Cluster Computing 5, 193–204, 2002

WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks *

MAINAK CHATTERJEE, SAJAL K. DAS and DAMLA TURGUT
Center for Research in Wireless Mobility and Networking (CReWMaN), Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019-0015, USA

Abstract. In this paper, we propose an on-demand distributed clustering algorithm for multi-hop packet radio networks. These types of networks, also known as *ad hoc* networks, are dynamic in nature due to the mobility of nodes. The association and dissociation of nodes to and from *clusters* perturb the stability of the network topology, and hence a reconfiguration of the system is often unavoidable. However, it is vital to keep the topology stable as long as possible. The *clusterheads*, form a *dominant set* in the network, determine the topology and its stability. The proposed weight-based distributed clustering algorithm takes into consideration the ideal degree, transmission power, mobility, and battery power of mobile nodes. The time required to identify the clusterheads depends on the diameter of the underlying graph. We try to keep the number of nodes in a cluster around a pre-defined threshold to facilitate the optimal operation of the medium access control (MAC) protocol. The non-periodic procedure for clusterhead election is invoked on-demand, and is aimed to reduce the computation and communication costs. The clusterheads, operating in "dual" power mode, connects the clusters which help in routing messages from a node to any other node. We observe a trade-off between the uniformity of the load handled by the clusterheads and the connectivity of the network. Simulation experiments are conducted to evaluate the performance of our algorithm in terms of the number of clusterheads, *reaffiliation* frequency, and dominant set updates. Results show that our algorithm performs better than existing ones and is also tunable to different kinds of network conditions.

Keywords: ad hoc networks, clusters, dominant set, load balancing

1. Introduction

Current wireless cellular networks solely rely on the wired backbone by which all base stations are connected, implying that networks are fixed and constrained to a geographical area with a pre-defined boundary. Deployment of such networks takes time and cannot be set up in times of utmost emergency. Therefore, mobile multi-hop radio networks, also called *ad hoc* or *peer-to-peer* networks, play a critical role in places where a wired (central) backbone is neither available nor economical to build, such as law enforcement operations, battle field communications, disaster recovery situations, and so on. Such situations demand a network where all the nodes including the base stations are potentially mobile, and communication must be supported untethered between any two nodes.

A multi-cluster, multi-hop packet radio network architecture for wireless systems should be able to dynamically adapt itself with the changing network configurations. Certain nodes, known as *clusterheads*, are responsible for the formation of *clusters* each consisting of a number of nodes (analogous to *cells* in a cellular network) and maintenance of the topology of the network. The set of clusterheads is known as a *dominant set*. A clusterhead does the resource allocation to all the nodes belonging to its cluster. Due to the dynamic nature of the mobile nodes, their association and dissociation to and from clusters perturb the *stability* of the network and thus reconfiguration of clusterheads is unavoidable. This is an important issue since frequent clusterhead changes adversely affect the performance of other protocols such as scheduling, routing and resource allocation that rely on it. Choosing clusterheads optimally is an NP-hard problem [4]. Hence existing solutions to this problem are based on heuristic (mostly greedy) approaches and none attempts to retain the stability of the network topology [4,9]. We believe a good clustering scheme should preserve its structure as much as possible when nodes are moving and/or the topology is slowly changing. Otherwise, re-computation of clusterheads and frequent information exchange among the participating nodes will result in high computation overhead.

The concept of dividing the geographical region to be covered into small zones has been presented implicitly as *clustering* in the literature [13]. A natural way to map a "standard" cellular architecture into a multi-hop packet radio network is via the concept of a virtual cellular network [9]. Any node can become a clusterhead if it has the necessary functionality, such as processing and transmission power. Nodes register with the nearest clusterhead and become members of that cluster. Clusters may change dynamically, reflecting the mobility of the underlying network. The focus of the existing literature in this area has mostly been on partitioning the network into clusters [4,5,11,17,18], without taking into consideration the efficient functioning of all the system components. The lack of rigorous methodologies applicable to the design and analysis of peer-to-peer mobile networks has motivated in-depth research in this area. There have been solutions for

* This work is partially supported by Texas Advanced Research Program grant TARP-003594-013, Texas Telecommunications Engineering Consortium (TxTEC) and Nortel Networks.

efficient ways of interconnecting the nodes such that the latency of the system is minimized while throughput is maximized [11]. Most of the approaches [4,9,11] for finding the clusterheads do not produce an optimal solution with respect to battery usage, load balancing and MAC functionality.

In this paper, we propose a weight based distributed clustering algorithm which takes into consideration the number of nodes a clusterhead can handle ideally (without any severe degradation in the performance), transmission power, mobility, and battery power of the nodes. Unlike other existing schemes which are invoked periodically resulting in high communication overhead, our algorithm is *adaptively* invoked based on the mobility of the nodes. More precisely, the clusterhead election procedure is delayed as long as possible to reduce the computation cost. Balancing the loads between clusterheads is another desirable feature of any clustering algorithm, however, it is very difficult to maintain a completely balanced system due to the dynamic nature of the nodes. Our algorithm achieves *load balancing* by specifying a pre-defined threshold on the number of nodes that a clusterhead can handle ideally. This ensures that none of the clusterheads are overloaded at any instance of time. We define *load balancing factor* (LBF) to measure the degree of load balancing among the clusterheads. Connecting the nodes is another important issue since the nodes need to communicate with each other. In order to consider any kind of routing between the clusters, it is essential that the clusters are connected and the nodes are able to route messages via the clusterheads. We define *connectivity* as the probability that a node is reachable from any other node. Clusterheads in our scheme work in dual power mode. The clusterheads can operate at a higher power mode (resulting in a higher transmission range) for inter-cluster communication while they use lower power for intra-cluster communication. Finally, we conduct detailed simulation experiments and demonstrate that our clustering algorithm yields better results as compared to the existing heuristics in terms of the number of reaffiliations (detachment of a node from its current cluster and attachment to another existing cluster) and dominant set updates.

The rest of the paper is organized as follows. In section 2, we summarize previous work and their limitations. In section 3, we propose the Weighted Clustering Algorithm (WCA). Simulation results are presented in section 4 while conclusions are offered in section 5.

2. Previous work

Several heuristics have been proposed to choose clusterheads in ad hoc networks. They include (i) Highest-Degree heuristic, (ii) Lowest-ID heuristic, and (iii) Node-Weight heuristic. The Lowest-ID and the Highest-Degree were the two clustering algorithms which were based on the *link-cluster architecture* [2,3,10]. In the assumed graph model of the network, the mobile terminals are represented as nodes; there exists an edge between two nodes if they can communicate with each other directly (i.e., one node lies within the transmission range of another). The performance of the above three heuristics were studied in [5,11] by simulation experiments in which mobile nodes were randomly placed in a square grid and moved with different speeds in different directions.

2.1. Highest-Degree heuristic

The Highest-Degree, also known as *connectivity-based clustering*, was originally proposed by Gerla and Parekh [11,20] in which the degree of a node is computed based on its distance from others. Each node broadcasts its id to the nodes that are within its transmission range. A node x is considered to be a neighbor of another node y if x lies within the transmission range of y. The node with maximum number of neighbors (i.e., maximum degree) is chosen as a clusterhead and any tie is broken by the unique node ids. The neighbors of a clusterhead become members of that cluster and can no longer participate in the election process. Since no clusterheads are directly linked, only one clusterhead is allowed per cluster. Any two nodes in a cluster are at most two-hops away since the clusterhead is directly linked to each of its neighbors in the cluster. Basically, each node either becomes a clusterhead or remains an ordinary node (neighbor of a clusterhead).

