(Submitted for its possible publication in the 2002 International Computer Symposium (ICS2002)

Workshop on Computer Networks)

Minimizing Both the Number of Clusters and the Variation of Cluster Sizes for Mobile Ad Hoc Networks*

Pi-Rong Sheu and Chia-Wei Wang

Department of Electrical Engineering, National Yunlin University of Science & Technology
Touliu, Yunlin 640, Taiwan, R.O.C.
Email: sheupr@pine.yuntech.edu.tw

Abstract

The research of mobile ad hoc networks has attracted a lot of attentions recently. In particular, extensive research efforts have been devoted to the design of clustering strategies to divide all nodes into a clustering architecture such that the transmission overhead for the update of routing tables after topological changes can be reduced. The performance of a clustering architecture has been demonstrated to be closely related to the number of its clusters. In the paper we will address the problem of reducing the number of clusters in a clustering architecture. Since the major power of a mobile ad hoc network is battery power, a cluster head may exhaust its battery power and becomes inactive. This will lead to the cluster head's cluster becoming broken up. Basically, the power consumption rate of a cluster head is proportional to its node degree. Thus, reducing the node degree of a cluster head will extend its lifetime and improve its cluster's stability. The second objective in this paper is to make the number of nodes in each cluster to be equal as much as possible. To sum up, in this paper, we will propose an efficient clustering algorithm to minimize the number of clusters and the variation of cluster sizes. Computer simulations show that both the number of clusters and the variation of cluster sizes generated by our clustering algorithm are less than those generated by other existing clustering algorithms that also aim to minimize the number of clusters.

Keywords: Ad Hoc Network, Clustering Algorithm, Clustering Architecture, Distributed Network Algorithms, Wireless Mobile Networks.

Preferred Subject Areas: Ad-Hoc Network, Wireless Communications

*This work was supported by the National Science Council of the Republic of China under Grant # NSC 90-2213-E-224-025

Correspondence Address:

Dr. Pi-Rong Sheu
Department of Electrical Engineering
National Yunlin University of Science & Technology
Touliu, Yunlin 640, Taiwan, R.O.C.
Tel: 886-5-5342601-4261
Fax: 886-5-5312065
Email: sheupr@pine.yuntech.edu.tw

1. Introduction

A mobile ad-hoc network (MANET) is formed by a group of mobile hosts (or called mobile nodes) without an infrastructure consisting of a set of fixed base stations. A mobile host in a MANET can act as both a general host and a router; i.e., it can generate as well as forward packets. Two mobile hosts in such a network can communicate directly with each other through a single-hop route in the shared wireless media if their positions are close enough. Otherwise, they need a multi-hop route to finish their communications. In a multi-hop route, the packets sent by a source are relayed by multiple intermediate hosts before reaching their destination. MANETs are found in applications such as short-term activities, battlefield communications, disaster relief situations, and so on. Undoubtedly, MANETs play a critical role in situations where a wired infrastructure is neither available nor easy to install.

The research of MANETs has attracted a lot of attentions recently. In particular, since host mobility causes frequent unpredictable topological changes, the task of finding and maintaining routes in MANETs is nontrivial. Therefore, extensive research efforts have been devoted to the design of clustering strategies to divide all nodes into a clustering architecture such that the transmission overhead for the update of routing tables after topological changes can be reduced [10] [11] [13] [14]. In fact, the researches have demonstrated that routing on top of clustered topologies is much more scalable than flat routing [10] [11] [13] [14]. In addition, a clustering architecture can facilitate the spatial reuse of resources to increase the system capacity [8] [12]. For example, under a non-overlapping clustering architecture, two clusters may use the same frequency or code set if they are not neighboring clusters to each other. Furthermore, in a clustering architecture, when a mobile node changes its position, it is sufficient for only the nodes in its clusters to update their topology information, not all in this system.

In appearance, a clustering architecture is similar to a single-hop cellular architecture [8] [12]. Figure 1 shows a clustering architecture for a MANET. There exists a link between two nodes if the two nodes are in the transmission

range of each other. The black nodes are the cluster heads of the clustering architecture. Nodes within a circle belong to the same cluster. Each node in a MANET is assigned a unique identifier (ID) that is a positive integer. We assume a cluster's identifier is the same as its cluster head's node identifier. For example, the identifier of the cluster with node 15 as its cluster head is 15. Nodes 3, 8, 13, 15, and 16 all belong to cluster 15. A cluster head in each cluster acts as a coordinator to resolve channel assignment, perform power control, maintain time division frame synchronization, and enhance the spatial reuse of bandwidth. The major characteristics of a clustering architecture are as follows. Firstly, there is only one cluster head in each cluster. Secondly, each node in a clustering architecture is either a cluster head or adjacent to one or more cluster heads. A node belonging to two or more clusters is called a gateway node (or a border node). Thirdly, any two cluster heads are not adjacent to each other. Fourthly, any two nodes in the same cluster are at most two hops away from each other.

