t. \, j' \neq j} \left\{ a(i,j') + s(i,j') \right\} \tag{1}$$

$$\underset{i \neq j}{a(i,j)} \leftarrow \min \left\{ 0, r(j,j) + \sum_{\forall i' \notin \{i,j\}} \max \left\{ 0, r(i',j) \right\} \right\} \tag{2}$$

$$a(j,j) \leftarrow \sum_{i' \, s.t. \, i' \neq j} \max \left\{ 0, r(i',j) \right\} \tag{3}$$

Responsibility and availability message updates must be damped to avoid numerical oscillations that will prevent the algorithm from converging. This is done by updating new messages as follows: $m_{new} = \lambda m_{old} + (1-\lambda) m_{new}$, where λ is a weighting factor between 0 and 1. In AP, the clustering is complete when the messages converge. Another feature of the algorithm is the ability to determine when a specific data point has converged to cluster head status in its given cluster. When a point's self-responsibility plus self-availability becomes positive, that point has become the cluster head.

Upon convergence, each node i's cluster head is:

$$CH_i = \arg\max_j \{ a(i,j) + r(i,j) \} \tag{4}$$

III. Proposed VANET Clustering Scheme

The proposed clustering technique uses the fundamental idea of Affinity Propagation from a communications perspective and in a distributed manner. We call this algorithm, Affinity PROpagation for VEhiclar networks, (APROVE). In our algorithm, each node in the network transmits the responsibility and availability messages to its neighbors, and then makes a decision on clustering independently. This results, in a distributed algorithm, where every node is only clustering with those in its one-hop neighborhood.

We design a similarity function for our algorithm with the goal of creating stable clusters, and tailored to the VANET environment. Our similarity function, shown below in (5), is a combination of the negative Euclidean distance between node

positions now and the negative Euclidean distance between node positions in the future. This is a simple way to consider both node position and node mobility in cluster creation.

$$s(i,j) = -\left(\|\mathbf{x_i} - \mathbf{x_j}\| + \|\mathbf{x'_i} - \mathbf{x'_j}\|\right) \quad (5)$$

$$\mathbf{x_i} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \mathbf{x'_i} = \begin{bmatrix} x_i + v_{x,i}\tau_f \\ y_i + v_{y,i}\tau_f \end{bmatrix}$$

where $\mathbf{x_i}$ is a vector of node i's current position, and $\mathbf{x'_i}$ is a vector of node i's predicted future position. The function predicts each node i's future position in τ_f seconds from now, based on node i's current velocity $v_{x,i}$ in the x direction and velocity $v_{y,i}$ in the y direction. The Time Future parameter, τ_f, can be tuned for different types of mobility.

The self-similarities were initialized to the same value and was set such that the number of clusters produced was minimized. We gave equal preference to each node, however it is possible to assign certain vehicles (such as large trucks) a higher preference, making them more likely to become the cluster head.

A. Message Passing and the Neighbor List

Every node i will maintain a neighbor list, $\mathbf{N_i}$, which has a neighbor list entry, $\mathbf{N_i^j}$, for every neighbor j. Each neighbor list entry, $\mathbf{N_i^j}$ contains the following fields:

$(x,y)_j$:	position vector of node j
$(v_x,v_y)_j$:	velocity vector of node j
$s(i,j)$:	similarity for i and j
$a(i,j)$:	last availability received from j
$a(j,i)$:	last availability transmitted to j
$r(j,i)$:	last responsibility received from j
$r(i,j)$:	last responsibility transmitted to j
$CH_{cnvg,j}$:	cluster head converge flag for node j
CH_j:	Index of j's current cluster head
t_{expire}:	Time that node j expires

Each node j will periodically broadcast a **HELLO** beacon containing its ID, position, velocity and current cluster head. The hello broadcast period is defined as T_H, where we have used $T_H = 1s$ in our simulations. Upon reception of a **HELLO** beacon from node j, node i will calculate its current similarity with j, $s(i,j)$, using (5) and update its neighbor list with the new information. A node only considers neighbors moving in the same direction, and ignores broadcasts from traffic in the opposite direction. This procedure is outlined in Procedure 1.

