A Distributed Key Management Scheme based on Multi hop Clustering Algorithm for MANETs

Abdelmajid HAJAMI, Kamal OUDIDI and Mohammed ELKOUTBI

SI2M, ENSIAS, Université Mohammed V – Souissi B.P. 715, ENSIAS – Rabat Maroc

Summary
Mobile ad hoc networks (MANETs) have been proposed as an extremely flexible technology for establishing wireless communications. In comparison with fixed networks, some new security issues have arisen with the introduction of MANETs. Secure routing, in particular, is an important and complicated issue. Clustering is commonly used in order to limit the amount of secure routing information. In this work, we propose an enhanced solution for ad hoc clustering based on multi hops and network density. This solution will be used as a framework to manage cryptographic keys in a distributed way. This paper details the density-based clustering algorithm for the standard OLSR protocol. Our algorithm takes into account the node mobility and gives major improvements regarding the number of elected cluster heads. Our objective is to elect a reduced and less mobile cluster heads that will serve for keys exchange.

Keywords :
Clustering; Key Management; MANET

1. Introduction

MANETs are strongly based on self-organization and self-stabilization. Several Ad hoc routing protocols proposed in the MANET working group at IETF[1] make flat routing. That means that there is no hierarchy and all terminals have the same role. However, as the size of the network grows, the performances of the network decrease. This is due to the additional control traffic generated by nodes in the network. To minimize the effect of this additional traffic, some mechanisms were adopted, as shown in the OLSR protocol [1].

Clustering is commonly used to limit the amount of routing information. In this work, we aim to define a new clustering approach based on multi-hops and network mobility. The proposed approach must enhance the routing process and produces a small number of stable (less mobile) cluster heads that can be used as a framework for key management and distribution.

A cluster is formed by a set of nodes gathered around a node which represents them, named cluster head. The choice of the cluster head is done according to some QoS defined criteria.

In the literature, several clustering approaches were proposed. They generally differ on the cluster head selection criterion. In our proposal, we present a clustering approach that elects a reduced number of cluster heads having a low mobility.

The security in networking depends, in many cases, on proper key management. The key management service must ensure that the generated keys are securely distributed to their owners. Any key that must be kept secret has to be distributed so that confidentiality, authenticity and integrity are not violated. For instance whenever symmetric keys are applied, both or all of the parties involved must receive the key securely. In public-key cryptography the key distribution mechanism must guarantee that private keys are delivered only to authorized parties. The distribution of public keys need not preserve confidentiality, but the integrity and authenticity of the keys must still be ensured. Also, we propose in this paper a novel solution for key management in ad hoc networks based on a clustered MANET architecture.

This paper is organized as follows: in Part II, we'll present an overview of the OLSR standard protocol. Part III discusses in more detail our clustering proposal in which we will show the results obtained from the simulations that we perform. Finally, in part IV, we'll present our idea for key management.

2. The OLSR protocol

The optimized link state routing (OLSR) protocol [1] is a proactive routing protocol that employs an efficient link state packet forwarding mechanism called *multipoint relaying*. Optimizations are done in two ways: by reducing the size of the control packets and also by reducing the number of links that are used for forwarding the link state packets. The reduction in the size of link state packets is made by declaring only a subset of the links in the link state updates. The subset neighbors that are designated for link state updates are assigned the responsibility of packet forwarding are called *multipoint relays*. The optimization by the use of multipoint relaying facilitates periodic link state updates. The link state update

[1] http://www.ietf.org/html.charters/manet-charter.html

Manuscript received February 5, 2010
Manuscript revised February 20, 2010

mechanism does not generate any other control packet when a link breaks or when a link is newly added. The link state update optimization achieves higher efficiency when operating in highly dense networks. The set consisting of nodes that are multipoint relays is referred to as *MPRset*. Each given node in the network elects an *MPRset* that processes and forwards every link state packet that this node originates. Each node maintains a subset of neighbors called *MPR selectors*, which is nothing than the set of neighbors that have selected the node as a multipoint relay. A node forwards packets that are received from nodes belonging to its *MPRSelector* set. The members of both *MPRset* and *MPRSelectors* keep changing over time. The members of the *MPRset* of a node are selected in such a manner that every node in the node's two-hop neighborhood has a bidirectional link with the node.
The selection of nodes that constitute the *MPRset* significantly affects the performance of OLSR. In order to decide on the membership of the nodes in the *MPRset*, a node periodically sends Hello messages that contain the list of neighbors with which the node has a bidirectional link. The nodes that receive this Hello packet update their own two-hop topology table. The selection of multipoint relays is also indicated in the Hello packet. A data structure called *neighbor table* is used to store the list of neighbors, the two-hop neighbors, and the status of neighbor nodes. The neighbor nodes can be in one of the three possible link status states, that is, unidirectional, bidirectional, and multipoint relay.

3. The clustering solution

3.1 Clustering in ad hoc networks

Clustering consists in grouping the nodes into clusters (groups) where one node in each cluster functions as clusterhead, responsible for some tasks. Clusters are used for different targets, we distinguish [14]:

- Clustering for transmission management: Clustering provides a mutual organization of network nodes that simplifies coordination of transmission among neighboring nodes. In fact, this technique reduces interference in a multiple access broadcast environment by forming distinct clusters of nodes in which transmissions can be scheduled in a contention free manner by using, for example, different spreading codes in adjoining clusters. Each cluster contains a clusterhead, one or more gateways and zero or more ordinary nodes. The clusterhead schedules transmission and allocates resources within the clusters while gateways connect adjacent clusters. Generally, all cluster members are within one hop of the clusterhead and hence within two hops of each other. This arrangement provides low delay paths between cluster members that may communicate frequently and it places clusterheads in the ideal location to coordinate transmissions among their cluster members.