The performance of a clustering architecture has been demonstrated to be closely related to the number of its clusters and gateway nodes [7]. This is because the overhead of broadcasting task, where packets initiated at a source is retransmitted by only cluster heads and gateway nodes, can be significantly reduced when the number of clusters and border nodes is decreased. Therefore, in the paper we will address the problem of reducing the number of clusters in a clustering architecture for MANETs.

Since the major power of a MANET is battery power, a node may become inactive because of the exhaustion of its battery power. The inaction of node may lead to communication between two nodes far away becomes broken. Similarly, when a cluster head exhausts its battery power and becomes inactive, the cluster which it belongs to will be broken up. Thus, it becomes a significant work to provide a stable clustering architecture for MANETs. Recall that a cluster head plays a role as a coordinator in its cluster. Hence, a cluster head will consume more battery power than an ordinary node. In fact, it is feasible to assume that the power consumption rate of a cluster head is proportional to its

node degree. Thus, reducing the node degree of a cluster head will extend its lifetime and improve its cluster's stability. Therefore, one of the ways to provide a stable clustering architecture is to make the number of nodes in each cluster to be equal as much as possible. This is the second objective in this paper.

To sum up, in this paper, we will propose an efficient clustering algorithm whose objective is to minimize the number of clusters and the variation of cluster sizes. A small variation of cluster sizes implies that the difference between the numbers of nodes in the largest cluster and in the smallest cluster is small. Besides, our clustering algorithm can preserve its structure as much as possible when nodes are moving and/or the topology is slowly changing. Computer simulations show that both the number of clusters and the variation of cluster sizes generated by our clustering algorithm are less than those generated by other existing clustering algorithms that also aim to minimize the number of clusters.

The rest of this paper is organized as follows. In Section 2, we address backgrounds and related researches. In Section 3, our proposed clustering algorithm is presented. In Section 4, the performance of our clustering algorithm is evaluated by computer simulations. Finally, in Section 5, we make some conclusions.

2. Background and related researches

In this section, the background and related researches will be addressed.

2.1 Background

Generally speaking, a clustering architecture can be classified into two different kinds of types: overlapping and non-overlapping. In an overlapping clustering architecture [2] [3] [4][5] [7] [8], a node that is not a cluster head may belong to more than one cluster and such a node is named a gateway node. Communication between any two adjacent clusters has to rely on their common gateway nodes. A node belonging to only one cluster is called an ordinary node.

For example, in Figure 1, nodes 8, 12 and 13 are gateway nodes and the other white nodes are ordinary nodes. On the other hand, in a non-overlapping clustering architecture [9] [15], each node belongs to only one cluster and if it is not a cluster head, then it is named an ordinary node.

Now let us observe the clustering architecture shown in Figure 1. There are no gateway nodes between cluster 2 and cluster 6. To overcome the problem, the concept of distributed gateway (DG) is proposed in [8]. A DG is a pair of ordinary nodes that belong to different clusters but there exists a link between them. For example, the pair of node 9 and node 10 may form a DG. One of the advantages introduced by using the DG is that the hop-counts of a route may be reduced. Observe Figure 1 and consider the route from node 7 to node 1. The route will be {7, 6, 12, 5, 13, 15, 8, 2, 1} if the DG is not introduced. However, the route can be changed to become {7, 6, 10, 9, 2, 1} if the technique of the DG is adopted. This is because if the pair of node 9 and node 10 is a DG, then the two nodes can transmit packets directly to each other. It is shown in [8] that the DG technique can be extremely effective when connectivity is weak. Furthermore, the technique can be applied to both the overlapping and non-overlapping clustering architecture. For simplification, we will adopt the non-overlapping architecture and the DG technique in the following discussion.

2.2 Related researches

In the following, some related clustering algorithms will be reviewed. Especially, we will focus on those clustering algorithms whose objective is to minimize the number of clusters.

The lowest-ID clustering algorithm and the highest-connectivity clustering algorithm proposed in [8]

To provide a convenient framework for the development of important features such as code separation, channel access, routing, power control, virtual circuit support and bandwidth allocation, two well-known clustering algorithms: the lowest-ID clustering algorithm and the highest-connectivity clustering algorithm have been proposed in [8].

The lowest-ID clustering algorithm:

Each node is assigned a distinct identifier (ID). Periodically, each node broadcasts its own ID to its neighbors.

- A node has ID lower than its neighbors is a cluster head.
- The lowest-ID neighbor of a node is its cluster head.
- A node which can hear more than one cluster head is a gateway node.
- Otherwise, a node is an ordinary node.

The highest-connectivity clustering algorithm:

Each node is assigned a distinct identifier (ID). Periodically, each node broadcasts the list of nodes that it can hear.

- A node that has not elected its cluster head yet is an "uncovered" node; otherwise, it is a "covered" node.
- A node is elected as a cluster head if it is the most highly connected node of its "uncovered" neighbors (if there is a tie, the node with lowest ID prevails).
- A node that has already elected another node as its cluster head gives up its role as a cluster head.

Both the two clustering algorithms divide the entire MANET into clusters and there is a cluster head elected for each cluster. The cluster head election criteria are ID-based and degree-based, respectively. The feature of their clustering architectures belongs to overlapping.

