Dynamic Clustering of Self Configured Adhoc Networks Based on Mobility

Kyriakos Manousakis and John S. Baras

***Abstract*— If heterogeneous ad hoc battlefield networks are to scale to hundreds or thousands of nodes, then they must be automatically split into separate network domains. Domains allow routing, QoS and other networking protocols to operate on fewer nodes, with cross-domain interaction only through a few border nodes. This division greatly reduces overall overhead (e.g., routing overhead with n nodes goes from $O(n^2)$ to $O(n \log n)$) and allows protocols to be tuned to more homogenous conditions [1]. On the other hand, the benefits from grouping the nodes are obvious only when the clustering is done in a way that the overhead that is produced due to its application does not offset the gain from the grouping of nodes. A significant source of overhead is the reclustering of nodes due to dynamic changes in network topology. If we manage to minimize the effect of reclustering then we expect the performance of the network to be improved (e.g., more scalable and survivable network). In this work we try to identify the various mobility groups and cluster the nodes accordingly. By grouping together nodes with similar mobility characteristics we can minimize/eliminate the effect of reclustering, since the topology changes will not affect the intra cluster connectivity.**

***Index Terms*—adhoc networks, self configured networks, dynamic clustering, mobility.**

I. Introduction

THE survivability and deployment of Future Combat Systems (FCS) depends on the efficient and careful design of the algorithms which constitute the basic modules of their functionality. Two of the most important characteristics that these systems must have are the robustness and scalability. If we think of the existing algorithms designed for mobile ad hoc networks and study their performance we can observe that these algorithms perform poorly when there is a large number of participating nodes or when the nodes are more mobile than the algorithm can handle. Combining the last two observations, it is obvious that we can not consider large, flat mobile ad hoc networks.

Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation thereon.

Kyriakos Manousakis is with the Electrical and Computer Engineering and the Institute for Systems Research of University of Maryland, College Park, MD 20742 USA (e-mail: kerk@ isr.umd.edu).

John. S. Baras, is with the Electrical and Computer Engineering and the Institute for Systems Research of University of Maryland, College Park, MD 20742 USA (e-mail: baras@isr.umd.edu).

The generation of hierarchy seems to be part of the remedy for the design of survivable MANETs but has to be done carefully and efficient. There are many clustering algorithms designed for MANETs but their drawback is that they group the nodes without taking into consideration the characteristics of the network environment. If the clustering algorithm operates independently from the network dynamics then it is destined to fail and harm instead of improving the performance of the network. For the latter reason the intuitive approach is to incorporate the network dynamics in the generation of clusters so that the clustering can be adapted each time to the requirements of the network.

Consider the following example that proves the importance of our approach. Assume the following network environment, where the nodes 1-7 are static, so there is no variation of their initial position during the lifetime of the network (i.e., these nodes can be sensor nodes). The nodes 8-11 are mobile nodes, but they are moving as a group so they are relatively static (i.e., group of soldiers). The former group of nodes follows a cyclic trajectory around the static nodes.

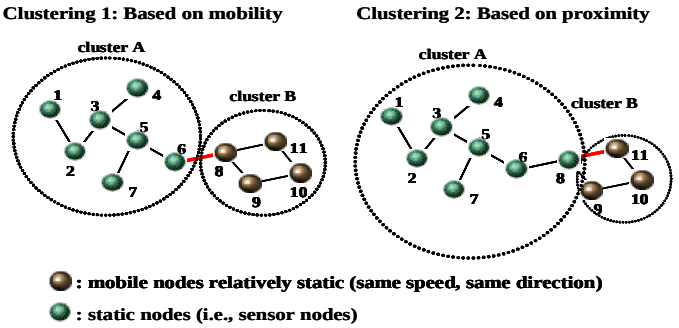


Fig. 1. Dynamic Clustering Motivation Example (mobility vs. proximity)