Experiments demonstrate that the system has a low rate of clusterhead change but the throughput is low under the Highest-Degree heuristic. Typically, each cluster is assigned some resources which is shared among the members of that cluster on a round-robin basis [11,17,18]. As the number of nodes in a cluster is increased, the throughput drops and hence a gradual degradation in the system performance is observed. The reaffiliation count of nodes are high due to node movements and as a result, the highest-degree node (the current clusterhead) may not be re-elected to be a clusterhead even if it looses one neighbor. All these drawbacks occur because this approach does not have any restriction on the upper bound on the number of nodes in a cluster.

2.2. Lowest-ID heuristic

The Lowest-ID, also as known as *identifier-based clustering*, was originally proposed by Baker and Ephremides [2,3,10]. This heuristic assigns a unique id to each node and chooses the node with the minimum id as a clusterhead. Thus, the ids of the neighbors of the clusterhead will be higher than that of the clusterhead. However, the clusterhead can delegate its responsibility to the next node with the minimum id in its cluster. A node is called a *gateway* if it lies within the transmission range of two or more clusterheads. Gateway nodes are generally used for routing between clusters. Only gateway nodes can listen to the different nodes of the overlapping clusters that they lie. The concept of distributed gateway (DG) is also used for inter-cluster communication only when the clusters are not overlapping. DG is a pair of nodes that lies in different clusters but they are within the transmission range of each other. The main advantage of distributed gateway is maintaining connectivity in situations where any clustering algorithm fails to provide connectivity.

For this heuristic, the system performance is better compared with the Highest-Degree heuristic in terms of throughput. Since the environment under consideration is mobile, it is unlikely that node degrees remain stable resulting in frequent clusterhead updates. However, the drawback of this heuristic is its bias towards nodes with smaller ids which may lead to the battery drainage of certain nodes. One might think that this problem may be fixed by renumbering the node ids from time to time, which is however non-trivial. There are other problems associated with such renumbering. For instance, the optimal frequency of renumbering would need to be determined so that the system performance is maximized. More importantly, every time node ids are reshuffled, the neighboring list of all the nodes need also to be changed. If we consider that the nodes are numbered in the increasing order of their remaining battery power, then a centralized algorithm is required. We can avoid this by exchanging ids between nodes and making sure that the uniqueness of ids are maintained. Even then, the clustering has to be redone which would add unnecessary computational complexity to the system. For example, suppose two nodes mutually exchange their ids in order to keep the ids according to their remaining battery power. After exchanging, all nodes that were connected to these two nodes, regardless of their status (clusterhead or ordinary node), need to be notified of the change so that they can update their neighbor list. This effect may propagate and add overhead to the system. Moreover, it does not attempt to balance the load uniformly across all the nodes.

2.3. Node-Weight heuristic

Basagni et al. [5,6] proposed two algorithms, namely *distributed clustering algorithm (DCA)* and *distributed mobility-adaptive clustering algorithm (DMAC)*. In this approach, each node is assigned weights (a real number $\geqslant 0$) based on its suitability of being a clusterhead. A node is chosen to be a clusterhead if its weight is higher than any of its neighbor's weight; otherwise, it joins a neighboring clusterhead. The smaller node id is chosen in case of a tie. The DCA makes an assumption that the network topology does not change during the execution of the algorithm. Thus, it is proven to be useful for "quasi-static" networks when the nodes either do not move or move very slowly. The other assumptions are: (i) the messages are guaranteed to be delivered to all of the nodes' neighbors within a finite amount of time, and (ii) every node is aware of the ids and the corresponding weights of all the nodes which are only one hop away. The DMAC algorithm, on the other hand, adapts itself to the network topology changes and therefore can be used for any mobile networks.

To verify the performance of the system, the nodes were assigned weights which varied linearly with their speeds but with negative slope. Results proved that the number of updates required is smaller than the Highest-Degree and Lowest-ID heuristics. Since node weights were varied in each simulation cycle, computing the clusterheads becomes very expensive and there are no optimizations on the system parameters such as throughput and power control.

2.4. Limitations of existing heuristics

None of the above three heuristics leads to an optimal election of clusterheads since each deals with only a subset of parameters which can possibly impose constraints on the system. Each of these heuristics is suitable for a specific application rather than for arbitrary wireless mobile networks.

To be precise, the Highest-Degree heuristic states that the node with the largest number neighbors should be elected as a clusterhead. However, a clusterhead may not be able handle a large number of nodes due to resource limitations even if these nodes are its immediate neighbors and lie well within its transmission range. For example, *Bluetooth* [12] employs a Master-Slave model where the clusterhead is the master and can handle up to seven slaves [21]. Thus, the load handling capacity of the clusterhead puts an upper bound on the node-degree. In other words, simply covering the area with the minimum number of clusterheads will put more burden on the clusterheads. On the other hand, a large number of clusterheads will lead to a computationally expensive system. Although this may result in good throughput, the data packets have to go through multiple hops thus implying high latency.

Similarly, the Lowest-ID heuristic concerns only with the lowest node ids which are arbitrarily assigned numbers without considering the qualifications of a node possibly being elected as a clusterhead. Since the node ids do not change with time, those with smaller ids are more likely to become clusterheads than nodes with larger ids. Thus, certain nodes are prone to power drainage due to serving as clusterheads for longer periods of time.

The Node-Weight heuristic assigns node-weights based on the suitability of nodes acting as clusterheads and the election of the clusterhead is done on the basis of the largest weight among its neighbors. This means that a node decides to become a clusterhead or stay as an ordinary node depending on the weights of its one hop neighbors. Basically, the node has to wait for all the responses from its neighbors to make its own decision to be a clusterhead or an ordinary node. This heuristic does not account for the amount of time that a node may need to wait to receive responses from its neighbors.

3. Weighted Clustering Algorithm (WCA)

In this section, we present our weighted clustering algorithm. We give the design philosophy and the basis of our algorithm before discussing the details.

3.1. Preliminaries

The network formed by the nodes and the links can be represented by an undirected graph $G = (V, E)$, where V represents the set of nodes v_i and E represents the set of links e_i. Note that the cardinality of V remains the same but the cardinality of E always changes with the creation and deletion of links. Clustering can be thought as a graph partitioning problem with some added constraints. As the underlying graph

does not show any regular structure, partitioning the graph optimally (i.e., with minimum number of partitions) with respect to certain parameters becomes an NP-hard problem [7]. More formally, we look for the set of vertices $S \subseteq V(G)$, such that

$$\bigcup_{v \in S} N[v] = V(G).$$

Here, $N[v]$ is the *neighborhood* of node v, defined as

$$N[v] = \bigcup_{v' \in V,\ v' \neq v} \{v' | \operatorname{dist}(v, v') < tx_{\text{range}}\},$$

where tx_{range} is the transmission range of v. The neighborhood of a clusterhead is the set of nodes which lie within its transmission range. The set S is called a *dominating set* such that every vertex of G belongs to S or has a neighbor in S.

The dominating set of the graph is the set of clusterheads. It might be possible that a node is physically nearer to a clusterhead but belongs to another clusterhead. This is because of the other considerations discussed in section 2.4. For example, a node might be physically closer to a clusterhead that is over loaded. In that case it will attach itself to a clusterhead which is far off due to mobility, the nodes may go outside the transmission range of their clusterhead thus changing its neighborhood. However, this does not result in a change of the dominant set. It might so happen that the detached node is not able to attach itself to any of the nodes in the dominant set. This implies that the existing dominant set can no longer cover the entire graph, hence the clustering algorithm has to be invoked to find a new dominant set.