The connectivity based k-hop clustering algorithm proposed in [7]

To minimize the number of clusters and gateway nodes in a k-hop clustering architecture, a connectivity based *k*-hop clustering algorithm has proposed in [7]. In fact, when $k = 1$, the algorithm is similar to the highest-connectivity clustering algorithm above. That is, in both the algorithms, the node degree is adopted as the primary criterion in electing a cluster head while the node identifier is the secondary criterion. To be more specific, a node with a larger node degree will have a larger probability to become a cluster head. If a tie happens, the node with the largest ID will be the winner. In [7], this idea is applied to a k-hop clustering architecture and the node degree of a node is re-defined

as the number of the node's k-hop neighbors. Each node initiates the clustering by flooding a request for clustering to all the other nodes. A node whose priority (based on the above two criteria) is the highest among its k-hop neighbors will broadcast its declaration of being the cluster head to all its k-hop neighbors. The simulation results in [7] demonstrate that the connectivity based k-hop clustering algorithm can indeed reduce the numbers of both clusters and gateway nodes in a k-hop clustering architecture.

In the cluster maintenance procedure proposed in [7], when a node switches on, it joins the cluster whose cluster head is located at the node's k-hop neighborhood. If there exists no proper cluster, the node will form a new one. If a cluster head switches off, the nodes in the cluster will elect a new cluster head by using the number of k-hop neighbors within the cluster as the main criterion. The number of overall k-hop neighbors within the whole MANET is the secondary criterion and node ID is adopted as the third criterion. If an existing link is disconnected and this causes some nodes to have hop counts from the cluster head greater than k, then these nodes will create their own new cluster(s).

The efficient k-hop clustering scheme proposed in [15]

In order to support an efficient k-hop clustering routing, an efficient k-hop clustering (EKC) algorithm for selecting suitable cluster heads has proposed in [15]. The goal of EKC is to form less clusters and more stable clusters. To minimize the number of clusters, EKC tries to avoid as much as possible that nodes whose degrees are one are elected as clusterheads. In EKC, a label is assigned to each node. A node's label indicates the node's status. Label B indicates the node is not clustered. Label R indicates the node is a cluster head. Label F indicates the node is clustered but not a cluster head. Label B is the initial status. The following notations are defined and used in EKC. V represents the set of all nodes. $N_k(v)$ represents the set of nodes that are less than or equal to k hops away from node v, except

node *v*. $N_k[v]$ is the set of nodes that are less than or equal to k hops away from node *v*, including node *v*. $d(u,v)$ is defined to be the least hops from node *u* to node *v*. $\deg(v)$ is defined to be the degree of node *v*. $L_v = \{x \mid x \in N_k(v), \deg(x) = 1\}$. $B_v^k = \{x \mid x \in N_k[v], label(x) = B\}$. $F_v^k = \{x \mid x \in N_k[v], label(x) = F\}$.

The primary criterion $r_1(v)$ of selecting a cluster head is calculated as $r_1(v) = \sum_{u \in L_v} d(u,v)$ if $L_v \neq \phi$, $r_1(v) = 0$, otherwise. Roughly speaking, a higher $r_1(v)$ implies that node *v* dominates more nodes of degree one. The secondary criterion $r_2(v)$ is calculated as $r_2(v) = |B_v^k|$. It represents the number of non-clustered k-hop neighbors of node *v*. After exchanging packets, a node whose criterion is the largest among its k-hop neighbors will broadcast its claim on becoming the cluster head to all its k-hop neighbors.

In the cluster maintenance procedure presented in [15], when a new node switches on within an existing clustering architecture, it selects the neighbor most closed to its clusterhead and joins the cluster which the neighbor belongs to. A link failure may occur through node switching off or node movement. When a link failure happens between two clusters, their individual cluster information has to be updated. If a link failure happens within a cluster, a node will still belong to the original cluster provided that it can still be k-hop dominated by its cluster head. Otherwise, the node will find another new cluster head as at the initial state.

3. Our proposed clustering algorithm

3.1 Our cluster head election criteria

Basically, the problem of finding a clustering architecture with the minimum number of clusters is equivalent to the so-called *minimum independent dominating set problem* in graph theory [6]. Two of the main approaches to solve the problem are as follows. One is to try to avoid that nodes whose degrees are too low are elected as cluster heads [1]. The other is to let nodes with high node degrees become cluster headers as much as possible. Obviously, the

highest-connectivity clustering algorithm adopts the second approach while the efficient k-hop clustering algorithm seems to use the first approach. As will be shown in the following, neither of them can yield a clustering architecture with the minimum number of clusters in many cases.

The main idea behind our clustering algorithm is to cleverly combine the above two approaches to form a new cluster head election scheme. To be more specific, we will use the first approach as our first criterion in electing a cluster head. Then, if a tie occurs, then the second approach will be involved. The computer simulations given in Section 4 have demonstrated that our idea is efficient in reducing the number of clusters and the variation of cluster sizes.