In this case if we attempt to cluster based on the proximity of the nodes or utilize the traditional methods of clustering (lower ID, highest degree) then we may end up with frequent re-clustering. The response of clustering algorithms to the topology changes is to re-cluster the network so as to maintain the clusters consistent to the principals of the specific algorithm. If we cluster based on the mobility of the nodes then we will not need any re-clustering, since, although the positions of the nodes change, their mobility characteristics remain the same. By using the mobility pattern as criterion for cluster generation we can achieve the following:

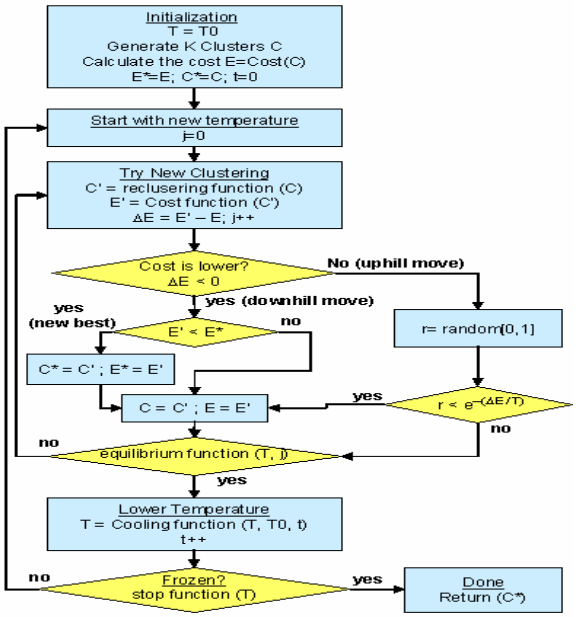
- More stable and robust clusters
- Minimize re-clustering/maintenance overhead

For the presentation of our approach we will introduce the appropriate methods and metrics that generate clusters subject to the mobility of the nodes. In the following paragraph we present the clustering algorithm we propose and the metrics/cost functions that we applied for the identification of the various mobility groups. In section 3, we present a prototype networking framework where the proposed clustering algorithms can be incorporated and applied in real world scenarios. The methods of interaction of the proposed algorithms with the specific networking framework are also discussed in the same section. Section 4 is related to the performance evaluation of the clustering algorithms. In the last section we will conclude this paper and will give some future directions.

II. Clustering Algorithm, Cost Functions and Metrics

A. Simulated Annealing

Simulated annealing (SA) has been widely used for tackling different combinatorial optimization problems [5]. The process of obtaining the optimum configuration is similar to that followed in a physical annealing schedule. In SA, however, the temperature is merely used as a control parameter and does not have any physical meaning.



Inputs	Examples
Equilibrium function	Constant (j = 5000); "Stop repeats" (function of
Cost function	$\sum_{i=1}^{k} Diameter\ (C_i)$
Cooling function	Geometric or logarithmic
Stop function	Minimum temperature (e.g., T=0.1); "Stop repeat" criteria
Reclustering function	Random move of one node
T0	Initial Temperature
K	Number of clusters

Variable	Definition
T	Current Temperature
C	Current Cluser map (e.g., {1,2},{3,6},{4,5})
C'	New cluser map to test (eg. {1,2},{3,5,6},{4})
C*	Champion cluster map (eg. {1,6},{3,5},{2,4})
E	Current cost
E'	Cost of new cluster map
E*	Champion cost
j	Inner loop counter
t	Outer loop counter

Fig. 2. Simulated Annealing algorithm for network partitioning

Figure 2 highlights the general steps in the algorithm. The objective of the algorithm is to obtain the K cluster network partition configuration, C^*, that optimizes a particular cost function. The process starts with an initial temperature value, T_0, which is iteratively decreased by the cooling function until the system is frozen (as decided by the stop function). For each temperature, the SA algorithm takes the current champion configuration C^* and applies the recursive function to obtain a new configuration C' and evaluates its cost, E'. If E' is lower than the cost of the current E^*, C' and E' replace C^* and E^*. Also, SA randomly accepts a new configuration C' even though E' is greater than E^* to avoid local minima. In the latter case C' and E' replace C^* and E^* *respectively*.

