Secured Clustering Algorithm for Mobile Ad Hoc Networks

B. Kadri[†], A. M'hamed[††], M. Feham[†]

[†]/ STIC Lab., Department of Electronics, University of Tlemcen, Tlemcen, Algeria
[††]/ National Institute of Telecommunications, Evry, France

Summary
Future Mobile ad hoc networks may increase in size to reach the threshold of thousands of nodes per system (commercial or military). A way to handle this increasing number of nodes is to manage them in a hierarchically structure also called clustering. In this paper, we propose a secured weight-based clustering algorithm allowing more effectiveness, protection and trust in the management of cluster size variation. This algorithm is called Secured Clustering Algorithm (SCA), since it includes security requirements by using a trust value defining how much any node is trusted by its neighbourhood, and using the certificate as node's identifier to avoid any possible attacks (Spoofing). SCA elects cluster-head according to its weight computed by combining a set of system parameters (Stability, Battery, Degree … etc). It also overcomes some limits in existed algorithms by defining new mechanisms as cluster division, merging diminution and extension. We also propose to use efficient flooding to send beacons within the cluster to avoid overhead due to flooding. SCA forms a hierarchical structure which can be used for both security and routing protocols.

Key words:
Ad Hoc, MANET, Security, Trust, Weight-based Clustering.

1. Introduction

Mobile ad hoc network (MANET) is a collection of wireless hosts that communicate with each other through multi-hop wireless links. Due to the absence of fixed infrastructure, nodes must collaborate between them to accomplish some operations like routing and security. Some envisioned MANETs, such as mobile military networks or future commercial networks may be relatively large (e.g. hundreds or possibly thousands of nodes per autonomous system). A way to support the increasing number of nodes in MANET is to subdivide the whole network into groups, and then create a virtual backbone between delegate nodes in each group. In ad hoc network this operation is called clustering, giving the network a hierarchical organization. A cluster is a connected graph including a cluster head responsible of the management of the cluster, and (possibly) some ordinary nodes. Each node belongs to only one cluster. Clustering has several advantages. First clustering allows the reuse of resource which can improve the system capacity, in the way that information is stored once on the cluster head. Secondly clustering may optimally manage the network topology, by dividing this task among specified nodes which can be very useful for routing since any node is identified by its identity and the identity of the cluster-head of the cluster to which it belongs, simplifying by this way the forwarding of messages. Several cluster based adaptations has been proposed for existed routing protocols [1, 2] and other protocol as ZRP (*zone routing protocol*) [3], CBRP (*cluster based protocol*) [4] have originally exploited this concept. Clustering for security can simplify the management of Certificate Authority in a Public Key Infrastructure (PKI) by affecting the full or a subset of Certificate Authority services to cluster-heads, ensuring in this way the availability of the Certificate Authority [5,6,7].

The usefulness of clustering is justified by the increasing number of node in MANET, so most recent research work has given more attention to this concept regarding the election of cluster-head and the maintenance of cluster architecture [8]. The remainder of this paper is organised in 5 sections. Section 2 gives an overview of existing algorithms in clustering while Section 3 shows their limitations in features required by ad hoc networks. Section 4 is devoted to the description of our proposed secured clustering algorithm. Section 5 shows the simulations results compared to two other known algorithms. Finally, section 6 conclude by reminding the main improvements introduced by our algorithm.

2. Related work

2.1 Highest-Degree Algorithm

The Highest-Degree Algorithm, also known as connectivity-based algorithm [9], was originally proposed by Gerla and al. This algorithm is based on the degree of nodes assumed to be the number of neighbours of a given node. Whenever the election procedure is needed, nodes broadcast their Identifier (ID) which is assumed to be unique in the same network. According to the number of received IDs every node computes its degree and the one having the maximum degree becomes cluster-head.

