CATP: An Enhanced MANETs Clustering Algorithm Based on Nodes Trusts and Performances

Mohamed DYABI, Abdelmajid HAJAMI, Hakim ALLALI

Abstract— A mobile ad hoc network (MANET) is a wireless network without the support of any fixed infrastructure. Security is one of the main challenges in ad hoc network due to dynamic topology and mobility of nodes.Organizing mobile nodes into manageable clusters can limit the amount of secure routing information. Under a cluster structure, mobile nodes are managed by nodes called cluster heads. The clusterhead role is resource consuming since it's always switched on and is responsible for the long-range transmission, for example to send a bit over 10 or 100 m distance, Manet's nodes consume resources that can perform thousands to millions of arithmetic operations. In this work, we present a clustering algorithm based on node trust and performances called (CATP) , where the clusters are formed around the trustworthy , the densest and the most powerful nodes.

Index Terms— Adhoc, Clustering , OLSR , trust.

I. INTRODUCTION

A mobile ad hoc network (MANET) is a collection of wireless mobile hosts that form a temporary network without the aid of any centralized administration or support. In such a network, each mobile node operates not only as a host but also as a router. Several Ad hoc routing protocols proposed by the MANET working group at IETF[1] make flat routing. That means that there is no hierarchy and all terminals have the same role. Clustering is a promising approach for building hierarchies and managing keys in mobile ad-hoc network environments. The main objective of clustering is to identify suitable node representatives called cluster heads (CHs) that can act as a local coordinator for its cluster.

The cluster head role is resource consuming since it's always switched on and is responsible for key generation, key distribution, and key maintenance. If a node has this role, it would burn it resource quickly, and after it died, all its members would be headless.

In this article, we present a clustering approach for efficient, scalable and secure clustering of MANETs. Our proposal consists at the first part on forming clusters around the nodes with the highest performances resource and the densest environment; in other words, the node that has the best material resources and the largest number of symmetric neighbors is elected as the cluster head. In general, it is assumed that all nodes behave according to the application and protocol specifications. This assumption, however, may be false, due to resource restrictions (e.g., low battery power) or malicious behavior. Assuming a perfect behavior can lead to unforeseen pitfalls, such as low network efficiency, high resource consumption and vulnerability attacks. To address this problem, the second part of our proposal improve the algorithm by adding the trust metric of each nodes to the performance computation, our aim is to elect the trustworthy, the densest and the most performance node in the network to act as clusterhead. This paper is organized as follows: in Part II, we will present an overview of the OLSR standard protocol. Part III presents the clustering solution. Part IV provides an overview of related work in the field of cluster based mobile ad-hoc networks. Part V describes the details of our proposed algorithm. Part VI, presents the threshold of performance. Part VII, presents the CATP algorithm and finally the part VIII concludes the article and draws directions for future work

II. THE OLSR PROTOCOL

The optimized link state routing (OLSR) protocol [2] 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 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.

Manuscript Received on June, 2014

Mohamed DYABI, Laboratories LAVETE, University of science and Technology Settat, Morocco

Abdelmajid HAJAMI, Laboratories LAVETE, University of science and Technology Settat, Morocco

Hakim ALLALI, Laboratories LAVETE, University of science and Technology Settat, Morocco

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.

III. THE CLUSTERING SOLUTION

Clustering is the most popular method developed to provide resource management over mobile ad hoc networks. This technique based on partitioning the network in smaller and manageable groups, each group called cluster [2].

Clustering offers, several benefits when it used with MANETs [3-8] listed as follows:

- Provides hierarchical architecture.
- Performs key management
- Helps to perform more efficient resource allocation
- Enhances routing process and mobility.