One of the key characteristics of simulated annealing is that it allows uphill moves at any time and relies heavily on randomization [6]. The higher the temperature, the higher the probability of accepting a configuration that worsens E^* instead of improving it. Indeed, if the temperature is sufficiently high, SA will simply take a random walk around the solution space. The lower the temperature, the lower the probability of accepting worse configurations.

The number of iterations required to reach equilibrium are defined by the equilibrium function. The function can be a simple constant (e.g., 100) or a function of the temperature and other parameters specific to the optimization problem, such as number of nodes in the network.

In order to complete the description of the proposed dynamic clustering algorithm we will present the metrics/cost functions we suggest so as to succeed in our objective, which is to cluster based on the mobility characteristics of the nodes for the reduction/elimination of the reclustering overhead and the improvement of robustness and scalability of the network. The next section elaborates on the problem of selecting the appropriate metrics/cost functions.

B. Metrics and Cost Functions

Our objective is to identify and group together the nodes that present similar mobility characteristics subject to the generation of topological domains[1]. The appropriate selection of metrics/cost functions in combination with SA will generate domains that are robust to the mobility of the nodes, since its effect on the intracluster connectivity will be minimized or even eliminated, because even though the topology will be changing, the nodes with similar mobility characteristics will be still moving together. By succeeding in this, we will have generated a hierarchy into our network, without penalizing performance because of reclustering overhead. This will result in network performance improvement because we can take advantage of the generated hierarchy without having to deal with the overhead of its maintenance.

There are various ways to characterize the mobility of the nodes. The cost function that we propose involves the direction of the mobile entities. In the following paragraphs we

will present the metric and the corresponding cost function that we applied in combination with SA for the clustering of the nodes.

The mobility of the nodes is characterized from two basic metrics: speed and direction. In this work we utilized only one of these two metrics and more specifically the cost function we propose is based on the direction of the nodes. The direction of the nodes is described from the angle that is defined counter-clockwise from the straight line that is defined from two consecutive points on the trajectory of the node and the straight line parallel to the positive x-axis ($\theta = 0^o$). Figure 3 gives a couple of examples:

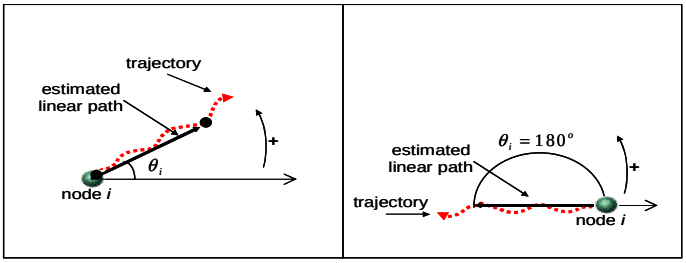


Fig. 3. Method to estimate the node direction

Since we have specified the metric, we need to define the appropriate cost function that will be optimized from the SA algorithm for the generation of clusters. The objective is to group together nodes with similar mobility characteristics. Specifically, due to the focus on the nodes direction metric, we expect the cost function to be successful in grouping together nodes that have similar directions. The reason behind this approach is that the nodes that will constitute these groups will most likely remain connected for a long period of time.

In the cost function we do not apply the raw value of the direction of each node but the relative direction of each pair of nodes that belong in the same cluster. The value of the cost function is evaluated after every iteration performed from the SA algorithm. The cost function is:

$$E = \min \sum_{z=1}^{K} \left(\sum_{i,j=1}^{|C_z|} \theta_{r_{i,j}} \right)^2 \qquad (1)$$

where

K : number of generated clusters

$|C_z|$: size of z^{th} cluster

$\theta_{r_{i,j}}$: Relative direction of nodes *i* and *j*

- Relative Direction $\theta_{r_{i,j}}$

  The relative direction of each pair of nodes *i,j* of the same cluster is defined as follows:

$$\theta_{r_{i,j}} = \min\left(\left|\theta_i - \theta_j\right|, 360 - \left|\theta_i - \theta_j\right|\right)$$

The proposed cost function results in *K* clusters which consist of nodes that present similarity in the direction of their movement. As we are going to show in the performance evaluation section, (1) is very successful and accurate in identifying the various mobility groups in the network.