3.2. Design philosophy

Choosing an optimal number of clusterheads which will yield high throughput but incur as low latency as possible, is still an important problem. As the search for better heuristics for this problem continues, we propose a new algorithm which is based on the use of a *combined weight* metric, that takes into account several system parameters like the ideal node-degree, transmission power, mobility and the battery power of the nodes. Depending on specific applications, any or all of these parameters can be used in the metric to elect the clusterheads. We could have a fully distributed system where all the nodes in the mobile network share the same responsibility and act as clusterheads. However, more clusterheads result in extra number of hops for a packet when it gets routed from the source to the destination, since the packet has to go via larger number of clusterheads. Thus this solution leads to higher latency, more power consumption and more information processing per node.

On the other hand, to maximize the resource utilization, we can choose to have the minimum number of clusterheads to cover the whole geographical area over which the nodes are distributed. The whole area can be split up into zones, the size of which can be determined by the transmission range of the nodes. This can put a lower bound on the number of clusterheads required. Ideally, to reach this lower bound, a uniform distribution of the nodes is necessary over the entire area. Also, the total number of nodes per unit area should be restricted so that the clusterhead in a zone can handle all the nodes therein. However, the zone based clustering is not a viable solution due to the following reasons. The clusterheads would typically be centrally located in the zone, and if they move, new clusterheads have to be elected. It might so happen that none of the other nodes in that zone are centrally located. Therefore, to find a new node which can act as a clusterhead with the other nodes within its transmission range might be difficult. Another problem arises due to non-uniform distribution of the nodes over the whole area. If a certain zone becomes densely populated due to migration of nodes from other zones, then the clusterhead might not be able to handle all the traffic generated by the nodes because there is an inherent limitation on the number of nodes a clusterhead can handle. We propose to elect the minimum number of clusterheads which can support all the nodes in the system satisfying the above constraints.

3.3. Basis for our algorithm

To decide how well suited a node is for being a clusterhead, we take into account its degree, transmission power, mobility and battery power. The following features are considered in our clustering algorithm:

- The clusterhead election procedure is not *periodic* and is invoked as rarely as possible. This reduces system updates and hence computation and communication costs. The clustering algorithm is not invoked if the relative distances between the nodes and their clusterheads do not change.
- Each clusterhead can ideally support only δ (a pre-defined threshold) nodes to ensure efficient medium access control (MAC) functioning. If the clusterhead tries to serve more nodes than it is capable of, the system efficiency suffers in the sense that the nodes will incur more delay because they have to wait longer for their turn (as in TDMA) to get their share of the resource. A high system throughput can be achieved by limiting or optimizing the *degree* of each clusterhead.
- The *battery power* can be efficiently used within certain transmission range, i.e., it will take less power for a node to communicate with other nodes if they are within close distance to each other. A clusterhead consumes more battery power than an ordinary node since a clusterhead has extra responsibilities to carry out for its members.
- *Mobility* is an important factor in deciding the clusterheads. In order to avoid frequent clusterhead changes, it is desirable to elect a clusterhead that does not move very quickly. When the clusterhead moves fast, the nodes may be detached from the clusterhead and as a result, a *reaffiliation* occurs. Reaffiliation takes place when one of the ordinary nodes moves out of a cluster and joins another existing cluster. In this case, the amount of information exchange between the node and the corresponding clusterhead is local and relatively small. The information update

in the event of a change in the dominant set is much more than a reaffiliation.

- A clusterhead is able to communicate better with its neighbors having closer *distances* from it within the transmission range [22]. As the nodes move away from the clusterhead, the communication may become difficult due mainly to signal attenuation with increasing distance.

3.4. Proposed algorithm

Based on the preceding discussion, we propose an algorithm called *weighted clustering algorithm* (WCA) that effectively combines each of the above system parameters with certain weighing factors chosen according to the system needs. For example, power control is very important in CDMA networks, thus the weight of the corresponding parameter can be made larger. The flexibility of changing the weight factors helps us apply our algorithm to various networks. The output of clusterhead election procedure is a set of nodes called the *dominant set*. According to our notation, the number of nodes that a clusterhead can handle ideally is δ. This is to ensure that clusterheads are not over-loaded and the efficiency of the system is maintained at the expected level. The clusterhead election procedure is invoked at the time of system activation and also when the current dominant set is unable to cover all the nodes. Every invocation of the election algorithm does not necessarily mean that all the clusterheads in the previous dominant set are replaced with the new ones. If a node detaches itself from its current clusterhead and attaches to another clusterhead then the involved clusterheads update their member list instead of invoking the election algorithm. A preliminary version of this algorithm appeared in [8].

3.4.1. Clusterhead election procedure

The procedure consists of eight steps as described below:

Step 1. Find the neighbors of each node v (i.e., nodes within its transmission range) which defines its *degree*, d_v, as

$$d_v = \big|N(v)\big| = \sum_{v' \in V,\ v' \neq v} \big\{\text{dist}(v, v') < tx_{\text{range}}\big\}.$$

Step 2. Compute the *degree-difference*, $\Delta_v = |d_v - \delta|$, for every node v.

Step 3. For every node, compute the *sum of the distances*, D_v, with all its neighbors, as

$$D_v = \sum_{v' \in N(v)} \big\{\text{dist}(v, v')\big\}.$$

Step 4. Compute the running average of the speed for every node till current time T. This gives a measure of mobility and is denoted by M_v, as

$$M_v = \frac{1}{T} \sum_{t=1}^{T} \sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2},$$

where (X_t, Y_t) and (X_{t-1}, Y_{t-1}) are the coordinates of the node v at time t and $(t-1)$, respectively.

Step 5. Compute the cumulative time, P_v, during which a node v acts as a clusterhead. P_v implies how much battery power has been consumed which is assumed more for a clusterhead than an ordinary node.

Step 6. Calculate the *combined weight* W_v for each node v, where

$$W_v = w_1 \Delta_v + w_2 D_v + w_3 M_v + w_4 P_v,$$

where w_1, w_2, w_3 and w_4 are the *weighing factors* for the corresponding system parameters.

Step 7. Choose that node with the smallest W_v as the clusterhead. All the neighbors of the chosen clusterhead are no longer allowed to participate in the election procedure.

Step 8. Repeat steps 2–7 for the remaining nodes not yet selected as a clusterhead or assigned to a cluster.

The first component, $w_1 \Delta_v$, contributing towards the combined metric W_v helps in efficient MAC functioning because it is always desirable for a clusterhead to handle upto a certain number of nodes in its cluster. The motivation of D_v is mainly related to energy consumption. It is known that more power is required to communicate to a larger distance. As a result, one might think that it would be more appropriate to use the sum of the squares (or higher exponent) of the distances, because the power required to support a link increases considerably faster than linearly with distance (at least in the far-field region). The usual attenuation in the signal strength is inversely proportional to some exponent of the distance, which is usually approximated to 4 in cellular networks where the distance between mobiles and base stations is of the order of 2–3 miles. But in ad hoc networks, the distances involved are rather small (approximately hundreds of meters). In this range, the attenuation can be assumed to be linear [16]. The third component for W_v is due to mobility of the nodes. As discussed in section 3.3, a node with less mobility is always a better choice for a clusterhead. The last component P_v, is measured as the total (cumulative) time a node acts as a clusterhead. We have assumed that the battery power of all nodes to be the same at the beginning. In that case, the battery drainage gives a direct measure of the available battery power. Also, we have taken into consideration that the battery drainage will be more for nodes acting as clusterheads. However, if the nodes have various battery power to start with, then it would be a more accurate metric to measure the power currently available at the node. This will in turn depends on the node's initial power and the power expended based on actual network traffic and length of the links used to support it.