The purpose of a clustering algorithm is to produce and maintain a connected cluster. In most clustering techniques, nodes are selected to play different roles according to a certain criteria. In general, three types of nodes are defined [9] as shown in Figure 1:

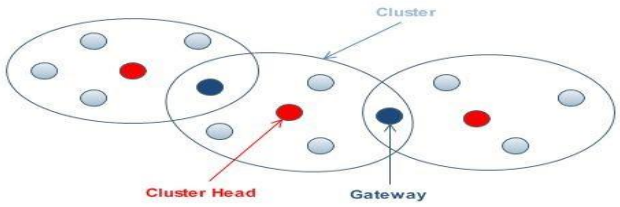

Fig. 1. Nodes Types

A. Cluster head:

Can defined as a local coordinator for its cluster. It performs key management, data forwarding and many other operations. Clusterheads are analogous to the base station concept in current cellular systems. However the difference of a clusterhead from a conventional base station resides in the fact that a clusterhead does not have special hardware, it is selected among the set of stations and it presents a dynamic and mobile behavior . Since clusterheads must perform extra work with respect to ordinary nodes, they can easily become a single point of failure within a cluster. For this reason, we suggest that the clustering election process should consider for the clusterhead role, those nodes with a higher performance.

B. Gateway:

Gateway nodes are nodes in a non-clusterhead state located at the periphery of a cluster. These types of nodes are called gateways because they are able to listen to transmissions from another node which is in a different cluster [10]. To accomplish this, a gateway node must have at least one neighbor that is a member of another cluster.

C. Cluster Member:

Ordinary nodes are members of a cluster, which do not have neighbors belonging to a different cluster [11].

IV. RELATED WORK

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 criteria. In [12], the authors propose a routing protocol based on clusters. To elect the cluster heads, the algorithm selects nodes having the weakest identifier.But it's not because a node has a small identifier, it's suitable to act as a cluster head. In [13], the 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. 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 [14], the 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. 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. In [15], authors uses location information for cluster formation: the highest degree node in a neighborhood, is elected as CH. Experiments demonstrate that the system is not scalable: as the number of nodes in a cluster is increased, a gradual degradation in the system performance is observed. Moreover, in highly mobile environments, the re-affiliation rate increases due to node movements and as a result, the highest-degree node may fail to be re-elected even if it loses a single neighbor. 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 most powerful nodes, i.e, the node that has the best material resources such as residual energy, free memory, processor speed and hard disk space is elected as cluster head. Our algorithm takes into account the node range by including in our calculation the node density.

Therefore, we are sure that the cluster head is represented by the most powerful and the less mobile node in the network that can perform its roles in the best conditions.

V. CLUSTERING ALGORITH BASED ON PERFORMANCE (CAP)

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 criteria. 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.

A. Node Performance computation: Perfi

To calculate the performance of a node, our clustering algorithm uses several metrics, including:
Residual energy, free memory, processor speed, disk space and node density.
To determine the weight associated with each metric we used a multi-criteria analysis method [16]

a. Multi-criteria analysis method:

Multi-Criteria Decision Analysis, or MCDA, is a valuable tool that can be applied to many complex decisions.
It can solve complex problems that Include qualitative and/or quantitative aspects in a decision-making process.

b. Why use multi-criteria analysis in performance assessment:

The performance of a node is calculated based on a number of criteria that the list is not exhaustive. So far we have identified five: autonomy, density, RAM, CPU and Hard Disk associated with each node.
The global performance of the node is obtained by adding the partial performances (criteria) affected by relative weights.
In decision analysis, this operation is called synthesis or additive aggregation.
Regarding the assessment of the relative weights of the criteria, there are several Multi-criteria Decision Analysis methods. We selected Rank Order Centroïde (ROC) [18] for its simplicity and its proven efficiency.

B. Rank Order Centroid (ROC)

Several methods for selecting weights, including equal weights (EW) and rank-order centroid (ROC) weights, have been proposed and evaluated [19–21].
A common conclusion of these studies is that ROC weights appear to perform better than the other rank-based schemes in terms of choice accuracy.
This method is a simple way of giving weight to a number of items ranked according to their importance. The decision-makers usually can rank items much more easily than give weight to them.
The centroid method assigns weights as follows, where w1 is the weight of the most important objective, w2 the weight of the second most Important objective, and so on

$$D = \left\{ W_1 \geq W_2 \geq \ldots \geq W_m \geq 0 \text{ et } \sum_{j=1}^{m} W_j = 1 \right\}$$

This method takes those ranks as inputs and converts them to weights for each of the items.