Apart from the above cost function that we will utilize in conjunction with the SA algorithm, we pose an extra constraint on the generation of clusters. We require that the resulting clusters are topological clusters. The definition of a topological cluster is:

Definition (**Topological Cluster)**: A cluster consisting of the set $\mathbb{S}$ of nodes is called topological if $\forall node_i, node_j \in \mathbb{S}$ and $i \neq j$, there is always a path P_{ij} from $node_i$ to $node_j$ s.t $\forall node_k \notin \mathbb{S}$ holds that $node_k \notin P_{ij}$.

In order to fulfill this extra constraint we allow the SA algorithm to search only among this set of feasible clustering maps. In order for a clustering map (CM) to be eligible for evaluation during the execution of SA, it has to involve only topological clusters. This constraint is included in the implementation of the SA algorithm, and we aim on the minimization of (1) by searching only among the set of feasible clustering maps.

III. Network Architecture for the Application of the Clustering Algorithms

A. IP Autoconfiguration Suite – IPAS

Rapidly deployable and survivable networks are very important requirements in the Objective Force. Thus, in order to support these requirements, the entire tactical battlefield network, possibly consisting of thousands of hosts, routers and MANET nodes, must be autoconfigured. Moreover, the networks must be rapidly reconfigured as conditions or requirements change. In this section, we present an approach to plug-and-play and survivable networking using the IP Autoconfiguration Protocol Suite (IPAS) [4]. We describe the IPAS protocol architecture, its elements and their functionalities.

Figure 4 shows the IPAS components and how they relate to each other. At its heart is the new Dynamic Configuration Distribution Protocol (**DCDP**). DCDP is a robust, scalable, low-overhead, lightweight (minimal state) protocol designed to distribute configuration information on address-pools and other IP configuration information (e.g., DNS Server's IP address, security keys, or routing protocol). DCDP was designed for dynamic wireless battlefield, operating without any central coordination or periodic messages. Moreover, DCDP does not require a routing protocol to distribute information or any interface to be configured (except for the link-local information in IPv6).

[1] Topological domains: Group of nodes/interfaces such that each pair of them can communicate with every other node/interface of the same topological domain only through nodes that belong in this domain.

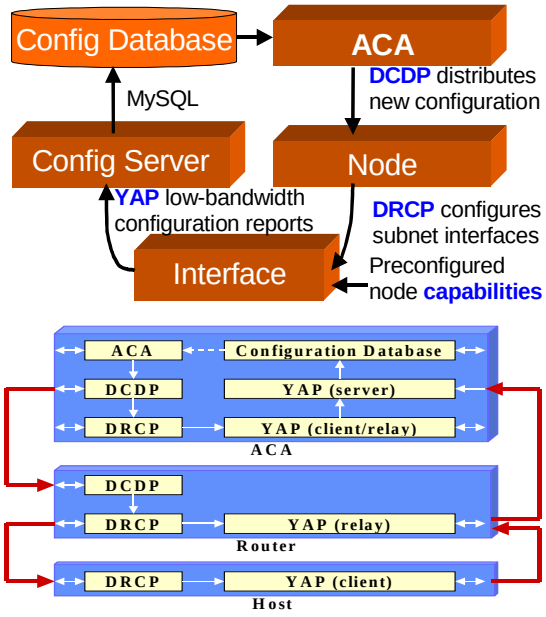


Fig. 4. IPAS network and node models

DCDP relies on the Dynamic and Rapid Configuration Protocol (**DRCP**) to actually configure the interfaces. DRCP borrows heavily from DHCP, but adds features critical to roaming users. DRCP can automatically detect the need to reconfigure (e.g., due to node mobility) through periodic advertisements. In addition, DRCP allows for: a) efficient use of scarce wireless bandwidth, b) dynamic addition or deletion of address pools to support server fail over, c) message exchange without broadcast, and d) clients to be routers.