Procedure 1 Broadcast and Reception of Hello Beacons

1) Every T_H, each node j broadcasts **HELLO** beacon: $\langle j, (x,y)_j, (v_x,v_y)_j, CH_j \rangle$
2) Each receiving neighbor, i, checks if j is traveling in the same direction
3) If **true**, i calculates similarity with j, $s(i,j)$
4) Node i adds/updates j's neighbor list entry, $\mathbf{N_i^j}$'s: $\langle j, (x,y)_j, (v_x,v_y)_j, s(i,j), t_{expire}, CH_j \rangle$

The broadcast period for availability and responsibility messages is defined as T_M, which we set to $1s$ in our simulations. Each node i will calculate its responsibility with each neighbor j using (1). This value is damped with the previous transmitted responsibility (where $\lambda = 0.5$), and stored as $r(i,j)$. Node i then accumulates $r(i,j)$ for each neighbor j in the responsibility array, $\mathbf{R_i}$, and broadcasts the array in the **RESP** packet. Each node i will calculate the availability with each neighbor j using the update equation (2). Node i will store j's damped availability in $a(j,i)$ and accumulates all $a(j,i)$'s in the availability array, $\mathbf{A_i}$. This array is broadcast in the **AVAIL** packet.

The **AVAIL** packet also includes the flag CH_{cnvg}. Due to the nature of the AP algorithm, a node's self-responsibility plus self-availability will become positive when it has converged to cluster head status. For every iteration of the algorithm, each node i checks for this condition, and then sets the CH_{cnvg} flag accordingly. This flag indicates to i's neighbor nodes whether or not they should consider i as a potential cluster head. The responsibility and availability broadcast procedure is outlined in Procedure 2.

Procedure 2 Broadcast of **RESP** and **AVAIL** messages

Every T_M, each node i will:

1) Calculate responsibility, $r(i,j)$ for each neighbor j
2) Update with damping factor: $r(i,j) = (1-\lambda)r(i,j)^{new} + \lambda r(i,j)^{old}$
3) Store responsibilities, $r(i,j)$, in array: $\mathbf{R_i}$
4) Calculate availability, $a(j,i)$ for each neighbor j
5) Update with damping factor: $a(j,i) = (1-\lambda)a(j,i)^{new} + \lambda a(j,i)^{old}$
6) Store availabilities, $a(j,i)$, in array: $\mathbf{A_i}$
7) Determine if converged to CH status: if $r(i,i) + a(i,i) > 0$, then set CH_{cnvg}
8) Broadcast the **RESP** packet: $\langle \mathbf{R_i} \rangle$
9) Broadcast the **AVAIL** packet: $\langle \mathbf{A_i}, CH_{cnvg} \rangle$

When node i receives a **RESP** or **AVAIL** packet from j, it will search for its id in the $\mathbf{R_j}$ or $\mathbf{A_j}$ array, and if found, it reads off its specific responsibility or availability message. These message are stored in the received message fields, $r(j,i)$ or $a(i,j)$ of j's neighbor list entry, $\mathbf{N_i^j}$. If the received packet is of **AVAIL** type, node i will also update the $CH_{cnvg,j}$ field for j according to the CH_{cnvg} flag received. This routine is summarized in Procedure 3.

Procedure 3 Reception of **RESP** and **AVAIL** messages

Upon reception of a **RESP** or **AVAIL** packet from node j, node i will:

1) Search for its id, i in the $\mathbf{R_j}$ or $\mathbf{A_j}$ array.
2) If a message addressed to i is found, update the $r(j,i)$ or $a(i,j)$ field in the neighbor list entry, $\mathbf{N_i^j}$
3) Check if CH_{cnvg} flag is set, and update $CH_{cnvg,j}$ field in j's neighbor list entry, $\mathbf{N_i^j}$

B. Cluster Formation and Maintenance

Clustering decisions are made periodically with a period of CI called the Clustering Interval. Note that the T_M message period must be small enough to allow the algorithm to converge within a CI period. Preliminary simulations show that a neighborhood of 40 nodes can converge in under 10 iterations. (E.g. A T_M of $1s$, requires a minimum CI of 10s)

Every CI, node i finds its cluster head using (4). However, node i only considers its neighbors with the $CH_{cnvg,j}$ flag set, which confirms node j will become a cluster head. In the event that none of the neighbors have set their $CH_{cnvg,j}$ flag, node i becomes its own cluster head.