First, let us define a node whose degree is k as a degree-k node; e.g. a node whose degree is one is called a degree-1 node, a node whose degree is two is called a degree-2 node, and so on. Next, we will define and explain the most important packet, *CRITERION*(*weighted_value*, *nc_degree*, *id*), used in our proposed clustering algorithm. The packet has three parameters. We use the first parameter, *weighted_value*, to attempt to keep nodes with too low degrees from becoming cluster heads. Thus, we might keep too many small clusters from being formed. This will naturally decrease the number of clusters in the whole MANET. The intention of the second parameter is to reduce the number of clusters by means of electing nodes with higher degrees as cluster heads. Because many nodes may have the same *weighted_value* and *nc_degree*, ties may happen frequently between nodes if the election of cluster heads only depends on the two parameters. Therefore, we need to introduce the third parameter, node identifier. When a tie occurs, the node with the highest identifier will become the cluster head.

The parameter *weighted_value* can be calculated as follows. When we attempt to keep any node with degree $\leq D_{avoid}$ from being elected as a cluster head, where D_{avoid} is a predefined integer, each degree-n node whose

$n \le D_{avoid}$ will broadcast a *DEGREE_n*(*id*) packet, where *id* is its identifier, to all its neighboring nodes. Then, each time when a node receives a non-duplicated *DEGREE_n*(*id*) packet, the node will add $\frac{D_{avoid}}{n}$ to its *weighted_value*, which is initially set to zero, For example, if we try to avoid that any node with degree less than or equal to 3 will be elected as a cluster head, then each of the degree-1 nodes (degree-2 nodes, degree-3 nodes) will broadcast a *DEGREE_1*(*id*) (*DEGREE_2*(*id*), *DEGREE_3*(*id*)) packet to its neighboring nodes. Then, each time when a node receives a non-duplicated a *DEGREE_1*(*id*) (*DEGREE_2*(*id*), *DEGREE_3*(*id*)) packet, the node will add 3 (1.5, 1) to its *weighted_value*. It is not hard to observe that the performance of our clustering algorithm is related to the value of D_{avoid}. In Section 4, we will study, by means of computer simulations, the influence of D_{avoid} on the performance of our clustering algorithm and determine the suitable values of D_{avoid} for different network sizes.

The parameter *nc_degree* is defined to be the non-clustered degree of the sending node, i.e., the number of non-clustered neighboring nodes of the sending node. Recall that each node can know its own degree through receiving beacons from its neighboring nodes. If each node is required to append its own cluster identifier to its beacons, then any node can calculate the value of its *nc_degree* parameter since its *nc_degree* is equal to the number of its neighboring nodes with cluster identifier being zero. Finally, the parameter *id* is the sending node's identifier.

In the process of electing a cluster head, we will first elect the node with the largest *weighted_value* as the cluster head. If there is a tie, then the second parameter will be involved in the election. The node with the largest *nc_degree* will become the cluster head. Finally, if there is still a tie, then the node with the highest identifier prevails.

3.2 Our clustering algorithm

Before describing our clustering algorithm in detail, we make the following assumptions that are common in designing clustering algorithms for MANETs [2][8][12]:

1. The network topology is static during the execution of the clustering algorithm.

2. A packet broadcasted by a node can be received correctly by all its one-hop neighbors within a finite time.

3. Each node has a unique ID and knows its degree (the number of its one-hop neighbors). At the same time, each node knows the ID and degree of its every one-hop neighbor.

In addition to the *CRITERION*(*weighted_value*, *nc_degree*, *id*) packet, which have already been defined in the above, we need to define two new packets used in our clustering algorithm. The *CH*(*cid*, *nc_degree*) packet is used by a node to declare itself as a cluster head. Parameter *cid* is a node's cluster identifier and is initially zero. The *JOIN*(*cid*, *id*) packet is used by a node to inform the cluster head which it wants to join, where *cid* is the identifier of the cluster which it wants to join and *id* is its own node ID. Besides, each cluster head has a *Member_Table* to record its cluster members. Each node has an *NC_Neighbor_Table* to record its non-clustered neighbors. Each time a beacon packet with *cid* equal to zero is received, a node will update its *NC_Neighbor_Table*.

Our clustering algorithm:

- Each node broadcasts its own a *CRITERION*(*weighted_value*, *nc_degree*, *id*) packet to all its neighboring nodes and receives multiple *CRITERION*(*weighted_value*, *nc_degree*, *id*) packets from its neighbors.
- If a node discover that it has a larger *weighted_value* than all its neighbors, then it sets its *cid* to its own node ID and broadcasts a *CH*(*cid*, *nc_degree*) packet to declare itself as a cluster head. If a tie occurs, then the cluster head will the one with the highest *nc_degree*. If there is still a tie, then the node with the highest ID will be the final cluster head.
- When a node receives multiple *CH*(*cid*, *nc_degree*) packets and it dose not belong to any cluster (i.e., its *cid* = 0), it will elect the cluster head with the lowest *nc_degree* (to reduce the variation of cluster sizes) (in

case of a tie, the cluster head with the highest *cid* prevails) and set it own *cid* to the *cid* of the elected cluster head. Then the node broadcasts a *JOIN*(*cid*, *id*) packet to join the cluster.