The Configuration Database Update Protocol (**YAP**) is a simple bandwidth efficient reporting mechanism for dynamic networks. YAP has three elements: 1) YAP Clients running on every node, 2) YAP Relays forwarding information from YAP clients to a server, and 3) a YAP Server. YAP clients periodically report its node's capabilities, configuration, and operational status to the YAP relay agents. The *capabilities* say, for example: "This node can be a DNS server with priority 0" or "a YAP server with priority 3" (priority reflecting a node's willingness to perform a function). Other YAP information include the node's: 1) name and IP address, 2) Rx/Tx packets, bit rate, link quality, 3) routing table, and 4) address pool. The YAP server stores this information in a configuration database (see Figure 1).

The brain of IPAS is the Adaptive Configuration Agent (ACA). It observes the state of the network in the Configuration Database (filled by YAP) and can perform some actions, such as server reconfiguration, based on some rules or policies. The ACA can also reset the network and can distribute an address pool from human input or from a predefined private address pool (e.g., 10.x.x.x).

B. *Application of Clustering Algorithm*

Our main objective is to design efficient clustering techniques that can be successful in dynamic and distributed network environments. Even though the proposed clustering algorithm is based on Simulated Annealing which is a centralized global optimization algorithm, it can successfully fit the targeted network environment. The architectural framework in which SA can be applied for the clustering of distributed and dynamic networks (i.e., adhoc networks) is described in this section.

If we utilize IPAS to configure and maintain the mobile adhoc network, then we can easily incorporate SA for the dynamic generation of clusters. Even though IPAS is a distributed algorithm, the configuration decisions are originated from a centralized entity, the ACA (Adaptive Configuration Agent). If we attach a SA module to the ACA then this centralized entity will be enriched with the extra capability of generating clustering decisions which can be distributed to the network following the exact same mechanism that is used for the distribution of the network entity configuration decisions. Using the already existing IPAS architecture, the application of a centralized algorithm (i.e., SA) is realizable. In the realization of SA, there are two more points that they haven't been clarified yet. The first has to do with the collection of metrics that SA utilizes for the clustering of nodes. The values of these metrics have to be collected from the network in real time. Once more the design of IPAS can help us achieve that without the development of new modules. A module of IPAS is YAP which collects configuration information from the network and stores it in a centralized database. This database is accessible from ACA and is used for the generation of the appropriate configuration decisions. YAP can be slightly modified to collect from the network the metrics of interest that can be used later from the SA algorithm in the generation of clusters. The second point for the realization of SA has to do with the post clustering decision generation phase and specifically with the distribution of the clustering decisions to the nodes. Up to this point ACA distributes configuration decisions that target specific network interfaces, but now we want to distribute configuration information that targets a group of interfaces. We have already designed this mechanism by extending ACA, DCDP and DRCP to distribute and process new domain configuration messages [1].

IV. Performance Evaluation

In this section we will present the results we collected for the evaluation of the proposed clustering algorithm. We will characterize the performance of the algorithm from two different perspectives. Initially, we will present the results we collected related to the accuracy of the algorithm in the identification of the various mobility groups that exist into the network. Results related to the speed of the algorithm will follow. The latter class of results is important for the characterization of the approach in terms of its effectiveness in a dynamic environment. Since, we want to apply the SA in a dynamic environment, we want to show that the algorithm that generates the clusters can be fast enough to cope with the

dynamic changes happening in the network.

Before going into the presentation details of the collected results we will highlight the important points of the experiments' set up. We implemented the SA algorithm on a LINUX platform (Redhat 9.0), which was running on a P4 machine (CPU:1.2GHZ, SDRAM: 256MB). We generated an application for the generation of connected networks. This application gets as inputs the desired size (e.g., number of nodes) of the network to be generated and the 2D dimensions of the area in which these nodes are going to be distributed. Two sample networks which were used in the evaluation of the proposed algorithm are shown.