Manuscript received March 5, 2007
Manuscript revised March 25, 2007

2.2 Lowest-ID Algorithm

The Lowest-ID, also known as *identifier-based clustering algorithm*, was originally proposed by Baker and Ephremides [10]. This algorithm assigns a unique ID to each node and chooses the node with the minimum ID as a cluster-head. Whenever a node with a lowest ID is detected in the cluster the cluster-head must delegate it responsibility to this node to be cluster-head.

2.3 Mobility-based *d*-hop Algorithm

The authors in [11] propose a clustering scheme based on the real distance between nodes. They propose to calculate an estimate value of the distance between nodes by measuring the received signal strength taken from periodic beaconing or hello messages used in some routing protocols. According to this estimated value they can determine the stability of every node. Then they elect the most stable node as cluster-head.

2.4 Weighted Clustering Algorithm WCA

This algorithm was proposed by SAJAL and TURGUT [12]. Their authors begin from the supposition that the algorithms cited above don't treat all the characteristics of ad hoc networks that influence on the cluster-head election. Since every one of them is intended to work for some specific kind of situation of ad hoc networks. For this purpose they have proposed a weight based algorithm, which means that the cluster-head is elected according to its weight, which is computed by combining a set of system parameters like battery and mobility.

3. Limitation of existing algorithms

As said before, the lot of algorithms proposed in literature have giving solution to only some specific problems of ad hoc networks. However none of them deals with the entire characteristics of ad hoc networks (mobility, transmission range, size of the network, capabilities of the nodes…).

For example the highest-degree, lowest-ID algorithms create one hop clusters, which are too small for large networks resulting on a big number of clusters, which complicate the virtual backbone management. They are also sensitive to small changing in the topology.

The WCA and Mobility based algorithms try to include the mobility (stability) of nodes as a factor in the election procedure, in order to elect the most stable node as cluster-head. But their methods to compute stability are based on some assumptions which are not always valid in all ad hoc networks. The method proposed in WCA for computing the mobility of nodes, is based on the idea that nodes can define their position at any time which isn't possible without the existence of GPS (Global Position System). Another method proposed in mobility based *d*-hop algorithm relies on the idea that nodes have equal antennas and transmit with the same power, to compute an estimate value of the distance between nodes. However this supposition is rarely guarantied because ad hoc networks are composed by heterogeneous nodes having different capabilities and antennas.

Another problem is the complex computing used in the two algorithms to compute stability (mobility) which may consume nodes resources. The mobility based *d*-hop algorithm tries to solve the problem of the small size of clusters encountered in other algorithm by creating *d* hops clusters. However it doesn't treat the aspect of the maximum number of nodes in clusters, because cluster-heads can't serve infinite number of nodes.

Another observation is related to security features which are not included in the above algorithms. As mentioned in [5,6,7], security problem must be taken into consideration in all schemes devoted to ad hoc networks.

4. Secured Clustering Algorithm

In this section, we will describe our proposed clustering algorithm called Secured Clustering Algorithm (SCA). To overcome the security limits of existing algorithms, we have included trust values and certificates making by this way our algorithm useful for security purposes as PKI based on clustering [5,6]. SCA is a weight based clustering algorithm which uses a weight computed from a set of parameters to elect cluster-heads.

4.1 Basis for our design

The main basic concepts used to derive the needed parameters are given below:

a- *The Max Value*: represents the upper bound of the number of nodes that can simultaneously be supported by a cluster-head. Since mobile nodes have limited resources, therefore they can't handle a great number of nodes. This value is defined according to the remainder of resources of the cluster-head.

b- *The Min Value*: represents the lower bound of the number of nodes that belong to a given cluster before proceeding to the extension or merging mechanisms. This value is global and the same for the entire network. The *Min Value* may avoid the complexity due to the management of great number of clusters.

c- *D hops Cluster*: as we have said, one hop clusters are too small for large ad hoc networks, therefore SCA creates *D* hops clusters where *D* is defined by the underlying protocol or according to the cluster-head state (busy or not). By the way, the diameter of the cluster can be extended in some situations.

d- *Identity (ID):* is a unique identifier for each node in the network to avoid any spoofing attacks or perturbation in the election procedure. We propose to use certificate as identity [5, 6], therefore we suppose the existence of an online or offline Public Key Infrastructure managing the certificate distribution.

e- *Weight*: each node is elected cluster-head according to its weight which is computed from a set of system parameters. The node having the greatest weight is elected as cluster-head.