- Clustering for backbone formation: In any network, the delay incurred by a packet at each hop is a function of the processing and queuing delays at the transmitting node and the transmission and propagation delays over the link. Thus, in a multihop network, reducing the number of hops in a route may significantly reduce the end to end delays experienced by packets traversing the route. Routing backbones consisting of small numbers of long range links are frequently employed to provide low delay, high speed connectivity between distant nodes in large networks. Thus, in ad hoc networks, reduced-hop backbone topologies can be formed by clusterheads which enables direct communication with a more distant node, but it may also increase interference because the node's transmissions will be received at higher power and by a large number of nodes. Thus, it is better to isolate local transmissions within a cluster from distant ones along the backbone.

- Clustering for routing efficiency: Ad hoc networks are known by their dynamically changing topology leading to frequent routes discovery and maintenance. Clustering reduces significantly the overhead costs imposed by routing without scarifying the quality of the routes produced. In addition, a node moving in the same cluster without entering in an overlapping zone doesn't make any problem since it doesn't affect the cluster structure. That means that the entries of both routing tables and neighbor tables won't be modified. Moreover, each node is localized in a single cluster by the correspondent clusterhead. This minimizes considerably the number of entries in the routing tables. Finally, routing could be accomplished via backbones, leading to more efficient routing algorithms. Thus, the clustering technique facilitates network management and ensures the best assets for this management (adaptability, scalability, autonomy, heterogeneity, survivability and economy).

3.2 Density based clustering

The network can be considered as a set of areas (or clusters). Each cluster is formed around a representative called Cluster Head. Cluster Heads are selected according to a well defined criterion.
A cluster is designated by an identifier that relates to its representative (i.e. its cluster head). Each node in the network carries the cluster identifier to which it belongs.

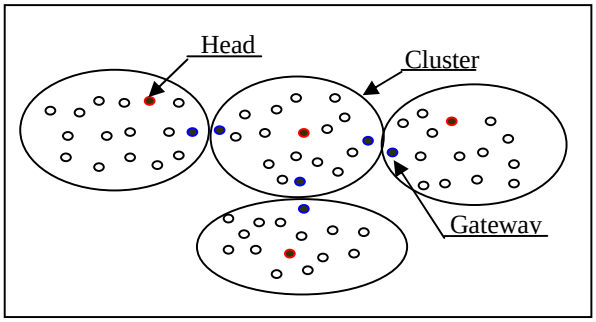

Figure 1: structure of a clusterized MANET

As shown in figure 1, nodes at the cluster border act as communication gateways and ensure the exchange between clusters.

- **Selection criterion of the cluster heads**

In the literature, several studies have addressed the problem of clustering in MANETs. To form clusters and elect cluster heads, each solution provides a different criterion.

In [3], the authors propose a routing protocol based on clusters. To elect the cluster heads, the algorithm selects nodes having the weakest identifier which is nothing but its IP address. But it's not because a node has a small identifier, it's suitable to act as a cluster head.

In [4], authors propose a clustering mechanism for the OLSR protocol. They introduce the concept of forest and tree. The entire network is seen as a forest, where each cluster is considered like a tree and the branches represent the links between nodes. To select a root of the tree, the algorithm uses maximum local connectivity, i.e. nodes having more neighbors are designated as roots. In order to enable OLSR nodes to form and maintain trees, OLSR nodes need to periodically exchange branch messages (in addition to usual OLSR control messages).

In [5], authors propose a hierarchical OLSR version. The hierarchy is built based on nodes capabilities. The capability of a node depends on the amount and properties of its wireless interfaces. A node with several interfaces and large radio range will be selected as cluster head. If the network nodes have the same wireless interfaces properties, the routing finds the OLSR standard operation and there will be no clustered structure. To form clusters, a new message called CIA (Cluster Id Announcement) is periodically sent by cluster heads to declare their leadership and invite other nodes to join their clusters.

Our proposal presents a simple, light and quiet solution. First, our proposal does not add any new control message and the network is not overloaded or slowed at all. No changes are made to standard control messages. Our solution works transparently with the OLSR standard protocol. Clusters are formed around the nodes with the densest environment; in other words, the node that has the largest number of symmetric neighbors is selected as the cluster head.
In this way, we are sure that the cluster is represented by the node that covers the largest number of nodes in the cluster. Thus we call "density of a node i" which is denoted Di, the number of symmetric neighbors of node i.

- **Density computation: Di**

Density information is carried in the structure of HELLO messages. To calculate the density of a node, we use the information contained in the HELLO message. These are periodically sent by each node in the network, and they contain the state information links with all neighbors.
Upon receiving a HELLO message, we can calculate the density of transmitting node *i* (*Di*); in this way each node can decide either to join a cluster or to become a cluster head.

- **OLSR clustering algorithm**

In a clustered OLSR network, each node can be in one of three states:
State 0: not decided. When a node has just arrived, or it has just left its cluster and has no neighbors in its neighborhood, its status is not decided yet. There is no cluster head or cluster member. It must wait for the receipt of HELLO messages.
State 1: Cluster head. The node was exchanged HELLO messages, and it has the highest density. It creates a cluster in which it was appointed head of the cluster.
State 2: member. The node has exchanged HELLO messages; it has a low density compared to its symmetric neighbors, and is part of the cluster members.
Upon receiving a HELLO message, each node calculates the density of the neighbor who sends this HELLO message. Then it compares the neighbor's density with its own density to decide whether to become a cluster head or join the neighbor's cluster.

Transitions between these states are illustrated by the diagram in Figure 2.

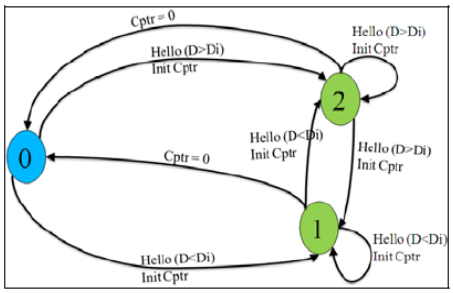

Figure 2: state diagram of a node i

- Initially, each node begins with a status 0 (not decided). Upon receiving a HELLO message, the node compares its own density (*Di*) with the density of the message it received (*D*).
- If (*D* <*Di*), the node goes to state 1 (cluster head) because its density *Di* is greater than *D* of the received message.