- When a cluster head receives a *JOIN*(*cid*, *id*) packet with *cid* equal to its own *cid*, it will record the *id* in the received *JOIN*(*cid*, *id*) packet in its *Member_Table*.
- When a non-clustered node receives a *JOIN*(*cid*, *id*) packet, it will remove the node with *id* equal to the *id* of the received *JOIN*(*cid*, *id*) packet from its *NC_Neighbor_Table*.
- Each time a non-clustered node removes a node from its *NC_Neighbor_Table*, it will check whether its *NC_Neighbor_Table* becomes empty or not. If empty, it sets its *cid* to its own node *id* and broadcasts a *CH*(*cid*, *nc_degree*) packet to declare itself as an orphan cluster.
- A node will terminate the clustering algorithm when it has joined or formed a cluster (i.e., its $cid \neq 0$).

Now, let us use the MANET shown in Figure 2 as an example to illustrate the operation of our clustering algorithm. We set D_{avoid} to two in this example. Thus, the *weighted_value* of node 4 is three, the *weighted_value* of node 11 is two, the *weighted_values* of node 1, 2, and 7 are one, and those of the other nodes are zero. Since node 4 has the largest *weighted_value* among nodes 3, 5 and 9, it sets its *cid* to 4 and broadcasts a *CH*(4, 3) packet to declare itself as a cluster head. Similarly, node 11 sets its *cid* to 11 and broadcasts a *CH*(11, 4) packet to declare itself as a cluster head.

Because node 1 and node 2 have the same *weighted_value* and *nc_degree*, the third parameter *id* is used to solve the tie. Node 2 is the winner. Node 2 sets its *cid* to 2 and broadcasts a *CH*(2, 4) packet to declare itself as a cluster head. When receiving multiple *CH*(*cid*, *nc_degree*) packets, each non-clusterhead node has to select a proper cluster to join. Since cluster head 4 has a lower *nc_degree* than cluster head 11, node 3 sets its *cid* to 4 and broadcasts a *JOIN*(4, 3) packet to join cluster 4. Similarly, node 9 joins cluster 4 rather than cluster 2 for the same reason. On the other hand,

although cluster head 11 and cluster head 2 have the same *nc_degree*, node 6 will join cluster 11 because cluster 11 has a higher *cid* than cluster 2. The final clustering architecture constructed by our clustering algorithm is shown in Figure 3, where three clusters are formed.

Figure 4 shows the resulted clustering architecture when we apply the highest-connectivity clustering algorithm in [8] to the MANET in Figure 2. As expected, this one is also the resulted clustering architecture generated by the connectivity based k-hop clustering algorithm in [7] when the parameter k is one. We can observe that both of the algorithms yield four clusters. When the MANET given in Figure 2 executes the efficient k-hop clustering algorithm in [15], the produced clustering architecture is shown in Figure 5, where there are also four clusters and two of them are orphan clusters. In this example, since our clustering algorithm avoids that two degree-2 nodes, node 8 and node 10, become cluster heads, the number of our clusters is less.

Note that when we set D_{avoid} to one, our clustering algorithm is similar to the efficient k-hop clustering scheme [15] if its parameter k is equal to one. Therefore, our clustering algorithm looks like a general case of the efficient k-hop clustering scheme in a one-hop clustering architecture [15].

3.3 Our cluster maintenance procedure

Due to the mobility of MANETs and the power-off or malfunctions of some nodes, the topologies of the clusters are easy to be destroyed. Therefore, after a clustering architecture is constructed by our clustering algorithm, the clustering architecture needs to be updated frequently. A straightforward method to update the clustering architecture is to execute our clustering algorithm periodically. However, when power consumption only makes few cluster heads or ordinary nodes inactive and node mobility only cause some changes in few clusters in a low or middle mobility environment, bandwidth overhead and battery power consumption may be very huge if each node is always required to periodically execute the clustering algorithm to do a global update. To avoid the drawback, we will propose a cluster

maintenance procedure that will only perform a local update as much as possible. The cluster maintenance procedure will be executed by any node that detects a link activation event or a link failure event.

First, we introduce two new packets used in our cluster maintenance procedure: the *DISCONNECT*(*id*) packet and the *INVITE*(*cid*) packet. A *DISCONNECT*(*id*) packet is broadcasted by a node when its user is prepared to turn off the node. An *INVITE*(*cid*) packet is used by a cluster head to invite a detected non-clustered node to join its cluster.

Next, we separately describe the cluster maintenance procedure for link activation events and for link failure events.

The cluster maintenance procedure for link activation events:

- When a link activation occurs between a cluster head and a non-clustered node,
  - The associated cluster head sends an *INVITE*(*cid*) packet to the node located at the other end of the detected active link.
  - If a non-clustered node receives multiple *INVITE*(*cid*) packets, then it joins the cluster with the highest *cid*.
- When a link activation occurs between two cluster heads,
  - Each of the associated cluster heads sends an *INVITE*(*cid*) packet to the other cluster head located at the other end of the detected active link.
  - If an associated cluster head receives at least one *INVITE*(*cid*) packet with higher *cid* than itself, then it joins the cluster with the highest *cid* by sending a *JOIN*(*cid*, *id*) packet to the winning cluster head.
  - If an ordinary node receives a *JOIN*(*cid*, *id*) packet whose *id* is equal to the ordinary node's *cid*, then it sets its *cid* to zero and becomes a non-clustered node.