The conversion is based on the following formula:

$$W_j = \frac{1}{m}\left(\frac{1}{j} + \frac{1}{j+1} + \ldots + \frac{1}{m} \right)$$

C. Calculation of weight by the classification rank order centroid:

Step 1: Sort criteria in descending order of importance

RAUT > RDENS > RRAM > RPRO > RHDD

Step 2: Fill the matrix

	RAUT	RDENS	RRAM	RPRO	RHDD	Control
R1	1,00	0,00	0,00	0,00	0,00	1,00
R2	0,50	0,50	0,00	0,00	0,00	1,00
R3	0,33	0,33	0,33	0,00	0,00	1,00
R4	0,25	0,25	0,25	0,25	0,00	1,00
R5	0,20	0,20	0,20	0,20	0,20	1,00
AVG	0,46	0,26	0,16	0,09	0,04	1,00
						1,00

Step 3: Provide weights

	RAUT	RDENS	RRAM	RPRO	RHDD	C
W	0,46	0,26	0,16	0,09	0,04	1,0

The column control (C) ensures that all weights are normalized (sum of weights = 1)
After this work, the formula becomes:

RPERF = 0, 46 * RAUT + 0, 26 * RDEN + 0, 16 * RRAM + 0, 09 * RPRO + 0, 04 * RHDD

D. 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.
- State 1: Cluster head. The node was exchanged HELLO messages, and it has the highest performance. 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 performance compared to its symmetric neighbors, and is part of the cluster members.

Each node calculates its own performance and send it to its neighbors. Upon receiving a HELLO message, it compares the neighbor's performance with its own performance to decide whether to become a cluster head or join the neighbor's cluster.

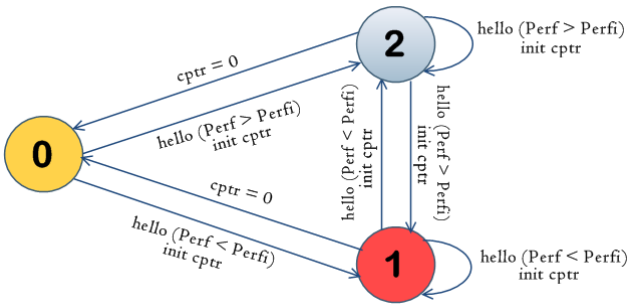

Fig. 2. Clustering algorithm

• Initially, each node begins with a status 0 (not decided). Upon receiving a HELLO message, the node compares its own performance (Perfi) with the performance of the message it received (Perf).

• If (Perf <Perfi), the node goes to state 1 (cluster head) because its performance Perfi is greater than Perf 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 (Perf> Perfi), the node goes to state 2 (member)

because its performance Perfi 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 state2), and it receives a HELLO message with (Perf <Perfi) (respectively (Perf > Perfi)), 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 state2), and it receives a HELLO message with (Perf >Perfi) (respectively (Perf< Perfi)), it moves to state 2 (respectively move to state 1) because its condition has to change.

E. 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 performance of the received message is greater than its own performance. 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 performance calculation

F. Simulation Results

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 used NS2 [22] as a network simulator with the following parameters:

TABLE I. NS2 PARAMETERS

Parameter	Value
Simulation area	1000 x 1000
Radio range	250 m
Number of nodes From	10 to 100 by step of 10
Velocity of nodes	From 0 m/s to 50 m/s by step of 5
Simulation time	300 s

We performed simulations with, and without clustering interval and we have recorded the average number of clusters built (which we note NC) and the average time during which a cluster is maintained

a. Performance of Cluster Head based on the number of nodes

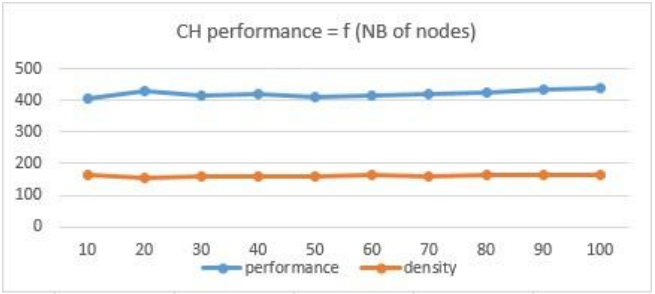

Fig. 3. Performance of CH = f (nb of nodes), V = 10 m/s

To approve the efficiency of our algorithm, we compared it with another algorithm in the literature, which is the algorithm of clustering based on density.