Once in state 1, node *i* triggers a counter *Cptr*. If after passing this timeout, the node *i* has received no HELLO message, that means it has no neighbors in its radio range, so it decides to move to state 0 (not decided state).

- If (*D*> *Di*), the node goes to state 2 (member) because its density *Di* is lower than that of the received message.

Once in state 2, node i triggers a counter *Cptr*. If after passing this timeout, the node *i* has received no HELLO message, that means it has no neighbors in its radio range, so it decides to move to state 0 (not decided state).

- If the node *i* is in state 1 (respectively in state 2), and it receives a HELLO message with (*D* <*Di*) (respectively (*D* > *Di*)), it remains in state 1 (respectively remains in state 2) because its state has not changed.
- If the node *i* is in state 1 (respectively in state 2), and it receives a HELLO message with (*D* >*Di*) (respectively (*D*< *Di*)), it moves to state 2 (respectively move to state 1) because its condition has to change.

- **System stability**

We note that the system may become unstable after receiving several Hello messages. A node may change either its state or its cluster whenever the density of the received message is greater than its own density. This may cause some instability in the clustering approach.
To prevent this phenomenon, we chose to keep the node to decide its status (i.e. head or member) for a longer time than the period of a HELLO message.
For simulations, we have taken a period equal to three times the emission range of Hello messages. This time, which we call clustering interval, represents the interval at which each node restarts the process of density calculation.

3.3 Simulations

To see the behavior of this approach and to measure the effect that will cause the implementation of our algorithm in an OLSR network, we performed several simulations with variable number of nodes and different nodes velocity.
We performed simulations with, and without clustering interval. By after we have recorded the average number of clusters built (which we note NC) and the average time during which a cluster is maintained (which we note CD).

- **number of clusters formed by the number of nodes in the network**

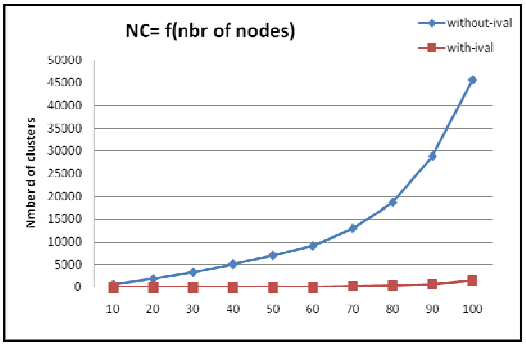

Figure 3 : NC = f(nbr nodes) . velocity = 10m/s

Figure 3 shows the evolution of the number of clusters in relation to the number of nodes in the network for a maximum speed of 10 m /s.
We notice a great improvement with the use of the clustering interval. The number of clusters varies between 500 and 45000 in the case where the clustering interval is not used, when this number varies between 35 and 1500 with the use of clustering interval for a network with 100 nodes as shown in figure 4. This figure shows the same information in figure 3 but at different scale.

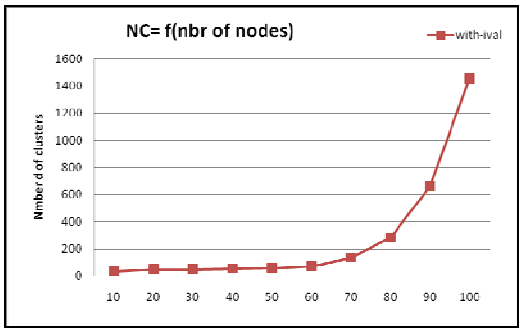

Figure 4 : NC = f(nbr nodes) . velocity = 10m/s

- **average cluster duration based on the number of nodes in the network**

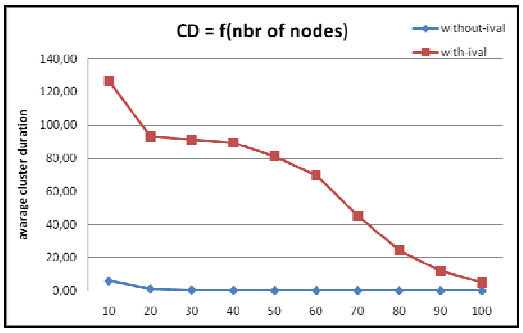

Figure 5 : cluster duration = f(nbr nodes) . velocity = 10m/s

Figure 5 shows the behavior of the average time during

which a cluster is built based on the number of nodes in the network. We notice a significant improvement brought by the clustering interval. The average duration of clusters varies between 0.08ms and 5.9ms in the case where the clustering interval is not used, when this number varies between 4.80ms and 126.67ms with the use of clustering interval for a network with 100 nodes as shown in figure 6.

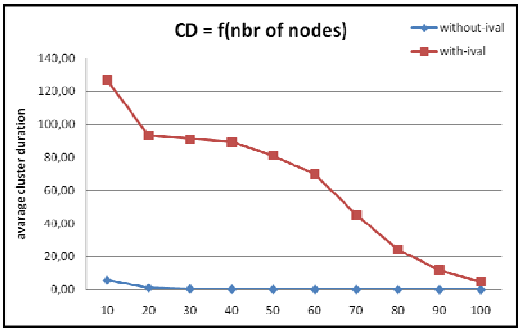

Figure 6 : cluster durat = f(nbr nodes) . velocity = 10m/s

- **number of clusters formed based on the nodes velocity**

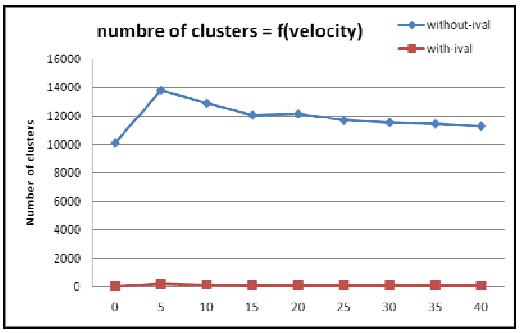

Figure 7 : NC = f(v) . with 70 nodes

Figure 7 shows the evolution of the number of clusters formed according to velocity in the network. The number of nodes in the network is fixed at 70.
We notice a great improvement with the use of clustering interval. The number of clusters turns around 12000 when clustering interval is not used, when this number is around 100 when clustering interval is used as shown in figure 8.