3.4.2. An illustrative example

We demonstrate our weighted clustering algorithm with the help of figures 1–6. All numeric values, as obtained from executing WCA on the 15 nodes as shown in figure 1, are tabulated in table 1. Figure 1 shows the initial configuration of the nodes in the network with individual node ids. Dotted circles with equal radius represent the fixed transmission range for each node. A node can hear broadcast beacons from the nodes which are within its transmission range. An edge between two

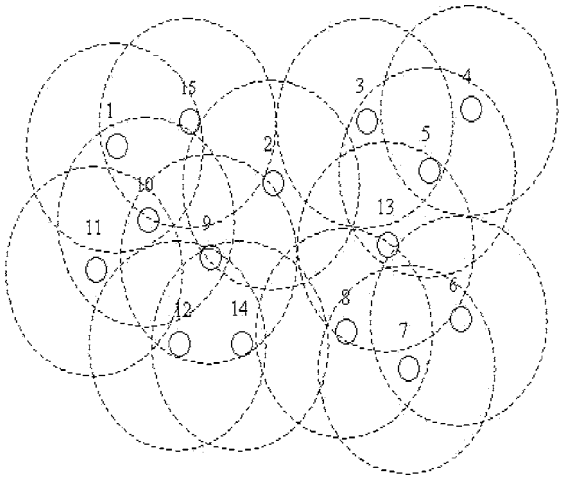

Figure 1. Initial configuration of nodes.

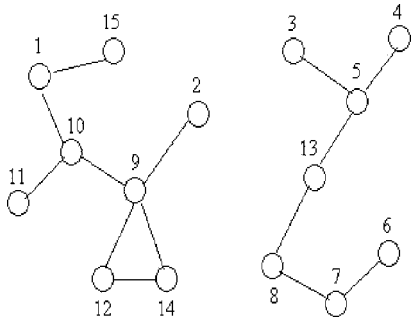

Figure 2. Neighbors identified.

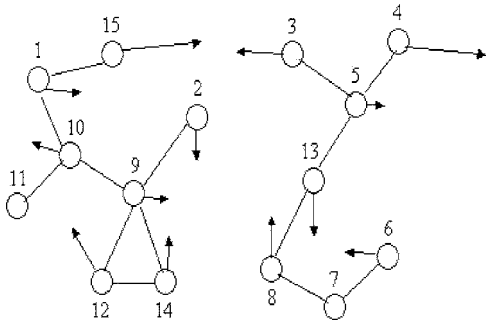

Figure 3. Velocity of the nodes.

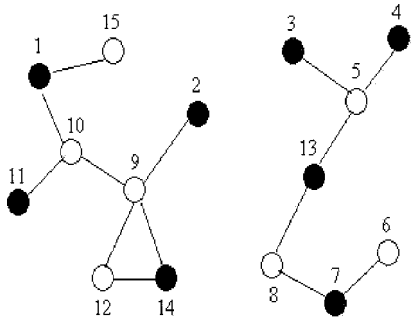

Figure 4. Clusterheads identified.

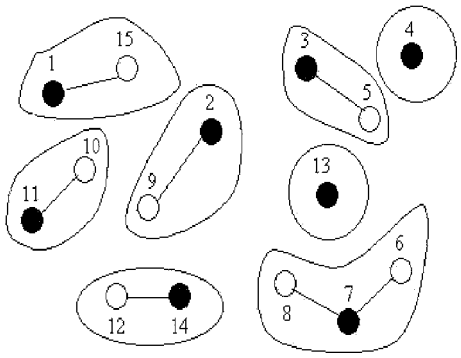

Figure 5. Clusters identified.

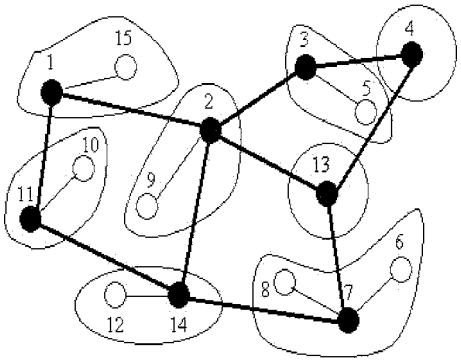

Figure 6. Connectivity achieved.

Table 1
Execution of WCA.

Node id	d_v step 1	Δ_v step 2	D_v step 3	M_v step 4	P_v step 5	W_v step 6
1	2	0	6	2	1	1.35
2	1	1	4	2	2	1.70
3	1	1	3	3	1	1.50
4	1	1	3	4	2	1.60
5	3	1	9	1	4	2.75
6	1	1	3	2	2	1.50
7	2	0	6	0	0	1.20
8	2	0	7	3	3	1.70
9	4	2	13	2	6	4.40
10	3	1	12	2	7	3.55
11	1	1	3	0	1	1.35
12	2	0	5	3	4	1.35
13	2	0	7	3	2	1.65
14	2	0	5	2	0	1.10
15	1	1	3	4	3	1.65

nodes in figure 2 signifies that the nodes are neighbors of each other. The degree, d_v, which is the total number of neighbors a node has is shown in step 1. The degree difference, Δ_v, of each node with ideal node degree $\delta = 2$ is computed in step 2. Sum of the distances, D_v, for each node is calculated as step 3, where the unit distance has been chosen arbitrarily. The arrows in figure 3 represent the speed and direction of movement associated with every node. A longer arrow represents faster movement and a shorter arrow indicates slower movement. The values for M_v (step 4), are chosen randomly. $M_v = 0$ implies that a node does not move at all. We choose

some arbitrary values for P_v which represent the amount of time a node has acted as a clusterhead. This corresponds to step 5 in our algorithm. After the values of all the components are identified, we compute the weighted metric, W_v, for every node as proposed in step 6 in our algorithm. The weights considered are $w_1 = 0.7$, $w_2 = 0.2$, $w_3 = 0.05$ and $w_4 = 0.05$. Note that these weighing factors are chosen arbitrarily such that $w_1 + w_2 + w_3 + w_4 = 1$. The contribution of the individual components can be tuned by choosing the appropriate combination of the weighing factors. Figure 4 shows how a node with minimum W_v is selected as the clusterhead in a distributed fashion as stated in step 7 in our algorithm. The solid nodes represent the clusterheads elected for the network. Note that no two clusterheads are immediate neighbors. Figure 5 shows the initial clusters formed by execution of the clustering algorithm. We observe that the total number of neighbors served by each clusterhead is close to the predefined ideal degree, $\delta = 2$. Figure 6 shows the achieved connectivity in the network. As discussed earlier, the connectivity is accomplished through the higher power (as a result of dual mode power) transmission range of a clusterhead. It can be noted that a single component graph is obtained in this case which means that there is a path from a node to any other node.

3.4.3. Complexity due to distributiveness

The time required for the selection of the node with minimum W_v depends on the implementation of the algorithm. In a centralized system with a central server, the minimum W_v can be found in linear time with respect to the number of nodes. But it is not possible to have a centralized server in ad hoc networks. So, we proceed with a distributed solution in which all the nodes broadcast their ids along with W_v values. A node receives broadcasts from its neighbors and stores the information. This stored information is again exchanged with the immediate neighbors and the process continues till all the nodes become aware of the node with the smallest W_v. The time required for the nodes to gather information about all other nodes will depend on the *diameter* of the underlying graph. It is to be noted that this procedure yields the *global* minima of W_vs unlike the Lowest-ID algorithm which finds only the local minima of ids.