We notice that the performances of the clusterhead in our proposal are much more important than in the algorithm based on density. In our algorithm the performance of the CH varies between 407 and 439, while in the algorithm of clustering based on density it varies between 158 and 164, 5.

b. Number of clusters formed based on the number of nodes in the network

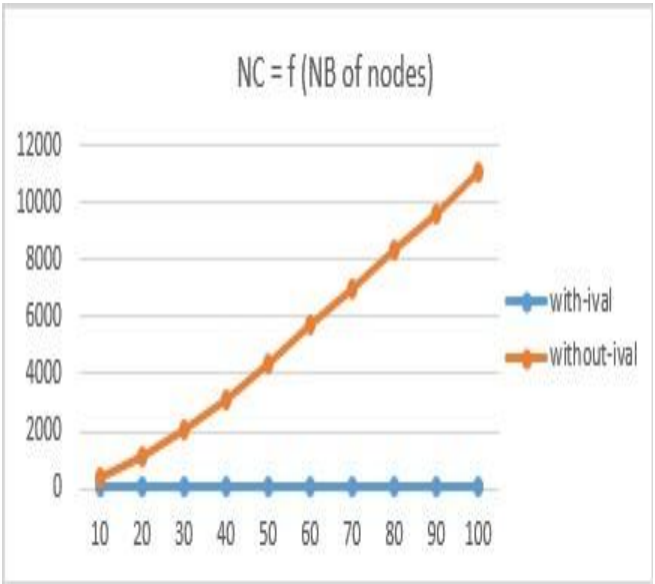

Fig. 4. Number of Cluster = f (nbr nodes), V = 10m/s

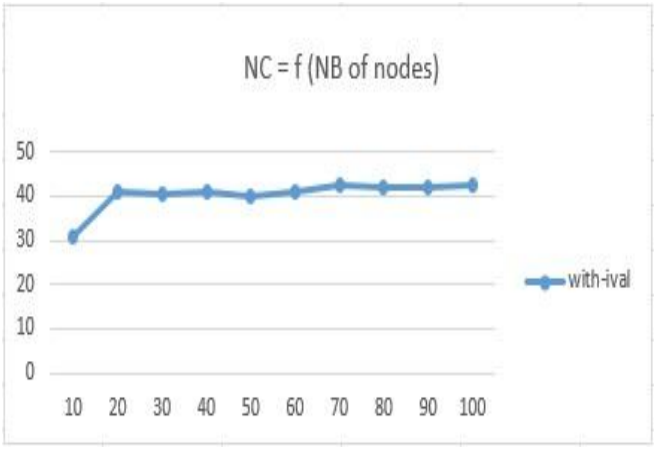

Fig. 5. Number of Cluster = f (nb nodes) , V = 10m/s

This figure shows the same information in figure 4 but at a different scale.