4.2 Election criteria

The following parameters define the criteria on which SCA rely to elect the cluster-head.

a- *Trust value*: it measures how much any node in the network is trusted by its neighbourhood. It's defined as the average of trust values received from each neighbouring node. In order to compute the trust value, we suppose that every mobile node has an intrusion detection mechanism [13] to determine if a node is considered as trust or not by periodically collecting information about the behaviour of each neighbour.

$$T = \frac{\sum_{i=1}^{N} T_i}{N} \qquad (1)$$

T_i is the received trust value from node *i*.

b- *Degree*: is the number of neighbours of a given node, within a given radius. This parameter is used to choose as cluster-head the node having the maximum neighbours to serve the more number of nodes.

c- *Battery power*: this factor is the capability of a node to serve as long as possible. Since cluster-head has extra responsibility and it must communicate as far/long as possible, thus it must be the most powered node.

d- *The Max Value*: as defined above, this parameter is used in the election procedure to elect as cluster-head the node which can handle the maximum of nodes.

e- *Stability* (Mobility): this is a useful parameter when electing the cluster-head. In order to elect the most stable node as cluster-head, avoiding frequent roaming, we have computed the *stability* using the following metrics:

i. *The distance*: the *distance* between two nodes A,B ($D_{A,B}$), is the number of hops between them, which can be obtained from the packets sent from one to other, or hello message used in routing protocols. The possibility of obtaining the number of hops between two nodes is evident and simple within all existed routing protocols.

ii. *The mean distance*: which is defined as the average of distances between node A and all its neighbours.

$$MD_A = \frac{1}{N}\sum_{n=1}^{N} D_{A,n} \qquad (2)$$

N is the degree of A

MD takes values between 1 and *D* and it defines the radius where there exists the great density of nodes. For example when *MD* = 2 this means that the majority of neighbours are within 2 hops.

iii. *Stability*: is defined as the difference between two measures of *MD* at *t* and *t-1*, it becomes large when the node goes far from its neighbours or whenever its neighbours are going in other direction than the one taken by the considered node. This value is compared with *D* and a node is considered as most stable if it has the less value of *ST*.

$$ST_A = MD_t - MD_{t-1} \qquad (3)$$

f- *Weight Factors*: each of the previous parameters is called partial weight. Each parameter is affected a weight factor defining its degree of importance for the underlying protocol or the network. Since only a subset of these parameters can be used according to the requirements of the network and the underlying protocol, these factors provide more flexibility and large scale of use to our algorithm. For example trust value may take the great value if the underlying protocol is a key management protocol. Factors are given values between 0 and 1, so that the sum of factors is 1.

$$\sum_{i=1}^{n} F_i = 1 \qquad (4)$$

n is the number of factors

g- *Global Weight*: using all parameters cited above every node in the network computes its global weight using equation (5). Depending on this weight a given node can be elected as cluster-head or not.

We denote W_T, W_D, W_B, W_M, W_S the partial weights and F_T, F_D, F_B, F_M, F_S are the weight factors corresponding respectively to *Trust value*, *Degree*, *Battery*, *Max Value*, and *Stability*. The global weight is computed as follow

$$W_G = F_T \times W_T + F_D \times W_D + F_B \times W_B + F_M \times W_M + F_S \times (-W_S) \qquad (5)$$

As we can observe the value of W_S is retrieved from the global weight in order to elect the node with the greatest weight, because the stable node is the node with the smallest value of W_S, which keeps the equation (5) coherent for our assumption.