It can be argued that the existing heuristics discussed in section 2 are all special cases of our algorithm. The Highest-Degree heuristic considers only the degree of a node and disregards all other system parameters ($w_2 = w_3 = w_4 = 0$). In Lowest-ID heuristic, the assignment of the ids are random. We can assume that the ids being assigned are based on mobility. The lowest id is assigned to the least mobile node and highest id for the most mobile. In that case, ($w_1 = w_2 = w_4 = 0$). The Node-Weight heuristic simply assigns weights to the nodes which are equivalent to W_v in our case. The basis for suitability of nodes being clusterheads is ignored there. However, in our approach, we define and formulate the parameters for choosing a clusterhead and we show how the weight W_v are calculated.

3.4.4. System activation and update policy

When a system is initially brought up, every node v broadcasts its id which is registered by all other nodes lying within v's transmission range, *tx_range*, as can be seen from figure 1. It is assumed that a node receiving a broadcast from another node can estimate their mutual distance from the strength of the signal received. GPS (Global Positioning System) can be another solution since it is mainly used to obtain the geographical location of nodes. Even though GPS might make the problem relatively simpler, but there is always a cost associated with the deployment of GPS since every mobile node must be a GPS receiver. Based on the received signal strength, every node is made aware of its neighboring nodes and their corresponding distances. Note that these neighboring nodes are only the geographical neighbors and do not necessarily mean neighbors within the same cluster. Once the neighbors list for each node is ready, our clustering algorithm chooses the clusterhead for the first time, as illustrated in figure 4. It can be noted that the mobility factor and the battery power would be the same for all the nodes when the system is initialized. Effectively, W_v will have only two terms Δ_v and D_v contributing to it. Each node maintains its status (i.e., clusterhead or not). A non-clusterhead node knows the cluster it belongs to and the corresponding clusterhead.

Due to the dynamic nature of the system considered, the nodes as well as the clusterheads tend to move in different directions, thus disorganizing the stability of the configured system. So, the system has to be updated from time to time. The update may result in formation of new clusters and possible change of point of attachment of nodes from one clusterhead to another within the existing dominant set. This is called *reaffiliation*. The frequency of update and hence reaffiliation is an important issue. If the system is updated periodically at a high frequency, then the *latest* topology of the system can be used to find the clusterheads which will yield a good dominant set. However, this will lead to high computational cost resulting in the loss of battery power or energy. If the frequency of update is low, there are chances that current topological information will be lost resulting in sessions terminated midway.

All the nodes continuously monitor their signal strength as received from the clusterhead. When the mutual separation between the node and its clusterhead increases, the signal strength decreases. In that case, the mobile has to notify its current clusterhead that it is no longer able to attach itself to that clusterhead. The clusterhead tries to hand-over the node to a neighboring cluster (existing clusterhead in the dominant set). The clusterhead of the reaffiliated node updates its member list. If the node goes into a region not covered by any clusterhead, then the clusterhead election algorithm is invoked and the new dominant set is obtained.

The objective of our clusterhead election algorithm is to minimize the number of changes in dominant set update. Once the neighbors list for all nodes are created, the degree-difference Δ_v is calculated for each node v. Also, D_v is computed for each node by summing up the distances of its neighbors. The mobility M_v is calculated by averaging the speed

of the node. The total amount of time, T_v, it remained as a clusterhead is also calculated. All these parameters are normalized, which means that their values are made to lie in a pre-defined region. The corresponding weights w_1, w_2, w_3 or w_4 are kept fixed for a given system. The weighing factors also give the flexibility of adjusting the effective contribution of each of the parameters in calculating the *combined weight* W_v. For example, in a system where battery power is more important, the weight w_4 associated with T_v can be made larger. Note that the sum of these weighing factors is 1. The node with the minimum total weight, W_v, is elected as a clusterhead. The elected clusterhead and its neighbors are no longer eligible to participate in the remaining part of the election process which continues until every node is found to be either a clusterhead or a neighbor of some clusterhead.

3.5. Load balancing

The load handled by a clusterhead depends on the number of nodes supported by it. A clusterhead, apart from supporting its members with the radio resources, has also to route messages for other nodes belonging to different clusters. Therefore, it is not desirable to have any clusterhead overly loaded while some others are lightly loaded [1]. At the same time, it is difficult to maintain a perfectly load balanced system at all times due to frequent detachment and attachment of the nodes from and to the clusterheads. To quantitatively measure how well balanced the clusterheads are, we introduce a parameter called *load balancing factor* (LBF). As the load of a clusterhead can be represented by the cardinality of its cluster size, the variance of the cardinalities will signify the load distribution. We define the LBF as the inverse of the variance of the cardinality of the clusters. Thus,

$$\text{LBF} = \frac{n_c}{\sum_i (x_i - \mu)^2},$$

where n_c is the number of clusterheads, x_i is the cardinality of cluster i, and $\mu = (N - n_c)/n_c$, (N being the total number of nodes in the system) is the average number of neighbors of a clusterhead. Clearly, a higher value of LBF signifies a better load distribution and it tends to infinity for a perfectly balanced system.

3.6. Connecting the clusters

As a logical extension to clustering, we investigate the connectivity of the nodes which is essential for any routing algorithm. Clustering ensures that the nodes within a cluster are able to communicate among themselves through the clusterheads, each of which acts as the central node of a *star*, as shown in figure 5. But, inter-cluster communication is not possible if the clusters are not connected. For two clusters to communicate with each other, we assume that the clusterheads are capable of operating in *dual* power mode. A clusterhead uses low power to communicate with the members in its transmission range, and high power to communicate with the neighboring clusterheads because of greater range. The links between the clusterheads are shown as solid lines in figure 6.

We define *connectivity* as the probability that a node is reachable from any other node. For a single component graph, any node is reachable from any other node and the connectivity is 1. If the network does not result in a single component graph, then we can say that all the nodes in the largest component can communicate with each other and the connectivity can be the ratio of the cardinality of the largest component to the cardinality of the graph. Thus,

$$\text{connectivity} = \frac{\text{size of largest component}}{N}.$$

The transmission range of a clusterhead can be made large enough by adjusting the power in such a way so as to yield a connected network.

3.7. Routing messages

As our clusterhead connecting technique assures that all nodes are connected with probability almost 1, we can route messages from any node to any other node. Several algorithms have been proposed for routing messages in ad hoc networks [13–15,19,20]. This paper does not propose or deal with any routing algorithm. It can be noted that the number of hops a message makes in a clusterhead based routing [19] depends on the number of clusterheads in the network. It is not recommended to have too few or too many clusterheads in the network. It is our belief that our clustering algorithm will help routing algorithms in terms of the number of hops. at the clusterhead is expected to be minimum due to the load distribution. If a source node A, wishes to establish a connection with a destination node B, then it first needs to discover a route to B. Node A sends a "route discovery" request message containing B's id to its clusterhead. If B is not present in the same cluster as A, then A's clusterhead propagates the request message to its neighboring clusterhead. On receiving the request, the clusterheads can check their member list for B. This query is done in parallel. If B is found, then a positive acknowledgement is sent from B which reaches A via the clusterheads and the route discovery procedure is terminated. If B is not found then the request message is propagated to the two-hop neighbors of A's clusterhead and the process continues till B is found.

The worst-case search time will arise in the case where all the clusterheads are such connected as to form linear graph. Here, searching a cluster has to be done one at a time, eradicating the possibility of parallel search among the clusters. The worst-case time complexity of finding a node will be $O(|clusterheads|)$, where $|clusterheads|$ is the cardinality of the dominant set.