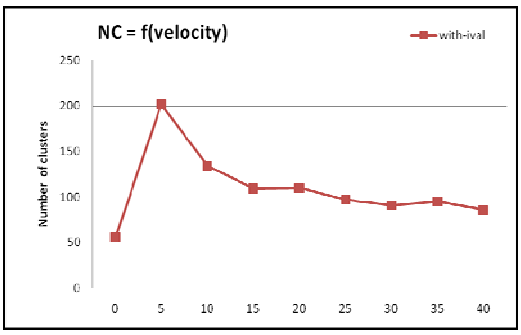

Figure 8 : NC = f(v) . with 70 nodes

- **average cluster duration based on the nodes velocity**

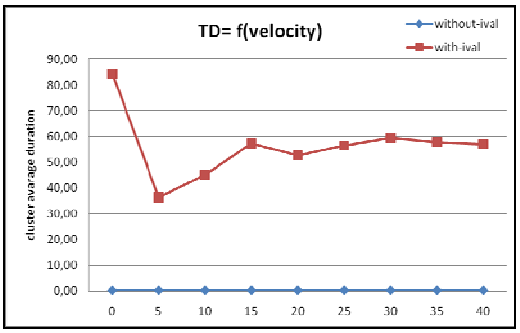

Figure 9 : cluster duration = f(velocity) . with 70 noeuds

Figure 9 shows the behavior of the average time during which a cluster is built based on the maximum speed of nodes in the network. The number of nodes in the network is 70.
We notice a significant improvement given by the clustering interval. The average does not exceed 1 ms in the case where the interval clustering is not used, when it is around 55 ms in the case where the range of clustering is used.

3.4 Impact of Mobility models

The performance of ad hoc network protocols can change significantly when tested with different mobility models, but also when the same mobility model is used with different parameters. Moreover, the choice of a model requires a traffic pattern, which also influences protocol performances. The performance of an ad hoc network protocol should be assessed with the closest mobility model to the real scenario expected, which may facilitate protocol improvement.
To evaluate the performance of our clustering algorithm, we performed simulations for four types of mobility models: Random waypoint, RPGM, Manhattan and Gauss Markov models.
The most popular mobility model proposed in the literature for modeling the MANET scenarios is perhaps the Random Waypoint (RWP) model. A node in the RWP model selects a random destination and a random speed between minimum speed and maximum speed, and then moves to the selected destination at the selected speed. Once the node reaches the destination, the node rests for some pause time, and then repeats the process by selecting a new destination, speed and resuming movement [10].
The Reference Point Group Mobility (RPGM) Model [11] is a typical group mobility model. In RPGM model, each node in a group has two components in its movement vector: the individual component and the group component. The individual component is based on the

Random Waypoint (RWP) model. A node randomly picks a destination within the group scope and moves towards that destination at a fixed speed. Once the node reaches the destination, it selects another destination randomly and moves towards it after a pause time. This behavior is repeated for the duration of the simulation. The group component of mobility is shared by all nodes in the same group and is also based on the random waypoint model. In this case, however, the destination is an arbitrary place in the entire system. Because the RPGM model is based on RWP model, it still cannot overcome the shortcomings caused by the characteristics of the RWP model, such as non-uniform network density, and it is not adequate to simulate the group movement in reality, such as group split and mergence, etc.

The Manhattan mobility model is proposed to model movement in an urban area [12]. In the Manhattan model, the mobile node is allowed to move along the horizontal or vertical streets on the urban map. At an intersection of a horizontal and a vertical street, the mobile node can turn left, right or go straight. The probability of moving on the same street is 0.5, the probability of turning left is 0.25 and the probability of turning right is 0.25. Manhattan mobility model focuses on nodes moving along horizontal or vertical streets, which is not enough to model nodes moving along non-horizontal and non-vertical streets. Moreover, Manhattan model is not suitable to model the movement happening in the intersections of highway systems, which is much more complex than the intersection of local streets.

The Gauss Markov mobility model was proposed in [13]. It is a memory model, in the sense that the position and velocity of a node at any instant (t + at), depend on the position and velocity at time t, which creates more movement flexible nodes. It is a memory model, i.e. the node position and velocity at any instant (t + at), depend on the position and velocity at time t, which creates a more flexible nodes movement. The position (x, y) and the mobile speed S are updated at each timeslot. To ensure that a node does not stay near simulation edges, nodes are pushed away from the board when they are within a certain distance from the edge

To observe our algorithm behavior, we retook the simulations for the four mobility models, and we obtained the following results.

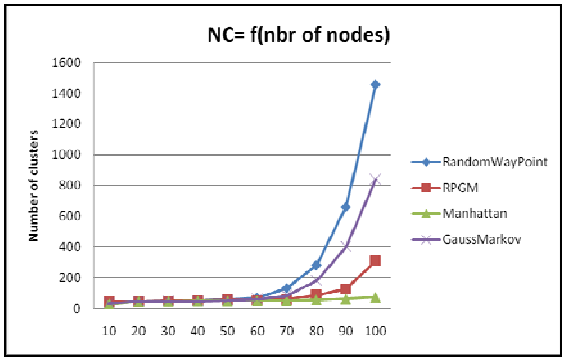

Figure 10 : nbr clusters = f(nbr nodes) . velocity = 10m/s

Figure 10 shows number of clusters formed along simulations in terms of number of nodes in the network. We note that our clustering solution gives best results with Manhattan model.