In between clustering iterations of CI, we perform cluster maintenance. Every T_{CM} (the period of cluster maintenance), node i purges its neighbor list of old entries by checking the t_{expire} fields. After purging, node i checks if its cluster head is still in its neighbor list. If the CH has been lost, node i searches through its neighbors for current CHs to join, by finding neighbors with $CH_j = j$. The $CH_{cnvg,j}$ flag is not used here because it indicates the potential cluster heads for the next round, not the current cluster heads. If multiple CHs are found in the neighbor list, node i uses (4) to select the best one. If node i can not find another neighbor that is currently a CH, it becomes its own cluster head.

There are some important notes regarding the passing of availability and responsibility messages. Firstly, the messages are not reset between clustering iterations, which gives memory to the algorithm and provides preference to previous cluster heads. This feedback results in less frequent cluster changes. Secondly, our algorithm does not assume synchronization, and each vehicle can run it independently of one another. In the asynchronous case, the received availability and responsibility messages will be at most one period old. Since these messages are averaged over time and a vehicle's movement over one time period is small, the algorithm's performance will not be effected. In the case of a lossy channel, messages can be older than one time period, which may cause the algorithm's performance to degrade. In this case, a more reliable MAC (such as [7]) can be used to increase the message reception probability.

IV. Simulation Results

We have implemented the proposed algorithm in NS2, which has been highly validated by the networking research community. All simulations were performed with 100 vehicles on a highway. Each simulation ran for 500s, however only the last 200s were used for performance metric calculations. This was to ensure that the algorithm had reached a steady state before measuring its performance. All of the simulation results were averaged over 10 different mobility scenarios, and used the following timing parameters: $T_H = T_M = T_{CM} = 1s$. The 802.11 MAC and the 914MHz Lucent WaveLAN DSSS network card with a radio range of 250m, have been used in the NS2 simulations. The MOBIC code was taken from a legacy version of NS2 provided by [12].

Highway scenarios were created using the VanetMobiSim traffic simulator [16], which generates realistic mobility patterns, including lane changing. The highway was generated in a looped formation with a 3km, 3-lane highway moving in each direction. The vehicles reach their maximum speed during the main highway pass, and then slow significantly at the turns, creating a realistic pattern with both low and high density traffic. The highways moving in either direction were separated by more than the 250m broadcast range, so that clustering could not occur across them. Our proposed algorithm will not cluster vehicles moving in opposite directions, but MOBIC will, which degrades cluster stability. This step was taken to provide a fair comparison of MOBIC and APROVE.

A. Clustering Performance Metrics

To evaluate the cluster stability and overall performance of our algorithm, we use the following metrics:

1) Average Cluster Head Duration: Long cluster head duration is important for MAC schemes where the cluster head is the central controller and scheduler.

2) Average Cluster Member Duration: This metric judges the overall stability of the initial clustering.

3) Average Rate of Cluster Head Change: This metric is useful since it takes into account both cluster head duration and the number of clusters formed.

4) Average Number of Clusters: To effectively decrease network contention, fewer clusters is desirable.

B. Performance Analysis

In the first set of simulations, the maximum velocity of the vehicles was 40m/s (144km/h). The Cluster Interval, CI, was swept from 10s to 150s and Time Future, τ_f, was swept from 0s to 120s. Figure 1 plots the effect of CI and τ_f on the clustering performance.

It can be seen from Figures 1a, 1b, and 1c that APROVE's cluster stability far exceeds that of MOBIC. Figure 1d indicates that MOBIC performs better in terms of number of formed clusters. However, the cluster change rate, which involves the joint effect of CH duration and the number of clusters, proves that APROVE's overall performance is superior. Notice that average cluster change rate is proportional to the number of clusters formed, and inversely proportional to the CH duration. From Figure 1c we can see that there is an optimal τ_f setting for each setting of CI. Based on this plot, a τ_f of 30s was chosen for future simulations, as it gave the best performance for this mobility scenario.