4.3 Beaconing

In order to maintain the structure of clusters, cluster-heads must broadcast periodically special messages called Beacons. These Beacons contains commands sent to cluster-members to collaborate in order to execute any one of the cluster management commands. Beacon has two fields; the first one contains the certificate of the

cluster-head, and the second contains the code of the command.

Fig. 1 Beacon structure

Beacons are periodically broadcasted in the network, generally using flooding. As the number of nodes increases, flooding suffers from the increase of (1) Redundant and superfluous packets, (2) Probability of collision, (3) Congestion of wireless medium, and (4) Security risks (vulnerability due to traffic analysis of the huge number of flooded encrypted packets). In our algorithm we propose an efficient flooding, such as only a subset of nodes participate in flooding using one of the heuristics defined in [14], which may minimize significantly overhead and congestion due to flooding.

4.4 The election procedure

This operation is invoked whenever a neighbourhood has no cluster-head, or whenever one of the cluster-heads isn't able to achieve its responsibilities. The invocation of the election procedure doesn't mean that all cluster-heads are replaced. Let's assume that a set of nodes desire to create or to maintain a clustering architecture, so they must collaborate to execute the following steps:

Discovery stage: the purpose of this step is to get information about the neighbourhood where the election procedure is invoked. Thus nodes desiring to be cluster-head send *cluster-head_ready* beacons within the radius of *D* hops. Each node when receiving this beacon estimates a trust value and sends it back to the asking node. After a discovery period T_d, nodes having initiated this operation can derive from the received responses the following information:
-*Degree*: this is the number of received responses.
-*Stability*: calculated using equation (2) and (3).
-*Trust value*: computed using equation (1).

Computing weight: after the discovery stage, each node adds to the previous parameters the state of its *battery* and the *max value*, then combines them with the corresponding *weight factors* and computes the *global weight* using equation (5). This weight is broadcasted within the same neighbourhood. Using the different received weights, nodes choose as cluster-head the node having the maximum weight.

Elaboration of the virtual backbone: Whenever the previous steps are successfully achieved, each elected cluster head need to discover each other to elaborate a virtual backbone to ensure inter-cluster services. Thus every new elected cluster-head broadcast a discovery request over the entire network; cluster-heads receiving this request register the certificate of the new cluster-head and send him their certificate.

4.5 Functions and notation

-*Join*: attach a given node to a cluster-head.
-*Get_Beacons*: returns all beacons received by a given node.
-*Broadcast:* broadcasts a message within a given area, using efficient flooding.
-*Send and receive*: respectively sends or receives any kind of messages.
-*Roam*: detach a node from a cluster-head and attach it to a new one, using Init procedure.
-We denote cluster-head as CH, cluster-member as CM.

4.6 Cluster management:

We define cluster management procedure as the actions executed to maintain the stability of clustering architecture. These actions mainly manage the increasing and decreasing number of nodes in clusters. Therefore we define the following procedures:

4.6.1 Initialisation of nodes

The *Init procedure* is executed by each node in a no determinist status. A node with this status is a node which isn't attached yet to any cluster, this may be caused by a link failure, a roaming, or whenever a node coming for the first time to the network. First the node listens if there is any neighbouring CH (Line 3). If this is the case it chooses the nearest cluster and joins it (Line 11, 12). Otherwise it launches the election procedure (section 4.4) to elect a new CH in this neighbourhood (Line 6).

Procedure Init ()

```
1. Begin
2.  If status=N then
3.  CH-list=Get_beacons();
4.  If CH-list=null then
5.        begin
6.          Election ( );
7.          exit();
8.        end;
9.  else
10.       begin
11.         CH=CH-list.Get_Nearest ( );
12.         JOIN(CH);
13.       end;
14.  exit;
15.  End;
```

4.6.2 Cluster division

As mentioned before a node can't serve for ever as CH, because it has limited resources (battery, memory…). So, whenever it becomes busy (can't support the increasing number of nodes), the CH launches a cluster division procedure to divide the cluster into two small clusters with reasonable number of nodes. Therefore the CH broadcasts *Cluster_Division* request to its CMs (Line 4). Whenever this request is received, each CM compute its weight and sends it back to the CH, which saves them (Line 7). Then the CH chooses as a new CH the farthest node with the maximum weight and sends him a grant response (Line 10). Then the new CH begins sending beacons and creates its own cluster.