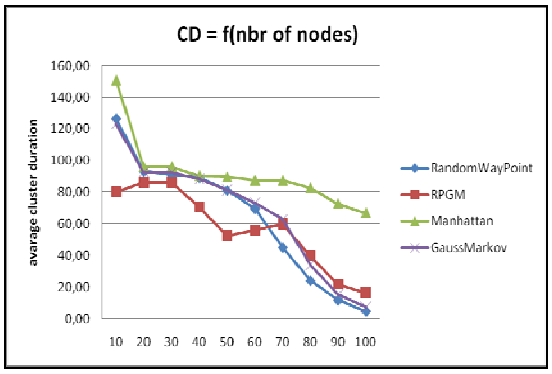

Figure 11 : clusters duration = f(nbr nodes). velocity = 10m/s

Figure 11 shows clusters duration in terms of the number of nodes in the network. We note that clustering behavior is practically the same for all models, except the Manhattan model wich gives the best duration..

3.5 Algorithm enhancement

- **Clustering interval enhancement**

As we have already seen, the clustering interval represents the period during which a cluster is maintained. We chose for this interval a period equal to three times the interval of Hello messages emission. According to the simulation results of the previous sections, we note that the clustering algorithm appears much more stable with a clustering interval of three times the Hello interval. And to see the algorithm behavior with other values of clustering interval, we made measurements for intervals of 6 times and 9 times of the Hello messages transmiting interval.

Results are as shown in following diagrams.

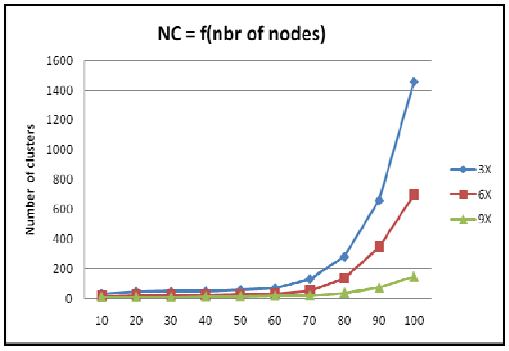

Figure 12 : nbr clusters = f(nbr of nodes) . velocity = 10m/s

Figure 12 shows the number of clusters formed during the simulation based on the number of nodes in the network. Nodes speed is lesse than 10 m / s. We note that the number of clusters decreases when interval clustering becomes important, which proves that our algorithm behaves well with clustering interval changes. To improve performance for dense networks; we propose to make an automatic choice of the clustering interval.

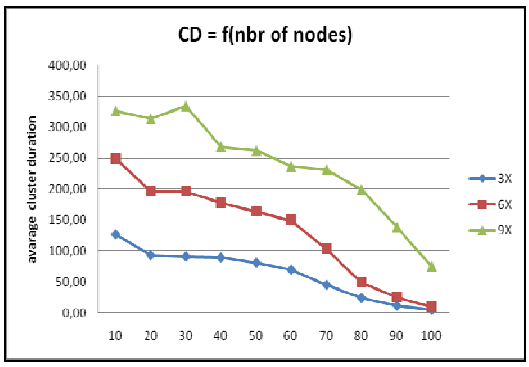

Figure 13 : cluster duration= f(nbr nodes). velocity = 10m/s

Figure 13 shows the lifetime of a cluster based on the number of nodes in the network; speed of nodes does not exceed 10 m / s. We note that clusters remain for a longer time for interval of 9X than for intervals of low values.

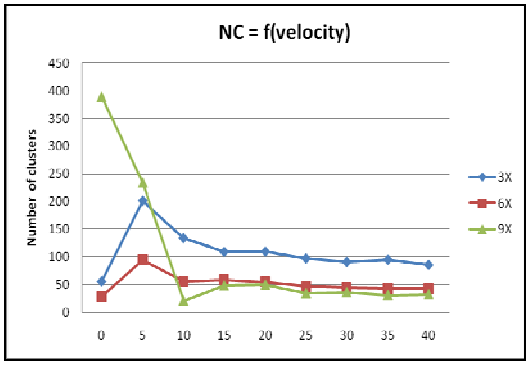

Figure 14 : nbr clusters = f(velocity). with 70 nœuds

Figure 14 shows the behavior of the clustering system for a network of 70 nodes with speeds ranging from 1 m /s to 40 m /s. the diagram shows the number of clusters formed during the simulation depending on the speed of nodes.
Obviously, we still note that the interval of 9X gives better results compared to low values of clustering interval.

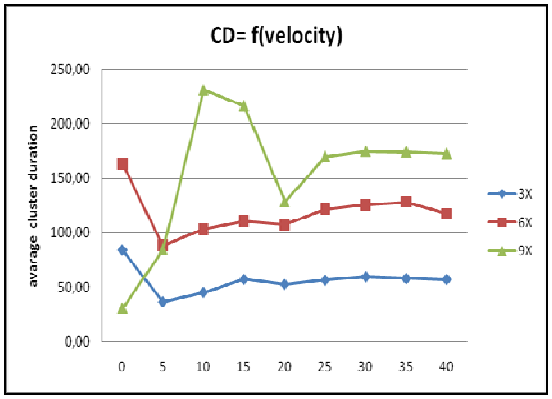

Figure 15 : cluster duration= f(velocity). with 70 nodes

In Figure 15 we observe the behavior of the clustering algorithm in a network of 70 nodes with speeds ranging from 1 m /s to 40 m /s. This figure gives the time during which a cluster is maintained depending on speed of nodes in the network. Again, the range of 9X gives better results compared to low values of clustering interval.