By sweeping over different velocities, we compare the clustering performance of APROVE and MOBIC. Using VanetMobiSim, highway scenarios were generated with approximate maximum velocities of 15, 25, 35, 40, and 50m/s. The performance results are displayed in Figure 2. Figure 2d shows that MOBIC generates a smaller number of clusters, however, it can be observed from Figures 2a, 2b, and 2c, that APROVE's stability performance far exceeds that of MOBIC. Figure 2c allows us to judge overall clustering performance, and when comparing APROVE and MOBIC at a typical highway speed

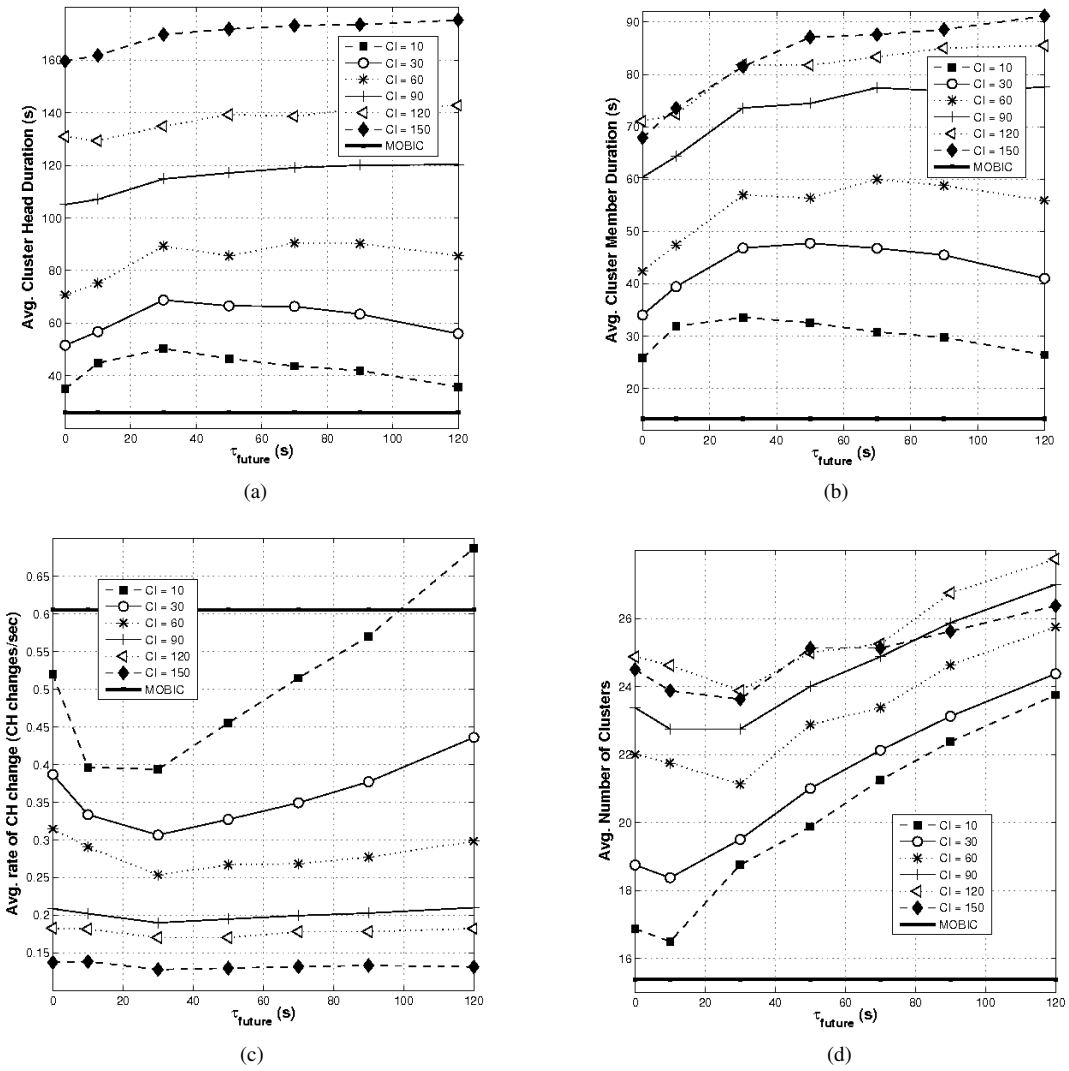

Fig. 1. The impact of τ_f on cluster performance for different CI settings. Simulations run with 100 nodes and maximum velocity = 40m/s. τ_f = 0, 10, 30, 50, 70, 90, and 120s, and CI = 10, 30, 60, 90, 120, and 150s. MOBIC's performance is also plotted. (a) The average cluster head duration. (b) The average cluster member duration. (c) The average rate of cluster head change. (d) The average number of clusters

of 40m/s, APROVE shows 300% improvement in average rate of cluster head change. It is also evident that while MOBIC's performance degrades at higher mobility, APROVE's performance degrades at lower mobility. A possible explanation for this trend is that parameter optimization was done for 40m/s, thus the timing parameters have been tuned to higher velocities.

MOBICs lesser stability performance could be caused by error in the mobility metric and cluster member suitability. The use of received power in the mobility metric can lead to inaccuracies in the relative mobility calculation due to channel fluctuations. In addition, the MOBIC algorithm does not consider cluster member suitability when forming clusters. Although the CH will initially have the lowest relative mobility in its neighborhood, a cluster member with high relative mobility will enter and exit the cluster quickly, and will choose a new cluster head without regard for the mobility metric.

V. Conclusion

Motivated by the ample research in cluster-based MACs and routing schemes for VANETs, we have proposed a novel and stable mobility-based clustering algorithm. Our algorithm elects cluster heads periodically, by using affinity propagation from a communications perspective, and in a distributed manner. The algorithm finds clusters that minimize both relative mobility and distance from each cluster head to its cluster members. The clusters created are stable and exhibit long average cluster member duration, long average cluster head duration, and low average rate of cluster head change. APROVE's high stability makes it a suitable candidate for clustering in a mobile and dynamic environment, such as VANETs.

References

[1] W. Chen and S. Cai, "Ad hoc peer-to-peer network architecture for vehicle safety communications," *Communications Magazine, IEEE*, vol. 43, no. 4, pp. 100–107, April 2005.