4. Simulation study

We simulate a system of N nodes on a 100×100 grid. The nodes could move in all possible directions with displacement varying uniformly between 0 to a maximum value (*max_disp*), per unit time. To measure the performance of our

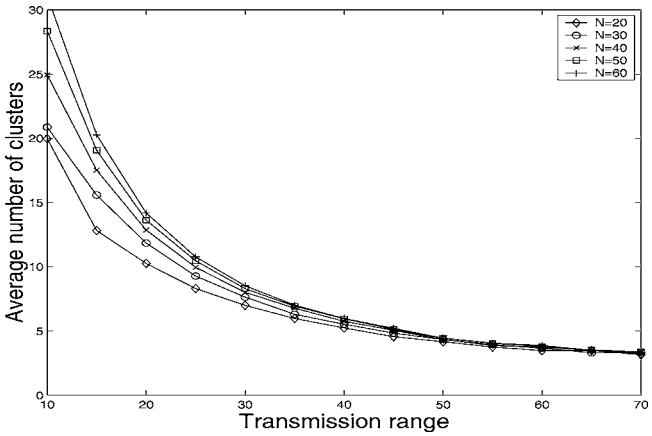

Figure 7. Average number of clusters, *max_disp* = 5.

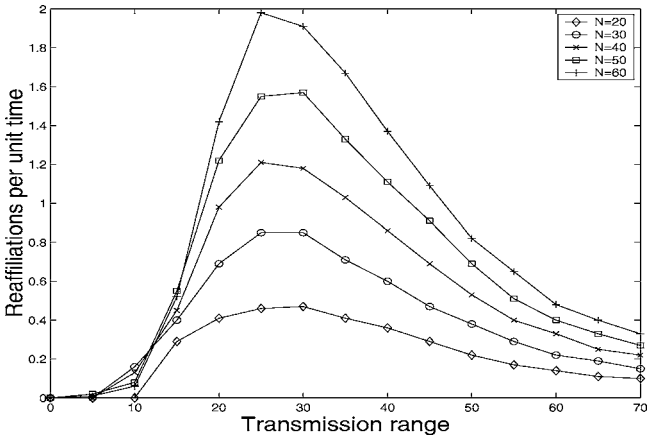

Figure 8. Reaffiliations per unit time, *max_disp* = 5.

algorithm WCA, we identify three metrics: (i) the number of clusterheads, (ii) the number of reaffiliations, and (iii) the number of dominant set updates. Every time a dominant set is identified, its cardinality gives the number of clusterheads. The reaffiliation count is incremented when a node gets dissociated from its clusterhead and becomes a member of another cluster within the current dominant set. The dominant set update takes place when a node can no longer be a neighbor of any of the existing clusterheads. These three parameters are studied for varying number of nodes (N) in the system, transmission range and maximum displacement. We also study how the load balance factor changes as the system evolves and how well connected the nodes are.

In our simulation experiments, N was varied between 20 and 60, and the transmission range was varied between 0 and 70. The nodes moved randomly in all possible directions with a maximum displacement of 10 along each of the coordinates. At every time unit, the nodes move a distance that is uniformly distributed between 0 and *max_disp*. Thus, the maximum Euclidean displacement possible is $10\sqrt{2}$. We assume that each clusterhead can handle at most $\delta = 10$ nodes (ideal degree) in its cluster in terms of resource allocation. Due to the importance of keeping the node degree as close to the ideal as possible, the weight w_1 associated with Δ_v was chosen high. The next higher weight w_2 was given to D_v, which is the sum of distances. Mobility and battery power were given low weights. The values used for simulation were $w_1 = 0.7$, $w_2 = 0.2$, $w_3 = 0.05$ and $w_4 = 0.05$. Note that these values are arbitrary at this time and should be adjusted according to the system requirements.

4.1. Experimental results

Figure 7 shows the variation of the average number of clusterheads with respect to the transmission range where *max_disp* of 5. The results are shown for varying N. We observe that the average number of clusterheads decreases with the increase in the transmission range. This is due to the fact that a clusterhead with a large transmission range will cover a larger area. Figure 8 shows the reaffiliations per unit time. For low transmission range, the nodes in a cluster are relatively close to

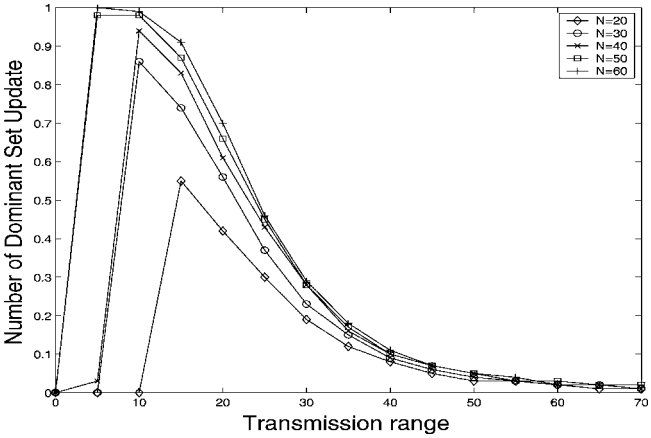

Figure 9. Dominant set updates, *max_disp* = 5.

the clusterhead, and a detachment is unlikely. The number of reaffiliations increases as the transmission range increases, and reaches a peak when transmission range is between 25 and 30. Further increase in the transmission range results in a decrease in the reaffiliations since the nodes, in spite of their random motion, tend to stay inside the large area covered by the clusterhead. Figure 9 shows the number of dominant set updates with respect to the transmission range. For smaller transmission range, the cluster area is small and the probability of a node moving out of its cluster is high. As the transmission range increases, the number of dominant set updates decreases because the nodes stay within their cluster in spite of their movements.

Figures 10–12 show the variation of the same parameters but for varying *max_disp* and constant transmission range of 30. Figure 10 shows that the average number of clusterheads is almost the same for different values of *max_disp*, particularly for larger values of N. This is because, no matter what the mobility is, it simply results in a different configuration but the cluster size remains the same. Figure 11 shows the reaffiliations per unit time with respect to the maximum displacement. As the displacement becomes larger, the nodes tend to move farther from their clusterhead, detaching themselves from the clusterhead and causing more reaffilia-

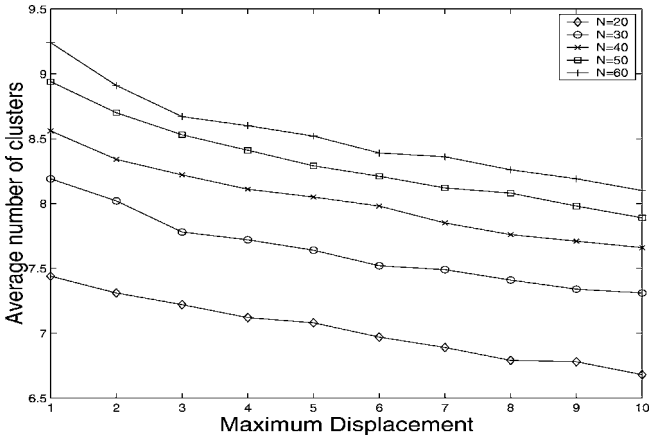

Figure 10. Average number of clusters, *tx_range* = 30.