Improvements

According to results presented in this section, we note that the clustering system, in general, behaves almost the same ways for networks with low density (number of nodes less than 60). But from 70 nodes, the curves begin to diverge, and the difference becomes remarkable.
This means that the number of clusters formed during the simulation increases when the number of nodes in the network also increases. And conversely, how long has maintained a cluster decreases when the number of nodes in the network increases.
To control this behavior, we propose an intrinsic management to the protocol, which automates the control and the choice of the clustering interval. Thus, the protocol monitor the number of nodes in the network, and according to this parameter can choose the most suitable interval to maximize the clustering algorithm performance. This improvement will be addressed in a near future work.

- **Clusters depth enhancement**

In our clustering solution, clusters are built around nodes with the densest neighborhood, i.e. node that has the highest number of one hop symmetric neighbors is elected as cluster head. In this way, the cluster head is represented by the node that covers the largest number of nodes in the

cluster. So we called "*density of a node i*", the number of one hop symmetric neighbors wich is denoted *Di.* The network will thus be divided into clusters of depth equal to 1 hop. Choosing the one hop neighborhood, will cause a large number of clusters in the network. To improve this parameter, we chose to deepen the level of clusters depth by choosing two hops symmetrical neighborhoods. The election of cluster heads will focus on nodes with the densest neighborhood of 1 and two hops.

We note (*D2i*) the sum of the number of one hop symmetric neighbors and two hops symmetric neighbors from node *i.* The choice of the head of the cluster will focus on the node with the highest value of *D2*.

We have implemented this improvement on an OLSR network and as shown in the following graphs, we obtained better results.

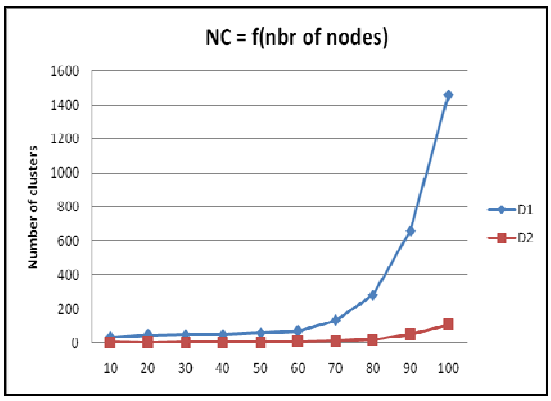

Figure 16 : nbr clusters = f(nbr nodes) . velocity = 10m/s

In Figure 16, we observe the number of clusters (NC) formed during the simulation based on the number of nodes in the network; the maximum speed of nodes is 10 m /s. We note a very good improvement of the clustering system in the case of deep level 2 (D2) where the number of clusters formed is around 10 clusters, while it is around 50 to a depth of level 1 .

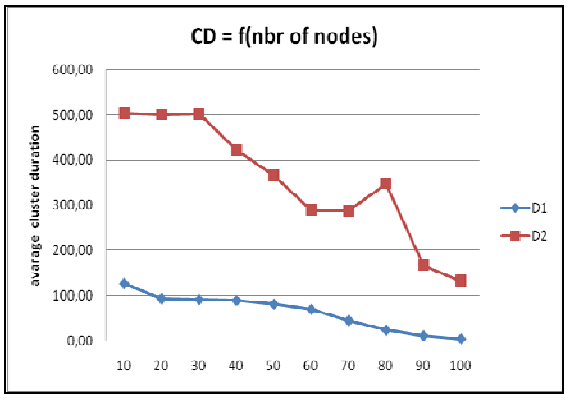

Figure 17 : cluster duration = f(nbr nodes). velocity = 10m/s

Figure 17 shows the average lifetime of a cluster based on the number of nodes in the network, the maximum speed of nodes is 10 m /s. We see a great improvement for this parameter with a depth of level 2. This figure shows that clusters last much longer with a depth of level 2 than those with a depth of level 1.

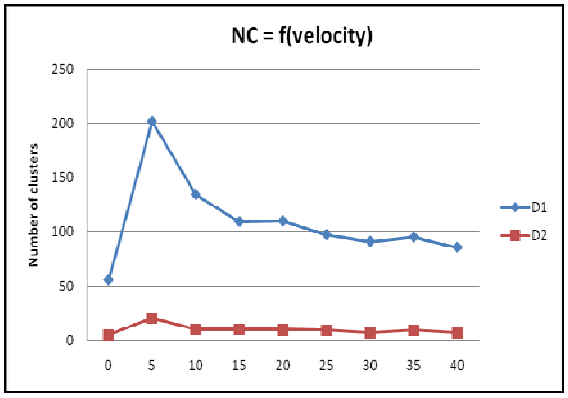

Figure 18 : nbr clusters = f(velocity). with 70 nodes

Figure 18 shows the number of cluster formed during the simulation depending on network speed of 70 knots. It is clear from the graph that, for the same number of nodes in the network, the number of clusters formed during the simulations to a depth of level 2 is much less than number of clusters for a depth of level 1.

In Figure 19, we observe the average time during which a cluster is built for different speeds ranging from 1 m /s to 40 m /s with a network size of 70 nodes. We note that with a depth D2, the average length of a cluster is 7 times greater than depth D1.

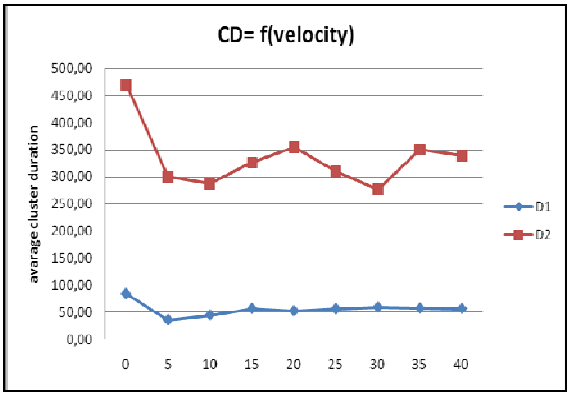

Figure 19 : cluster duration = f(velocity). with 70 nodes

The results presented in this section, we conclude that the transition from one hop neighborhood to a neighborhood with two hops, has remarkably increased the performance of our clustering algorithm. The number of clusters formed in the network has decreased by a factor of 10, and the lifetime of a cluster has increased by a factor of 10. Therefore, clustering system performance has almost

increased by a factor of 10.
In a forthcoming work, we propose to pass from a depth of level 2 to a three level range. We therefore propose to increase the size of a cluster to reach the three hops neighborhood while keeping criterion election of cluster heads, the densest two hops neighborhood.