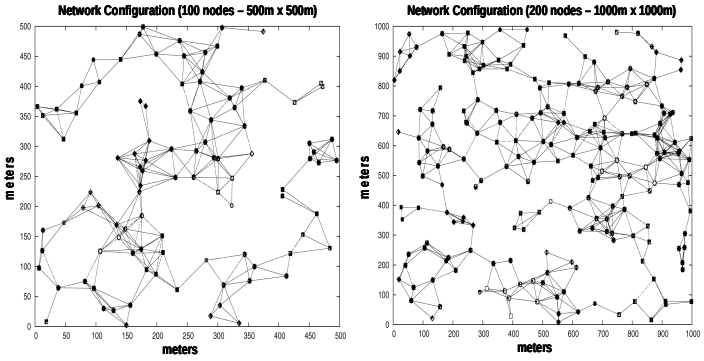


Fig. 5. Sample Network Configurations: 100 nodes (left), 200 nodes (right)

In order to test the accuracy of the algorithm on the identification of the various mobility groups we designed specific experiments so we could evaluate easier the performance of the algorithm. We manually assigned nodes to mobile groups so we could know beforehand the configuration of the mobile groups. We did that so after the application of the clustering algorithm we could compare the generated clusters with the preassigned mobile groups. We utilized the Reference Point Group Mobility (RPGM) [7] model for the movement of nodes. In RPGM we define a number of RP equal to the number of mobility groups we want to generate. To complete the generation of mobility groups each node is assigned to a Reference Point (RP). The movement of the nodes is defined from the mobility patterns of their corresponding RPs. These mobility patterns are assigned manually to the various RPs in the form of trajectories. When a RP moves to a new location each corresponding node is assigned to a random radius and direction around the new position of the RP. Because of the functionality of RPGM model and the randomness in the selection of the new node position, it is obvious that nodes that belong into the same group may have different speeds and directions, which makes our work of identifying the various mobility groups more difficult but makes the evaluation of our clustering approach more general.

The following results were collected by assuming two mobility groups, where the corresponding RPs where moving on a straight line and in constant relative direction θ_{RP_1,RP_2}. We varied the relative direction from 0^o to 360^o (e.g., $\theta_{RP_1,RP_2} \in \left[0^o...360^o\right]$) with a step of 15^o. In each run we measured the percentage (%) of nodes that they were assigned in an incorrect cluster. The variation of this percentage versus θ_{RP_1,RP_2} is given in the following graph:

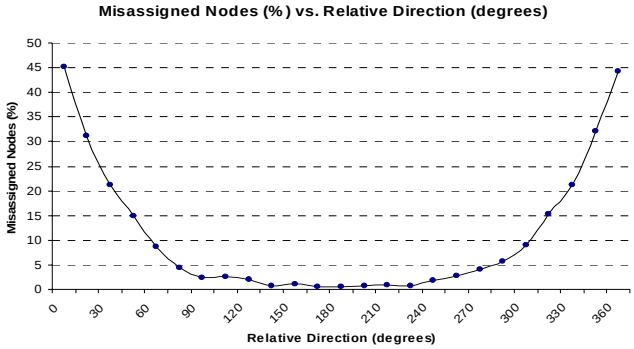


Fig. 6. Percentage of nodes that have been assigned incorrectly vs. the relative direction of the mobile groups

From the above figure, the cost function can identify accurately the various mobility groups especially when the groups are moving in relative directions such that $\theta_{RP_1,RP_2} \in \left[30^o...330^o\right]$. When $\theta_{RP_1,RP_2} \notin \left[30^o...330^o\right]$ then the proposed cost function has difficulty to identify the mobility groups. This is not a limitation of the algorithm since in this scenario, the selection of mobility groups is not restricted to the original mobility groups, because of the similarity in their directions. Even in this case the reclustering overhead will be low, because the nodes of the mobility groups are moving towards an almost similar direction.