Procedure Cluster_Division( )

```
1.  Begin
2.  If  Member > MAX then
3.  Begin
4.    Broadcast (Cluster_Division);
5.     Wait;
6.     While T_D do
7.        list= Receive (Cert, W);
8.     enddo ;
9.     Cert= List.get-max-weight.
10.    Send(grant,Cert);
11. endif;
12. End;
```

4.6.3 Cluster size reduction

This operation is executed after the division of the cluster and aims to reduce the cluster radius from *D* to *D-1*, which means that beacons don't reach the boundaries of the cluster, resulting on the roaming of boundaries nodes to other clusters including new cluster creation.

4.6.4 Cluster merging

In the algorithms taken from the literature, no lower bound is defined to limit the minimum number of nodes in a cluster, resulting on some clusters with two or one nodes which is not suitable. Therefore, in our algorithm we propose to merge such clusters immediately with the nearest cluster if it exists by executing the merging procedure. First, the CH begins to listen if there are any neighbouring CHs (Line 4); if this is the case it broadcast a merging request (Line 7). Then it wait until receiving all confirmation from its CMs or the expiration of the delay T_M (Line 10-14) to choose the nearest cluster and roams to that cluster (Line 16).

Procedure Cluster_Merging( )

```
1.  Begin
2.  If Member<min then
3.  Begin
4.   CH-list=Get_beacons();
5.   If CH-list=null then
6.    exit (0);
7.   Broadcast (Cluster_Merging);
8.   Wait();
9.   i=0;
10.  While T_M or I< =Member do
11.   begin
12.    Receive(Confirmation);
13.     i++;
14.   end;
15. if Member=0 then
16.        Roam(CH-list.Get_Nearest);
17. End;
18. End;
```

4.6.5 Cluster size extension

This operation is executed whenever the merging procedure isn't successfully executed, thus the CH proceed to the extension of the radius of the cluster from *D* to *D+1*. Therefore beacons are broadcast within a largest area allowing new nodes to join the network which may increase the number of nodes in this cluster.

4.6.6 Other scenarios

- Roaming: this operation is executed whenever the node goes far from its cluster, so it must be detached from the old CH (change the status to no determinist), and attached to the nearest CH by executing the Init procedure.
- Link failure: in this situation we assume that the contact with the CH isn't active due to any causes, so the node permutes to no determinist status and executes the Init procedure.

5. Simulation results

The performance of SCA is evaluated using ClusterSIM, a simulator we have developed to test and evaluate clustering algorithms. It mainly provides some results as the number and size of clusters, roaming requests and cluster-head election. To perform comparison tests we've implemented three algorithms (Highest Degree Clustering Algorithm (HDCA), Mobility Based Clustering Algorithm (MBCA) and Secured Clustering Algorithm (SCA)). The scenarios were generated using parameters listed in table 1.

Parameters	*Values*
Network size	500*500 and 100*100
Number of Nodes	25-300

Max speed	20 m/s
Pause Time	0 s
Transmission range	20 and 100m
Max of nodes in a cluster (used only for SCA)	Randomly chosen between 10 and 30
Simulation time	300s

Table 1: Simulation parameters

The purposes of the simulation runs are:
- To prove the unfeasibility of one hop clusters.
- To justify our choice to limit the size of clusters.
- To compare our algorithm to existing algorithms.
- To test the performance of SCA for dense networks and in different areas.