4. Key management scheme

As in any distributed system, in ad hoc networks the security is based on the use of a proper key management system. As ad hoc networks significantly vary from each other in many aspects, an environment-specific and efficient key management system is needed.

The security in networking depends, in may cases, on proper key management. Key management consists of various services, of which each is vital for the security of the networking systems. The services must provide solutions to be able to answer the following questions: Trust model, Cryptosystems, Key creation, Key storage and Key distribution [15].

4.1 The proposed solution

Approaches presented in the literature tried to solve key management problem in ad hoc networks, but these solutions still carry many limits (administrator availability and congestion, dependence of nodes on the administrator and so on). In this section, we are going to describe the approach that we propose for key management in ad hoc networks. Our solution is based on the clustering technique and is inspired from the partially distributed PKI solutionand uses a (k,n) Threshold Secret Sharing Scheme.

- (K,N)Threshold Secret Sharing Scheme

In secret sharing scheme, a secret is shared among a group of users called shareholders. The secret is shared in such a way that no single user can deduce the secret from his share alone and in order to construct the secret, one need to combine a sufficient number of shares. Adi Shamir [16] proposed a classical (k,n) secret sharing algorithm based on polynomial interpolation. The scheme describes how a secret S can be divided in to n partial shares (S1,S2,...,Sn) where a minimum of k out of n are partial shares are needed to generate a secret S. The threshold value k is balance point between fault tolerance and service availability. Asmuth and Bloom [17], Brickell [18], and Karin-Greene-Hellman [19] have enhanced this work. Also, work has been done in the issues related to verifiable secret sharing [20] and verifiable secret redistribution [21].

4.2 Description of Scheme

Once clusters are formed, and heads are designated, as described in above section, we propose in this section a scheme in which we gather the cluster heads services of cluster heads in a single service called Council. Each Council node will have equal functionality and utilize the (k,n) threshold secret sharing scheme for performing the cluster head functionality. The main functionality of Council will be key management. A certificate will be formed by participation of at least k nodes out of n Council member. The key management cluster head functionality will now be able to work even when more than one (but limited to min {k,n-k+1})cluster heads are compromised.

In our scheme, we propose a novel scheme that we call as Council based clusters. The scheme uses a collaborative approach to perform Council based clusters functionality throughout the network, making it extremely efficient. Once the Council based clusters are formed, each Council member can apply (k,n) threshold secret sharing such that a minimum of k cluster heads out n needs to participate together to perform any cluster head functionality. For example, for key distribution functionality, Council members (each serving as CA) will have a partial secret share and at least k such partial secrets will be required to form the secret. By having multi-cluster heads, each having partial shared secret, the network will be able to work even when more than one (but limited to min {k,n-k+1}) cluster heads are compromised. The requirement for the Council nodes is that they must be fully connected, i.e., each of them must have bi-directional links to all other nodes in the Council.

- Key Management Scheme on Council Based Cluster

Key management is an important aspect of ad hoc network security. To ensure security using public key cryptography scheme, each node carries a public-private key pair and a certificate issued by the CA. As discussed earlier, one of cluster head functionality can be to function as the CA. A CA certifies that a public key belongs to a particular entity. Having a single centralized CA is not suitable for highly vulnerable ad hoc networks. Using our scheme, the process of certification can be distributed among all Council nodes within each cluster. Each cluster will have a public/private key pair. The public key of the cluster is known to each Council member while only a share of cluster private key (also known as secret)is known to each member. Council issues a certificate to a member node‘s public key by digitally signing it with the private key of the cluster. In order to construct the private key of the cluster, at least k Council members out of the n need to work together and combine their partial shares. Since at least k among n partial shares of the private key

are required to generate the cluster private key, system will work even if more than 1 but limited to min (k,n-k+1)Council members are compromised.

- Why Limited to Min (K, N-K+1) Compromised Cluster Heads

In the above section we have mentioned that the cluster head functionality will be able to work even when more than one but limited to min {k,n-k+1}cluster heads are compromised. Let us discuss why our (k,n)threshold scheme is limited to min {k,n-k+1}. In (k,n)secret sharing scheme, a minimum of k cluster heads out n needs to participate together to perform any cluster head functionality. If k or more cluster heads are compromised than they can combine their secret share together to perform any compromised cluster head functionality. Thus the total number of compromised nodes cannot exceed k-1. Also in order to perform cluster head service we require at least k non-compromising cluster heads; the system will not if number of compromised cluster heads are equal to or greater than n-k+1. In general our (k, n) secret sharing scheme will work for any T compromised cluster heads where 1< T < min {k,n-k+1}. For ex. in (5, 12) secret scheme, the system will not work for 5 or more compromised cluster heads as minimum of 5 compromised cluster heads can participate together to perform any cluster head functionality. The (7,12)scheme will not work if 6 or more cluster heads are compromised, as minimum of 7 cluster heads are required for making the decision.

- Finding (K, N)

We have also addressed the problem of choosing a suitable (k,n) pair on Council based clusters. The whole network not being uniformly distributed makes the choice of (k,n) difficult. We find the value of n in an adaptive fashion depending on the availability in the networks. In short the number of Council members per cluster will give us the value of n. The threshold value k is a balance point between fault tolerance and service availability. Lets us discuss the special cases of choosing k:

• k =1: The secret is shared by n nodes and anyone of them can get the secret using just 1 share. This scheme is similar to single cluster head and hence vulnerable to single point of failure.