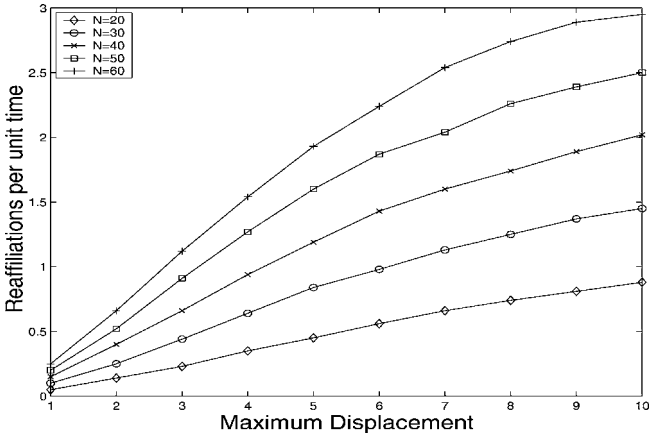

Figure 11. Reaffiliations per unit time, *tx_range* = 30.

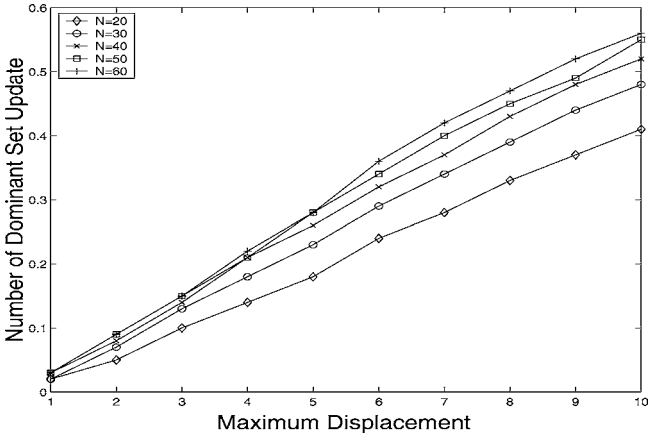

Figure 12. Dominant set updates, *tx_range* = 30.

tions per unit time and more dominant set updates. These are shown in figures 11 and 12, respectively.

The non-periodic invocation of our clustering algorithm can be observed from figure 13 and the reachability of one node from another is shown in figure 14. We observe that after every dominant set update, there is a *gradual* increase in the Load Balance Factor (LBF). This gradual increase in LBF is due to the diffusion of the nodes among clusters. This

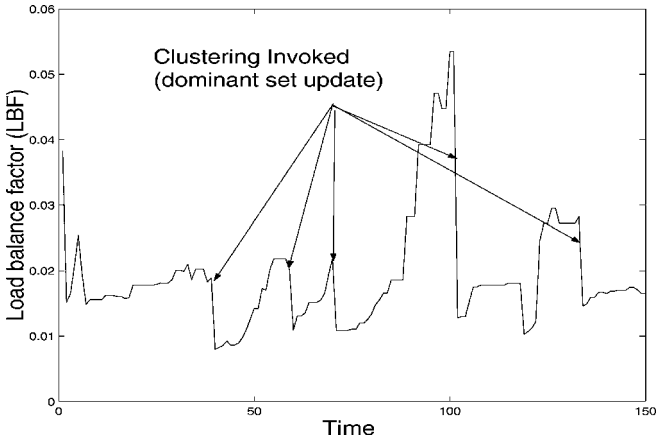

Figure 13. Load distribution.

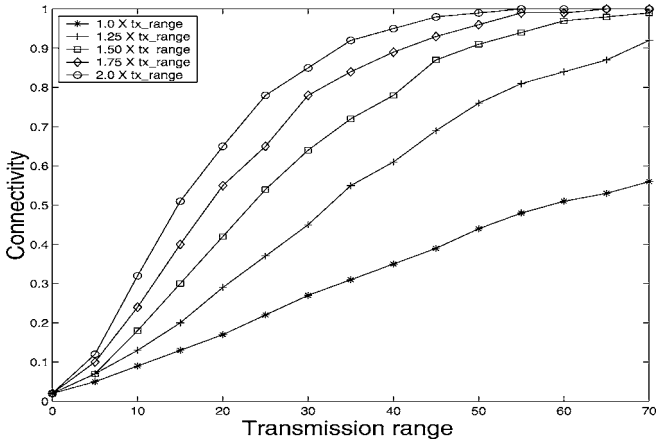

Figure 14. Connectivity.

improvement does not increase indefinitely because the nodes tend to move away from all possible clusterheads and the clustering algorithm has to be invoked to ensure connectivity. The clustering algorithm tries to connect all the nodes at the cost of load imbalance which is represented by the *sharp* decrease in LBF. To study the reachability of one node from another, it is essential that the clusters are connected. For this purpose, the clusterheads operate in dual power modes. As mentioned earlier, the lower power is used to communicate within the cluster whereas the higher power (transmission range) is used to communicate with the neighboring clusterheads. To obtain the higher transmission range, we scaled up the lower transmission range by a constant factor. Simulation was conducted for $N = 50$ and the constant factor was varied from 1.0 to 2.0 with increments of 0.25. Figure 14 demonstrates that a well connected graph can be obtained at the cost of a higher transmission range.

Figure 15 shows the relative performance of the Highest-Degree, Lowest-ID, Node-Weight heuristics and WCA in terms of the number of reaffiliations per unit time vs. transmission range where $N = 30$. The number of reaffiliations for WCA is at most half the number obtained from the Lowest-ID. The main reason is that the frequency of invoking the clustering algorithm is lower in WCA, thus resulting

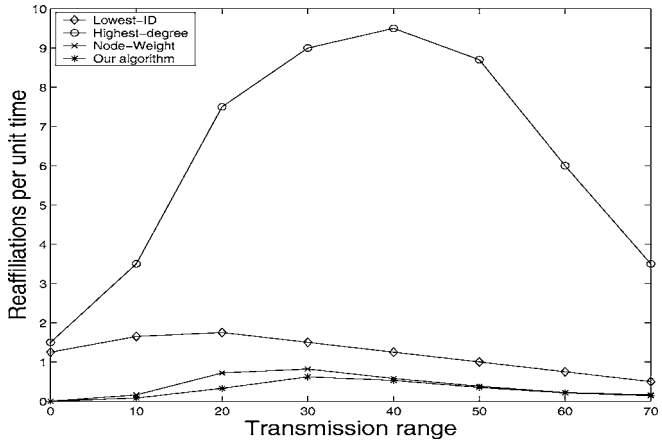

Figure 15. Comparison of reaffiliations, $N = 30$.

in longer duration of stability of the topology. Our algorithm performs marginally better than the Node-Weight heuristics which, however, does not give any basis of assigning the weights to the nodes. Our algorithm WCA describes a linear model which takes into consideration the four important system parameters in deciding the suitability of the nodes acting as clusterheads. It also provides the flexibility of adjusting the weighing factors according to the system needs.

5. Conclusions

We proposed a weight based distributed clustering algorithm (WCA) which can dynamically adapt itself with the ever changing topology of ad hoc networks. Our approach restricts the number of nodes to be catered by a clusterhead so that it does not degrade the MAC functioning. It has also the flexibility of assigning different weights and takes into account a combined effect of the ideal degree, transmission power, mobility and battery power of the nodes. The algorithm is executed only when there is a demand, i.e., when a node is no longer able to attach itself to any of the existing clusterheads. Our clustering algorithm tries to distribute the load as much as possible. We observe that there is a pattern of how the LBF (load balance factor) changes to distribute the load. There is a gradual increase in the LBF due to the diffusion of the nodes among the clusters. The sharp decrease is due to the imbalance caused by the clustering algorithm to ensure that the nodes are connected, which helps in routing messages from any node to any other node. Hence, there is trade-off between the load handled by the clusterheads and the connectivity of the network. We conducted simulation experiments to measure the performance of our clustering algorithm and demonstrate that it performs significantly better than both of the Highest-Degree and the Lowest-ID heuristics. In particular, the number of reaffiliations for WCA is about 50% of that obtained from the Lowest-ID heuristic. Though our approach performs marginally better than the Node-Weight heuristic, it considers more realistic system parameters and has the flexibility of adjusting the weighing factors.