For all the algorithms, the number of clusters is relatively high when the transmission range is small or when the area is big. As nodes are out of range of each other, therefore they form one or two node clusters. As

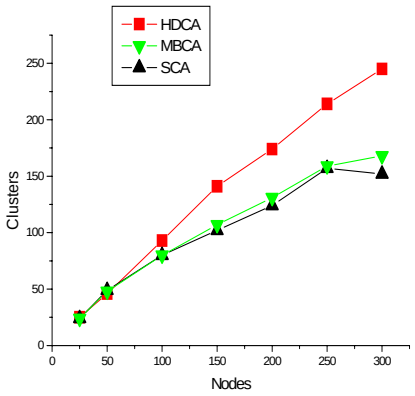

Fig. 2: Simulated 500*500 m^2 area with transmission range of 20m.

shown in Figures 2 and 3, the number of clusters created by the HDCA is very large. However MBCA and SCA provide less number of clusters because they create two hops clusters covering large area compared to HDCA which creates one hop clusters. This justifies our choice to extend the radius of clusters to more than one hop.

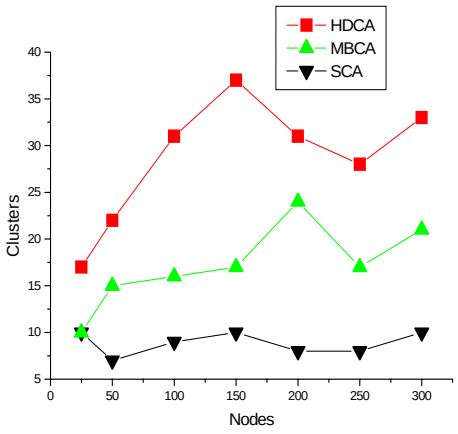

Fig. 3: Simulated 500*500 m^2 area with transmission range of 100 m.

In Figure 4 we compare SCA, HDCA and MBCA by modifying the cluster size from 2 to 3 hops for MBCA and SCA because this parameter is configurable. We observe that MBCA and HDCA keep the same number of clusters for any number of nodes, to reach the threshold of 300 nodes per cluster, which is far from the capability of mobile nodes in MANETs. However the SCA creates more clusters to manage the increasing number of nodes, and the number of nodes in each cluster remains within the threshold of 20 nodes per cluster, which is reasonable in ad hoc networks.

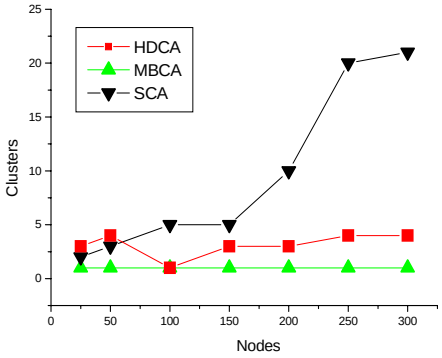

Fig. 4: Simulated 100*100 m^2 area with 100m-transmission range and D=3

From the simulations made on the MBCA, we can observe that for the transmission range of 100 m in small areas 100*100 m^2, 200*200 m^2 and 300*300 m^2, the algorithm creates a very small number of clusters which stays under 10 clusters in the 300*300m^2 area and under 3 in 100*100 m^2 and 200*200 m^2 areas. Obviously, this

is too small to manage the increasing number of nodes because CHs are mobile nodes, which can't support and serve a great number of nodes (150 nodes) simultaneously.

From the simulations made on the HDCA, as we can easily predict, we meet the same problems as for the MBCA, because it doesn't make any assumption on the maximum number of nodes supported by a CH. Thus it creates small number of clusters to manage the increasing number of nodes. For example, in 100*100 m^2 and 200*200 m^2 areas, it always creates less than 5 clusters to manage a set of nodes going from 20 to 300 nodes. However it creates more clusters for 300*300 m^2 and 1000*1000 m^2 areas because the nodes are out of the transmission range of each other.