• k =n: The secret is shared by n nodes and all these nodes need to participate together with their shares in order to get the secret. This scheme provides maximum security but requires accessibility to all the nodes. For highly secure network like military applications, we will choose k =n and apply (n,n) threshold secret share concept on Council.

• 1<k <n: W e chose such a k such that there is a balance between security and availability.

5. Conclusion and perspectives

The clustering mechanism allows dividing ad hoc network into several zones. The solution we propose in this work enables clustering OLSR networks without causing changes in the structure of control messages. Therefore, to make our algorithm more stable, we added the concept of clustering interval which represents the interval at which each node starts the calculation of densities. According to the results of simulations that we made, we notice a great improvement and better system stability with the adopted solution.

To evaluate the proposal performance, we also measured the behavior of our algorithm with several mobility models.

As an initial improvement of our algorithm, we measured its performance for different intervals of clustering, and we propose to automate clustering interval. This interval will be self-adjustable according to the nature of nodes and their behavior in the network.

The second improvement is done by increasing the clusters size to minimize their number. Thus, to calculate the density of a node, our solution considers all nodes in the one hop symmetric neighborhood in addition to nodes in the two hop neighborhood, and thus we have reduced the number of clusters in the network.

In a forthcoming work, we propose to pass from a 2 level depth to a three level range. We therefore propose to increase the size of a cluster to reach the three hops neighborhood while keeping criterion election of cluster heads, the densest two hops neighborhood.

Also, we project to add other suitable criteria to select the best cluster heads; because it is not enough that a node has the densest neighborhood for being elected as cluster head. Thus, in the next version of our algorithm, the criterion for electing cluster heads will focus on a system metric that will engage density, energy and the radio range of each node in the network.

Finally, for the key management scheme, we plane to implement our theoretical idea to evaluate the system behavior with a complete solution for key management in an ad hoc environment.

REFERENCES

[1] T. CLAUSEN ET P. JACQUET. Optimized Link State Routing Protocol (OLSR).http://www.ietf.org/rfc/rfc3626.txt, 2003, RFC 3626

[2] Qayyum, L. Viennot, A. Laouiti, "Multipoint Relaying: An Efficient Technique for Flooding in MobileWireless Networks," INRIA Research Report RR-3898, March 2000.

[3] M. L. Jiang, J. Y. Li, and Y. C. Tay, " Cluster Based Routing Protocol (CBRP) Functional Specification." draft-ietf-manet-cbrp-spec-01.txt, Aug. 1999.

[4] E. Baccelli. "OLSR Trees: A simple Clustering Mechanism for OLSR." Mediterranean Workshop on Ad-Hoc Networks (MED-HOC-NET), Porquerolles, France, June 2005.

[5] Y. Lacharite, M. Wang, P. Minet, T. Clausen. " Hierarchical OLSR " draft-lacharite-manet-holsr-02.txt July 13, 2009

[6] J.Y.Yu and P.H.J.Chong, "A survey of clustering schemes for mobile ad hoc networks," IEEE Communications Surveys & Tutorials, vol.7, no.1, pp.32–48, 2005.

[7] Francisco J. Ros, Pedro M. Ruiz "Cluster-based OLSR extensions to reduce control overhead in mobile ad hoc networks" IWCMC '07: Proceedings of the 2007 international conference on Wireless communications and mobile computing. August 2007. Honolulu, Hawaii, USA

[8] E. Baccelli, T. Zahn, J. Schiller. "DHT-OLSR" Published in INRIA Research Report RR-6194, May 2007.

[9] E. Baccelli. "OLSR Scaling with Hierarchical Routing and Dynamic Tree Clustering". Published in International Conference on Networks and Communication Systems (NCS), Chiang Mai, Thailand, March 2006.

[10] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, edited by T. Imielinski and H. Korth, Chapter 5, pp. 153-181, Kluwer Publishing Company, 1996.

[11] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A Group MobilityModel for Ad hoc Wireless Networks. In Proceedings of the ACM/IEEE MSWIM'99, pp.53-60, Seattle, WA, August.1999.

[12] Fan Bai, Narayanan Sadagopan, Ahmed Helmy. IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks. Infocom'03, April 1-3, 2003, San Francisco, California, USA.

[13] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, Vol. 2, No. 5. (2002), pp. 483- 502, 2002.

[14] C. E. Perkins, "Ad hoc networking", Addison-Wesley Pub Co, 1st edition December 29, 2000.

[15] Kärpijoki Vesa, .Security in Ad Hoc Networks., Telecommunications Software and Multimedia Laboratory 2002

[16] A. Shamir, "How to Share a secret", Communication of the ACM, Vol. 22, pp. 612-613, November 1979.

[17] C. Asmuth anf J. Bloom, "A Modular Approach to Key Safeguarding", IEEE Trans. On Information Theory, IT-29, pp. 208-211, 1983

[18] E. F Brickell, "Some Ideal Secret Sharing Schemes", Journal of Combinatorial Mathematics and Combinatorial Computing, No. 6, pp. 105-113, 1989.

[19] E. D. Karnin, J. W. Greene, and M. E. Hellman, "On Secret Sharing Systems", IEEE Trans. On Information Theory, IT-29, pp. 35- 41, 1983

[20] T. P. Pederson, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing ", Lecture Notes in Computer Science, pp. 129-140, 1992

[21] Y. Desmedt and S. Jajodia, "Redistribution Secret Shares to New Access Structures and its Applications", Technical Repport ISSE TR-97-01, George Mason University, Fairfax, VA, July, 1997.

[22] Vivek Shah, "Parallel Cluster Formation for Secured Communication in Wireless Ad hoc Networks", work submitted as part of requirement for the degree of Master of Science in Computer Science. University of Cincinnati. 2004