Acknowledgements

The authors are grateful to the anonymous referees and the guest editors for valuable suggestions which improved the quality of the paper.

References

[1] A. Amis and R. Prakash, Load-balancing clusters in wireless ad hoc networks, in: *Proceedings of ASSET 2000*, Richardson, TX, March 2000, pp. 25–32.

[2] D.J. Baker and A. Ephremides, A distributed algorithm for organizing mobile radio telecommunication networks, in: *Proceedings of the 2nd International Conference on Distributed Computer Systems*, April 1981, pp. 476–483.

[3] D.J. Baker and A. Ephremides, The architectural organization of a mobile radio network via a distributed algorithm, IEEE Transactions on Communications COM-29 11 (1981) 1694–1701.

[4] S. Basagni, I. Chlamtac and A. Farago, A generalized clustering algorithm for peer-to-peer networks, in: *Proceedings of Workshop on Algorithmic Aspects of Communication* (satellite workshop of ICALP), July 1997.

[5] S. Basagni, Distributed clustering for ad hoc networks, in: *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks*, June 1999, pp. 310–315.

[6] S. Basagni, Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks, in: *Proceedings of Vehicular Technology Conference, VTC*, Vol. 2, 1999-Fall, pp. 889–893.

[7] B. Bollbas, *Random Graphs* (Academic Press, 1985).

[8] M. Chatterjee, S.K. Das and D. Turgut, An on-demand weighted clustering algorithm (WCA) for ad hoc networks, in: *Proceedings of IEEE GLOBECOM 2000*, San Francisco, November 2000, pp. 1697–1701.

[9] I. Chlamtac and A. Farago, A new approach to the design and analysis of peer-to-peer mobile networks, Wireless Networks 5(3) (August 1999) 149–156.

[10] A. Ephremides, J.E. Wieselthier and D.J. Baker, A design concept for reliable mobile radio networks with frequency hopping signaling, in: *Proceedings of IEEE*, Vol. 75(1) (1987) 56–73.

[11] M. Gerla and J.T.C. Tsai, Multicluster, mobile, multimedia radio network, Wireless Networks 1(3) (1995) 255–265.

[12] http://www.bluetooth.com

[13] M. Joa-Ng and I.-T. Lu, A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks, IEEE Journal on Selected Areas in Communications (August 1999) 1415–1425.

[14] D.B. Johnson, Routing in ad hoc networks of mobile hosts, in: *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications* (December 1994) pp. 158–163.

[15] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad hoc wireless networks, *Mobile Computing*, eds. T. Imielinski and H. Korth (Kluwer Academic Publishers, 1996) ch. 5, pp. 153–181.

[16] W.C.Y. Lee, *Mobile Cellular Telecommunications* (McGraw Hill, 1995).

[17] C.-H.R. Lin and M. Gerla, A distributed control scheme in multi-hop packet radio networks for voice/data traffic support, in: *Proceedings of IEEE GLOBECOM* (1995) pp. 1238–1242.

[18] C.-H.R. Lin and M. Gerla, A distributed architecture for multimedia in dynamic wireless networks, in: *Proceedings of IEEE GLOBECOM* (1995) pp. 1468–1472.

[19] A.B. McDonald and T.F. Znati, A mobility-based framework for adaptive clustering in wireless ad hoc networks, IEEE Journal on Selected Areas in Communications 17(8) (1999) 1466–1487.

[20] A.K. Parekh, Selecting routers in ad-hoc wireless networks, in: *Proceedings of the SBT/IEEE International Telecommunications Symposium*, August 1994.

[21] L. Ramachandran, M. Kapoor, A. Sarkar and A. Aggarwal, Clustering algorithms for wireless ad hoc networks, in: *Proceedings of Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Boston, August 2000, pp. 54–63.

[22] E.H.-K. Wu, J.T.-C. Tsai and M. Gerla, The effect of radio propagation on multimedia, mobile, multihop networks: models and countermeasures, in: *Proceedings of IEEE Singapore International Conference on Networks*, SICON'97.

Mainak Chatterjee received his B.Sc. degree in physics (Hons) from the University of Calcutta in 1994. In 1998, he finished his M.E. in electrical communication engineering at the Indian Institute of Science, Bangalore. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at the University of Texas at Arlington. He has been recognized as a university scholar and has been included in the WHO'S WHO among students in American Universities and Colleges in 2001. His research interests include mobile computing, MAC layer protocols, CDMA data networking, multimedia communications and ad hoc networks. He has published over a dozen research papers and is currently working on several industry funded projects. He is a recipient of the TxTEC (Texas Telecommunications Engineering Consortium) fellowship and is a IEEE-CS student member.
E-mail: chat@cse.uta.edu

Sajal K. Das is currently a Full Professor of Computer Science and Engineering and the founding Director of the Center for Research in Wireless Mobility and Networking (CReWMaN) at the University of Texas at Arlington (UTA). During 1988–1999, he was a Professor of Computer Science at the University of North Texas, where he twice (1991 and 1997) received the Student Association Honor Professor Award for best teaching and outstanding research, as well as the Developing Scholar's award in 1996 for research excellence. He also received the Oustanding Senior Faculty Research Award in computer science and engineering at UTA. His current interests include mobile computing, wireless networks, resource and mobility management, QoS provisioning in wireless multimedia, mobile Internet, and distributed computing. He has published over 200 research papers in these areas and directed numerous funded projects, and holds four US patents in wireless Internet and data networking. He is a recipient of the Best Paper Awards in ACM MobiCom'99, ACM MSWiM-2000, and ACM/IEEE PADS'97. Dr. Das is an editor of four journals including Computer Networks, and the Subject Area Editor of mobile computing for the Journal of Parallel and Distributed Computing. He has guest-edited special issues of WINET, JPDC, IEEE PCS, and IEEE Transactions on Computers. He served as General Chair of WoWMoM-2000 and 2001, WNMC-2001 and MASCOTS'98 and 2002; General Vice-Chair of MobiCom 2000, HiPC 2000 and 2001; Founder of WoWMoM; TPC Chair of WoWMoM'98 and WoWMoM'99; Program Vice-Chair of HiPC'99; and TPC member of numerous conferences including INFOCOM, MobiCom, ICPP and IPDPS. He currently serves on IEEE TCPP Executive Committee.
E-mail: das@cse.uta.edu

Damla Turgut received both her B.S. and M.S. degrees in computer science and engineering from the University of Texas at Arlington (UTA) in 1994 and 1996, respectively. Currently, she is a Ph.D. candidate and an Assistant Instructor in the Department of Computer Science and Engineering at UTA. She coauthored a book "Introduction to Computer Science and Programming with C". She is a member of the Center for Research in Wireless Mobility and Networking (CReWMaN). Her current research interests include mobile ad hoc networks, wireless networks, mobile computing, mobile and object-oriented databases. She has published more than a dozen research papers in these areas. During 1997–1998, she was a project leader in Computer Based Training program at the Center for Advanced Engineering and Systems Automation Research (CAESAR). She served as an officer in a various student organizations. During 1998–1999, she received an outstanding Graduate Student Recognition Committee Chair and Meritorius Service awards from Graduate Student Council at UTA. She has been the recipient of the Texas Telecommunication Engineering Consortium (TxTEC) fellowship since 1999. She is a member of Upsilon Pi Epsilon (UPE) Honor Society for Computing Sciences.
E-mail: turgut@cse.uta.edu