The simulations made on the SCA for different size of the area shown in Figure 5. As we can observe, SCA manages the increasing number of nodes in the network by creating more clusters. This is done because the number of nodes in each cluster is limited, therefore when there is more nodes in the network, the algorithm manage them by creating more clusters

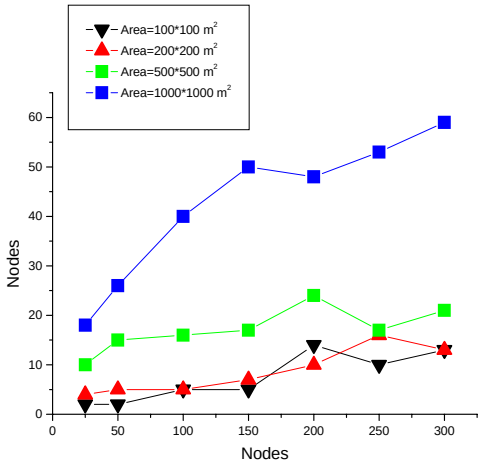

Fig. 5 Test of SCA in different areas,
with transmission range of 100 m, and clster size of 2 hops

For example, in 100*100 m^2, 200*200 m^2 and 300*300 m^2 areas, the average number of nodes in each cluster stays around 25 nodes per cluster which is reasonable considering the simulation inputs, in opposition of HDCA and MBCA which keep the same number of cluster for all network configurations

Figure 6 shows testing of the SCA reaction to different areas with the radius of 3 hops. Thus we observe that SCA manage always the increasing number of nodes in the network by creating more clusters to stay in the same average of nodes which is randomly chosen between 10 and 30 nodes. We can also observe that in large areas (1000*1000 m^2), it creates a big number of clusters such as in Figure 5. However to overcome this, the size of clusters is increased from 2 to 3 hops, which results on the decreasing of the number of clusters. This results is observed in Figure 6 in which the number of clusters created in the 1000*1000 m^2 area doesn't reach the 30 clusters, however with the same configuration and the size of 2 hops it creates more than 55 clusters.

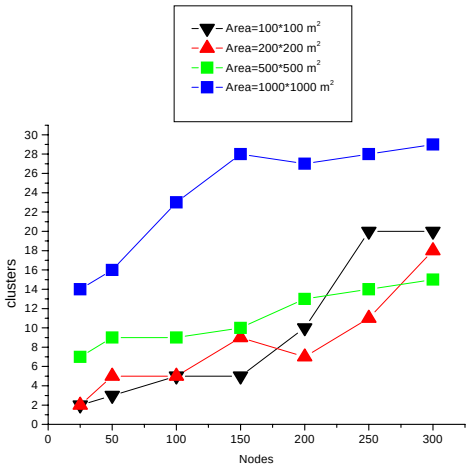

Fig. 6 Test of SCA in different areas,
with transmission range of 100 m, and cluster size of 3 hops

6. Conclusion

In this paper we have proposed a new clustering algorithm called Secured Clustering Algorithm. In this algorithm we have tried to solve some problems encountered in existed algorithms by:

- Including the security features through the use of voting mechanism to elect the most trusted node, we've also proposed to use certificate as identifier to avoid spoofing attacks.
- Defining a set of system parameters for the election procedure to elect the most power node as cluster-head, as well as a new method to compute stability more simple and possible to be used in ad hoc network.
- Defining an upper and lower bounds to limit the number of nodes in clusters, giving birth to new mechanisms like cluster size extension and cluster size diminution.
- Creating D hops cluster, resulting on a less number of clusters which may also reduce the number of roaming requests, and management difficulties.

- Proposing the use of efficient flooding to broadcast beacons or any other request over the network, in order to avoid overhead caused by flooding.

Finally, through the simulation results, we've highlighted the need of the previously mentioned choices. We have also evaluated the efficiency of our algorithm compared to other algorithms taken from the literature, considering the number and the behaviour of nodes in clusters.