If no link activations mentioned in the above are detected by a non-clustered node for a predefined time interval,

then the non-clustered node sets its *cid* to its own *id* and broadcasts a *CH*(*cid*, *nc_degree*) packet to declare itself as a cluster head.

The cluster maintenance procedure for link failure events:

- When a link failure occurs between a cluster head and one of its member nodes,
  - The associated cluster head removes the member node located at the other end of the detected failed link from its *Member_Table*.
  - Each of the associated member nodes sets its own *cid* to zero and becomes a non-clustered node.
- When a link failure occurs in a distributed gateway (a pair of ordinary nodes), each of the two ordinary nodes updates its own routing information and informs its own cluster head.
- When an ordinary node receives a *DISCONNECT*(*id*) packet and its *cid* is equal to the *id* of the received packet, it sets its *cid* to zero and becomes a non-clustered node.
- When a cluster head receives a *DISCONNECT*(*id*) packet and the *id* of the packet is equal to any *id* in its *Member_Table*, it removes the node sending the packet from its *Member_Table*.

4. Computer simulations

In this section, we will evaluate the performance of our clustering algorithm and compare it with the lowest-ID clustering algorithm [8], the highest-connectivity clustering algorithm [8], and EKC [15]. The performance metrics we will measure are the number of clusters and the variation of cluster sizes.

4.1 Determining the proper value for parameter D_{avoid}

Since the performance of our clustering algorithm heavily depends on the value of D_{avoid}, it is crucial to select a

proper value for D_{avoid}. In the following, we will use computer simulations to determine the suitable values of D_{avoid} for three different network sizes, i.e., a 40-node network, an 80-node network, and a 120-node network.

When the size of network is 40 nodes, the simulation results are shown in Figure 6. It can be observed that the number of clusters and the variation of cluster sizes of the clustering architecture constructed by our clustering algorithm are smaller when D_{avoid} is between 3 and 4. The simulation results for 80-node networks are shown in Figure 7. Obviously, the clustering architecture constructed by our clustering algorithm has fewer clusters and lower variation of cluster sizes when D_{avoid} is about 7. Finally, for the case that the size of network is 120 nodes, the simulation results are shown in Figure 8. At this time, the clustering architecture constructed by our clustering algorithm has fewer clusters and lower variation of cluster sizes if D_{avoid} is 10.

4.2 Comparisons in terms of the number of clusters and the variation of cluster sizes

Now we will compare the performance of our clustering algorithm with other existing clustering algorithms in terms of the number of clusters and the variation of cluster sizes. The clustering algorithms we will compare are the lowest-ID clustering algorithm [8], the highest-connectivity clustering algorithm [8], and EKC [15]. Note that the last two algorithms also focus on minimizing the number of clusters. In the following comparisons, depending on the sizes of networks considered, our clustering algorithm will adopt the different best values for D_{avoid} according to the simulation results presented in the above.

About the number of clusters, the simulation results are given in Figure 9(a). From these curves, it can be observed that the number of clusters generated by our clustering algorithm is about 23.25% less than that generated by the lowest-ID clustering algorithm, about 8.08% less than that generated by the highest-connectivity clustering algorithm, and about 6.64% less than that generated by EKC.

Figure 9(b) shows the simulation results on the variation of cluster sizes. We can observe that the

clustering architecture established by our clustering algorithm has always a smaller variation of cluster sizes than those established by the other three clustering algorithms under all the three different network sizes.

5. Conclusions

The performance of a clustering architecture has been demonstrated to be closely related to the number of its clusters. Furthermore, reducing the node degree of a cluster head will extend its lifetime and improve its cluster's stability. In this paper, we have proposed an efficient clustering algorithm to minimize the number of clusters and the variation of cluster sizes. In our clustering algorithm, we first elect the node with the largest *weighted_value* as the cluster head. If there is a tie, then the second parameter *nc_degree* will be involved in the election. The node with the largest *nc_degree* will become the cluster head. Finally, if there is still a tie, then the node with the highest identifier prevails. The computer simulations have demonstrated that our clustering algorithm is efficient in reducing the number of clusters as well as the variation of cluster sizes. In fact, the number of clusters generated by our clustering algorithm is about 23% less than that generated by the lowest-ID clustering algorithm, about 8% less than that generated by the highest-connectivity clustering algorithm, and about 7% less than that generated by EKC. Furthermore, the clustering architecture established by our clustering algorithm has always a smaller variation of cluster sizes than those established by the other three clustering algorithms under different network sizes. Finally, our clustering algorithm can preserve its structure as much as possible when nodes are moving and/or the topology is slowly changing.

References

[1] P. Alimonti, and T. Calamoneri. "Improved approximations of independent dominating set in bounded degree graphs," *In Proc. of 22nd WG, pp 2-16, LNCS 1197*, Springer-Verlag, 1997.