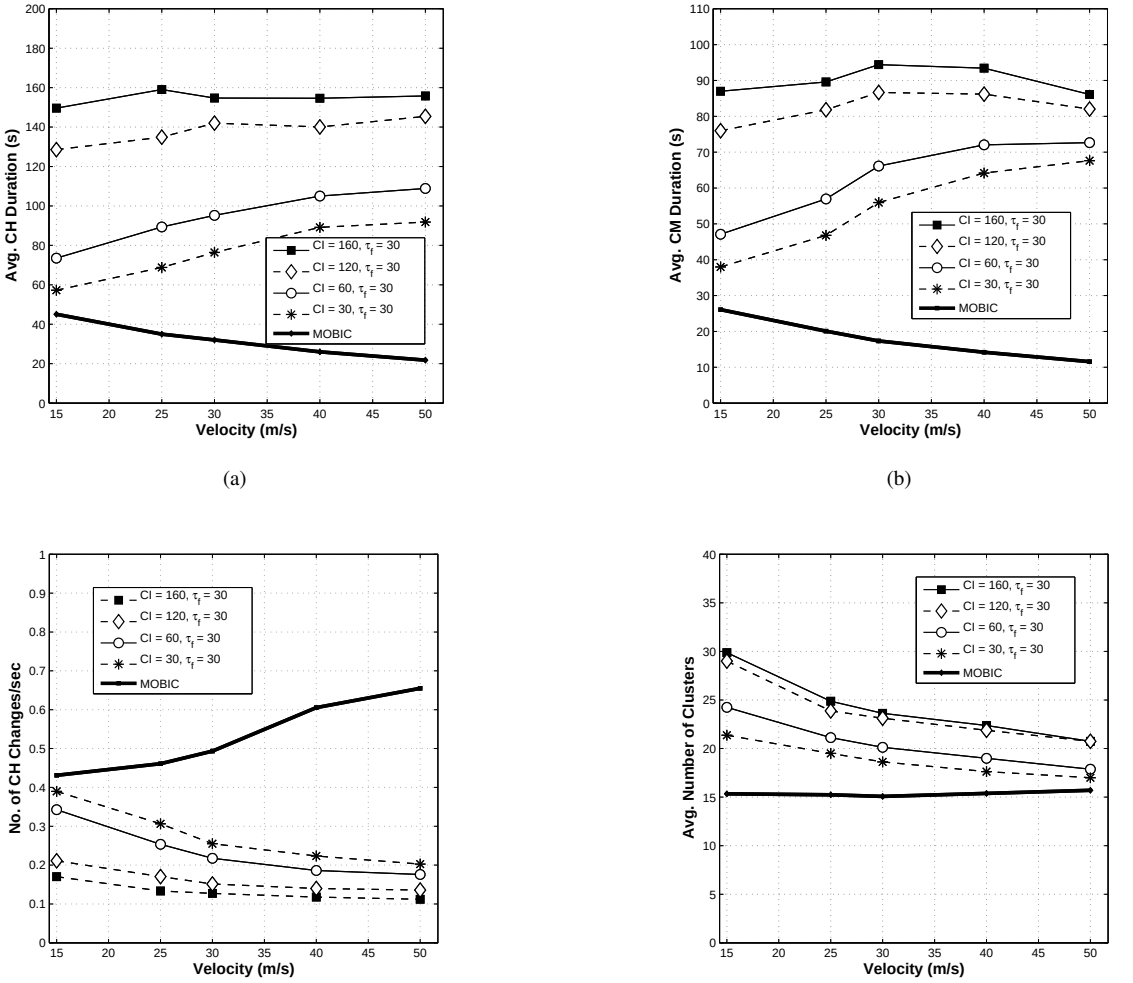

Fig. 2. The impact of velocity on cluster performance, comparing APROVE and MOBIC. APROVE is plotted with τ_f = 30s, and CI = 30, 60, 120, and 160s. Max velocity = 15, 25, 30, 40, and 50m/s. (a) The average cluster head duration. (b) The average cluster member duration. (c) The average rate of cluster head change. (d) The average number of clusters

[2] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks and Applications*, vol. 3, no. 1, pp. 101–119, 1998.

[3] L. Bononi and M. Di Felice, "A cross layered mac and clustering scheme for efficient broadcast in vanets," *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pp. 1–8, Oct. 2007.

[4] Y. Gunter, B. Wiegel, and H. Grossmann, "Cluster-based medium access scheme for vanets," *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 343–348, 30 2007-Oct. 3 2007.

[5] Z. Rawashdeh and S. Mahmud, "Media access technique for cluster-based vehicular ad hoc networks," *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pp. 1–5, Sept. 2008.

[6] F. Farnoud and S. Valaee, "Repetition-based broadcast in vehicular ad hoc networks in rician channel with capture," April 2008, pp. 1–6.

[7] B. Hassanabadi, L. Zhang, and S. Valaee, "Index coded repetition-based mac in vehicular ad-hoc networks," Jan. 2009, pp. 1–6.

[8] O. Kayis and T. Acarman, "Clustering formation for inter-vehicle communication," *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 636–641, 30 2007-Oct. 3 2007.

[9] 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, Nov. 2007.

[10] B. Wiegel, Y. Gunter, and H. Grossmann, "Cross-layer design for packet routing in vehicular ad hoc networks," *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pp. 2169–2173, 30 2007-Oct. 3 2007.

[11] R. E. R.A. Santos and N. Seed, "Inter vehicular data exchange between fast moving road traffic using ad-hoc cluster based location algorithm and 802.11b direct sequence spread spectrum radio," *PostGraduate Networking Conference*, 2003.

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

[13] 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, Sep 1997.

[14] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[15] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[16] M. Fiore, J. Harri, F. Filali, and C. Bonnet, "Vehicular mobility simulation for vanets," *Simulation Symposium, 2007. ANSS '07. 40th Annual*, pp. 301–309, March 2007.