References

[1] Farid Jaddi, Béatrice Paillase. *A Cluster Procedure for the Dynamic Source Routing Protocol in Ad hoc Networks.* Med-Hoc-Net 2004, The Third Annual Mediterranean Ad Hoc Networking Workshop.

[2] An Huiyao, Lu Xicheng, and Peng Wei. *A Cluster-Based Multipath Routing for MANET.* Med-Hoc-Net 2004, The Third Annual Mediterranean Ad Hoc Networking Workshop.

[3] Haas, Z.J., Pearlman, M.R. and Samar, P. *Zone Routing Protocol (ZRP).* IETF Internet Draft, draft-ietf-manet-zrp-04.txt, July 2002.

[4] Mingliang Jiang, Jinyang Li, Yong Chiang Tay. *Cluster Based Routing Protocol (CBRP) Functional Specification.* IETF Internet Draft, draft-ietf-manet-cbrp-spec-00.txt August 1998

[5] Mohamed Elhoucine Elhdhili, Lamia Ben Azzouz, Farouk Kamoun. *A Totally Distributed Cluster Based Key Management Model for Ad hoc Networks.* Med-Hoc-Net 2004, The Third Annual Mediterranean Ad Hoc Networking Workshop.

[6] M. Bechler, H.-J. Hof, D. Kraft, F. Pählke, L. Wolf. *A Cluster-Based Security Architecture for Ad Hoc* Networks. IEEE, INFOCOM 2004.

[7] Panagiotis Papadimitratos, and Zygmunt J. Haas. *Secure Data Communication in. Mobile Ad Hoc Networks.* IEEE Journal on Selected Areas In Communications, Vol. 24, No. 2, 2006 PP. 343,356.

[8] Stefano Basagni, Michele Mastrogiovanni, Alessandro Panconesi, and Chiara Petrioli. "Localized Protocols for Ad Hoc Clustering and Backbone Formation: A Performance Comparison". IEEE Transactions on parallel and distributed systems, Vol. 17, No. 4, 2006, pp 292,306.

[9] M. Gerla and J.T.C. *Tsai. Multicluster, mobile, multimedia radio network, Wireless Networks.* Vol. 1, No. 3, 1995, PP. 255–265.

[10] 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.

[11] I.I. ER, and Winston K. G. Seah, *Mobility-based D-hop Clustering Algorithm for Mobile Ad hoc Networks*. IEEE WCNC, Atlanta, USA, March 2004

[12] M. Chatterjee, S. K. Das and D. Turgut. *WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks.* Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), Vol. 5, No. 2, April 2002, pp. 193-204.

[13] H. Luo and S. Lu, 2000. *Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks.* Technical Report 200030, UCLA Computer Science Department.

[14] Yunjung Yi, Mario Gerla, Taek-Jin Kwon. Efficient Flooding in Ad-Hoc Networks using On-Demand (passive) Cluster Formation. Proceedings of Mobihoc, June 2003.

Benamar Kadri, received his engineer degrees in informatics from the University of Tlemcen, Algeria in 2004, and his M.S. degrees in networks and telecommunication systems within of the same University. Member of STIC laboratory in the University of Tlemcen, his recent work is dealing with mobile ad hoc networks, their security, routing and management.

Abdallah M'hamed, Associate professor in Network security and dependability. He received his Doctor degree in dependability studies from the Technological University of Compiègne, France. In 1990 he joined the National Institute of Telecommunications, in Evry, France. His current teaching activities are dealing with network security services , cryptographic protocols and access controls. Member of the Handicom laboratory, his recent research activities are focused on authentication protocols and architectures, security and privacy in smart environments.

Mohammed Feham received his PhD in Engineering in optical and microwave communications from the university of Limoges, France in 1987, and his PhD in science from the university of Tlemcen , Algeria in 1996. Since 1987 he has been assistant professor and professor of microwave and communication Engineering his research interest is in telecommunication systems and mobile networks.