[2] S. Basagni, "Distributed clustering for ad hoc networks," *Proc. of IEEE I-SPAN*, pp. 310-315, Jun. 1999.

[3] S. Basagni, "Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks," *Proc. of IEEE VTC*, vol. 2, pp. 889-893, Sept. 1999.

[4] S. Basagni, "Distributed and mobility-adaptive clustering for ad hoc networks," *Tech. Rep. UTD/EE-02-98, Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas,* July 1998.

[5] P. Basu, N. Khan, and Thomas D. C. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," *Proc. of IEEE ICDCS '01*, pp. 413-418, Apr. 2001.

[6] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," Discrete Mathematics, vol. 86, pp. 165-177, 1990.

[7] G. Chen, F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, "Connectivity based k-hop clustering in wireless networks," *Proc. of the 35th Hawaii International Conference on System Sciences*, 2002.

[8] M. Gerla and J. Tsai, "Multicluster, mobile, multimedia radio network," *ACM/Baltzer Journal of Wireless Networks*, vol. 1, pp. 225-238, 1995.

[9] T. C. Hou and T. J. Tsai, "An access-based clustering protocol for multihop wireless ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 19. No. 7, pp. 1201-1210. Jul. 2001.

[10] A. Iwata, C. C. Chiang, G. Pei, M. Gerla, and T. W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp.1369-1379, Aug. 1999.

[11] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan," A cluster-based Approach for Routing in Dynamic Networks," *ACM SIGCOMM Computer Commnications Review*, vol. 27, no. 2, pp. 49-64, Apr. 1997.

[12] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE J. Select. Areas Commun.*, pp. 1265-1275, Sept. 1997.

[13] B. McDonald and T.F. Znati, "A mobility-based framework for adaptive clustering in wireless ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1466-1487, Aug. 1999.

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

[15] D. C. Su, S. F. Hwang, C. R. Dow, and Y. W. Wang, "An efficient k-hop clustering routing scheme for ad-hoc wireless networks," Accepted by *Journal of the Internet Technology*.

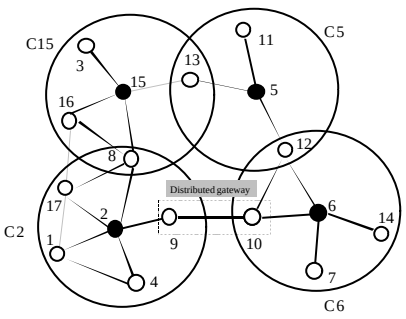


Figure 1. A clustering architecture

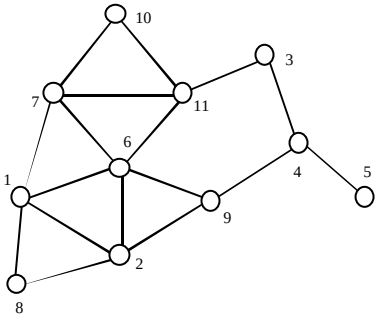


Figure 2. A mobile ad hoc network

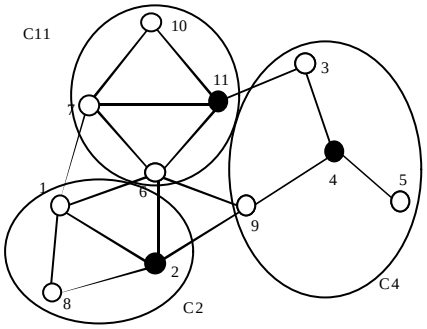


Figure 3. The clustering architecture established by our clustering algorithm

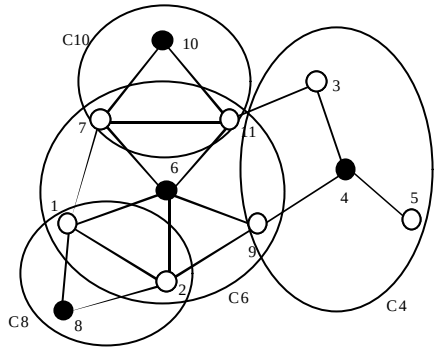


Figure 4. The clustering architecture established by the highest-connectivity clustering algorithm or the connectivity based k-hop clustering algorithm with k=1