The other important class of results is related to the convergence speed of the proposed algorithm. We measured the time to the completion of the algorithm for various network sizes. We let the number of nodes vary from 50 to 1000 with steps of 50 nodes. The collection of the running times was done with the help of the *gettimeofday(*struct timeval **tv*, struct timezone **tz)* function that exists under the UNIX/LINUX systems. The following graph represents the performance of the algorithm:

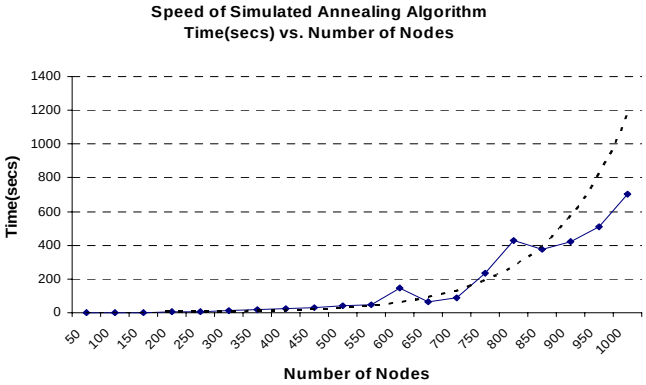


Fig. 7. Speed of Simulated Annealing algorithm measured in time (secs) vs. the size of the network

From figure 7, we can conclude that the proposed clustering algorithm can be used effectively as an optimization technique under either of the following two conditions a) small networks independently of network dynamics (e.g. few hundreds of nodes), or b) large networks with low rate of topology changes (i.e. sensor networks where we have none or very slow movement of the nodes). The latter conclusions are justified from the fact that the running time of SA increases exponentially with the network size.

V. Conclusions and Future Work

In this work we presented a novel approach for the dynamic organization of MANETs into clusters. The generation of hierarchy can only help the network if it is done in a way that the clustering/reclustering overhead is minimized or eliminated. To achieve our objectives, our approach takes into consideration the network environment. We follow this philosophy by attempting to cluster the nodes based on their mobility characteristics – nodes with similar mobility characteristics are clustered together, in order to minimize the effect of topology changes on the clustering overhead.

We introduced a cost function that is based on the relative direction of nodes. We showed that the cost function is very accurate in identifying the various mobility groups, especially when $\theta_{RP_1,RP_2} \in [30^o...330^o]$. For the cases, where $\theta_{RP_1,RP_2} \notin [30^o...330^o]$, the algorithm may not be very accurate subject to the predefined mobility groups, but this does not suggest that the clustering will result in large overhead, because of the similarity of nodes direction.

Our intention is to apply the proposed algorithm in dynamic environments. So, we had to evaluate the ability of the algorithm to capture the dynamics of the network. We presented the time required from the algorithm until its completion for various network sizes. The initial indication is that the algorithm can be applicable in small, quickly changing networks or in larger but slowly changing ones.

Currently we are investigating the improvement of the proposed clustering approach by looking into two different directions. The first one is related to the identification of the mobility groups, where we will try to make our approach more accurate by incorporating in the cost function the velocity and location of nodes. We expect that the utilization of the combination of metrics can be even more effective than utilizing only the node direction. The second direction has to do with the time to completion (speed) of SA algorithm where we aim on improving its speed by configuring appropriately the various input parameters of the algorithm. The latter can produce suboptimal clustering maps but if this is done carefully we can significantly improve the convergence speed of the algorithm and at the same time we can still obtain very good clustering maps.

References

[1] Manousakis K., McAuley J., Morera R., Baras J., "Routing Domain Autoconfiguration for More Efficient and Rapidly Deployable Mobile Networks," Army Science Conference 2002, Orlando, FL

[2] Lin Chunhung Richard and Gerla Mario, "Adaptive Clustering for Mobile Wireless Networks," IEEE Journal on Selected Areas in Communications, pages 1265-1275, September 1997

[3] Baker D., Ephremides A., and Flynn J. "The design and simulation of a mobile radio network with distributed control," IEEE Journal on Selected Areas in Communications, SAC-2(1):226--237, 1984

[4] McAuley A., Misra A., Wong L., Manousakis K., "Experience with Autoconfiguring a Network with IP addresses," IEEE Milcom, October 2001.

[5] Kirkpatrick, S, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science 220 (13 May 1983), 671-680

[6] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," in E. H. Aarts and J. K. Lenstra (eds.), "Local Search in Combinatorial Optimization," John Wiley and Sons, Ltd., pp. 215-310, 1997.

[7] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in ACM/IEEE MSWiM, August 1999.