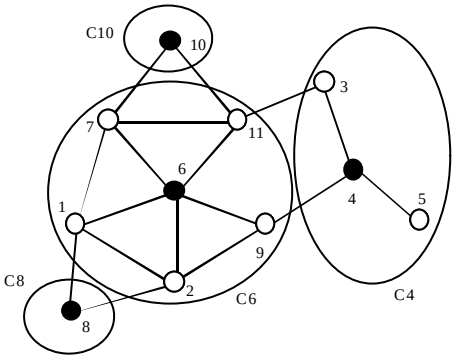
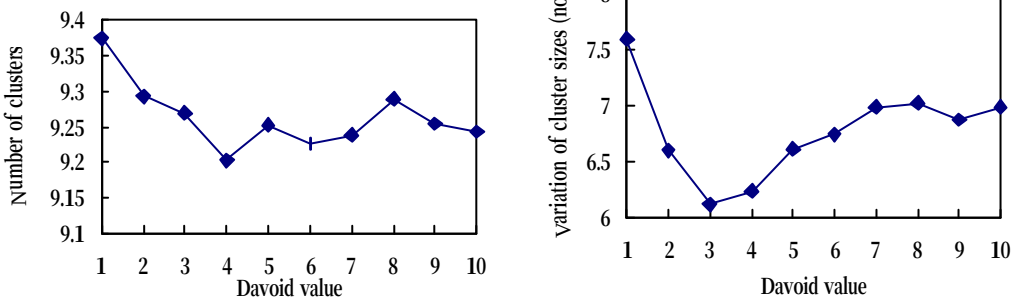
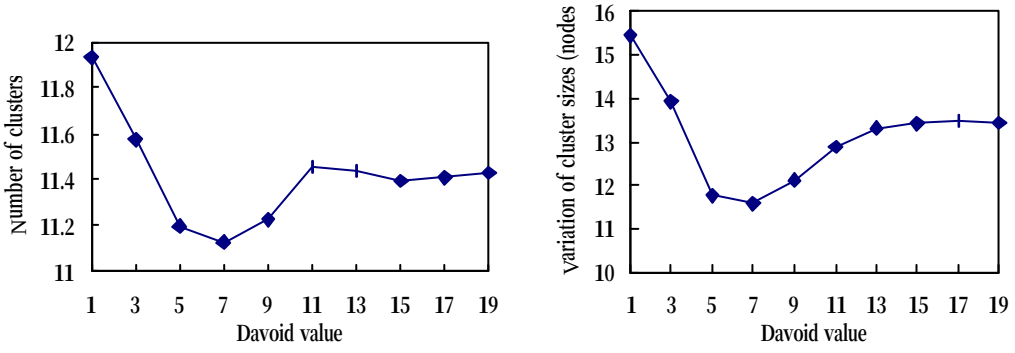


Figure 5. The clustering architecture established by EKC with k=1



(a) (b)

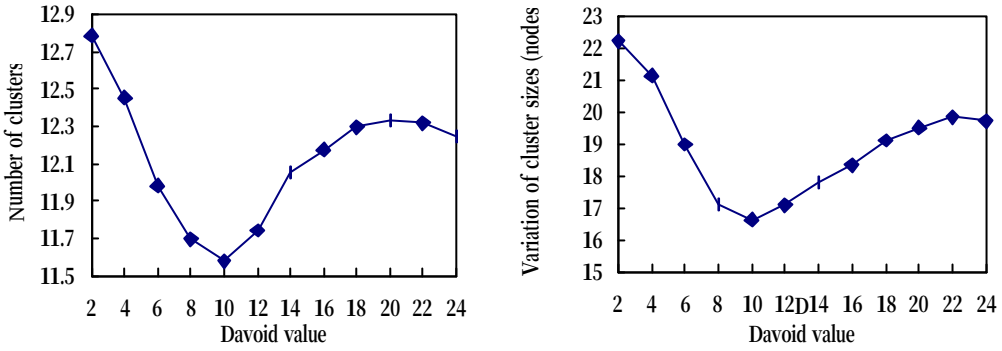
Figure 6. Determining the best value of D_{avoid} to minimize (a) the number of clusters (b) the variation of cluster sizes (when the network has 40 nodes)



(a) (b)

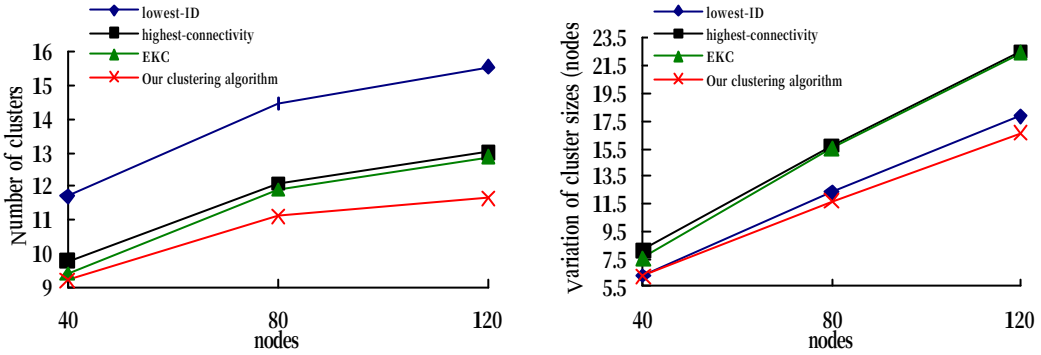
Figure 7. Determining the best value of D_{avoid} to minimize (a) the number of clusters (b) the variation of cluster sizes (when the network has 80

nodes)



(a) (b)

Figure 8. Determining the best value of D_{avoid} to minimize (a) the number of clusters (b) the variation of cluster sizes (when the network has 120 nodes)



(a) (b)

Figure 9. Comparisons between different clustering algorithms in terms of (a) the number of clusters (b) the variation of cluster sizes