c. Average cluster duration based on the number of nodes in the network

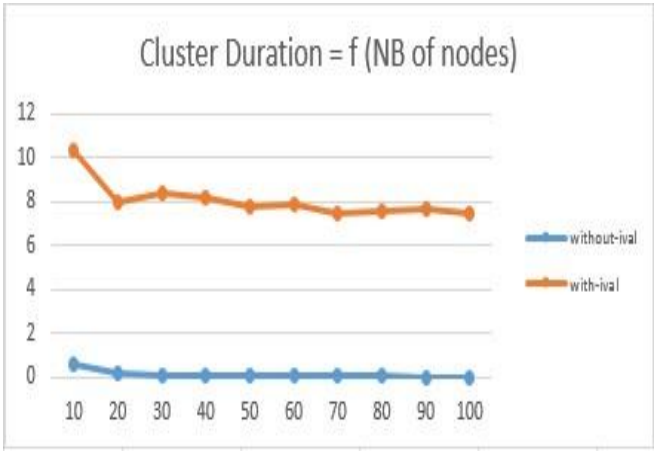

Fig. 6. Cluster duration = f(nbr nodes) , V = 10m/s

Figure 6 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 0.005 ms and 0.62 ms in the case where the clustering interval is not used, when this number varies between 7.45ms and 10.34 ms with the use of clustering interval for a network with 100 nodes as shown in figure 7

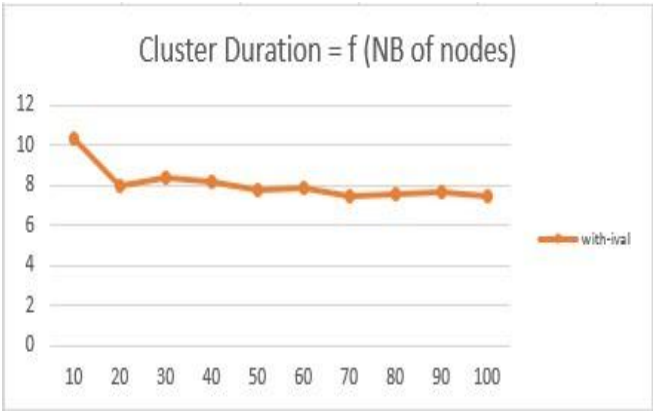

Fig. 7. Cluster duration = f (nbr nodes), V = 10m/s

d. Performance of CH based on the node velocity

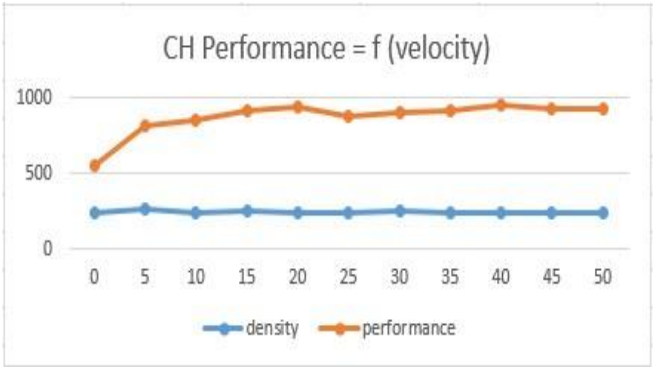

Fig. 8. Performance of CH = f (velocity) , V = 10 m/s

Figure8 shows the performance of clusterhead according to the velocity in the network.

We notice that the performances of the clusterhead in our proposal are much more important than in the algorithm based on density. In our algorithm the performance of the CH turns around 850, while in the algorithm of clustering based on density it turns around 350.

e. Number of clusters formed based on the nodes velocity

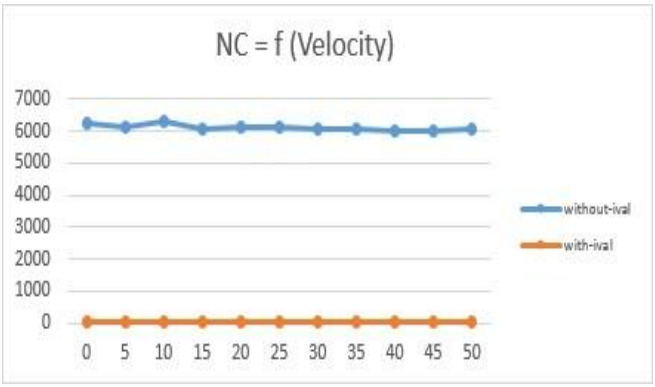

Fig. 9. Number of Clusters = f (velocity) 70 nodes

Figure 9 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 6000 when clustering interval is not used, when this number is around 48 when clustering interval is used as shown in figure 10

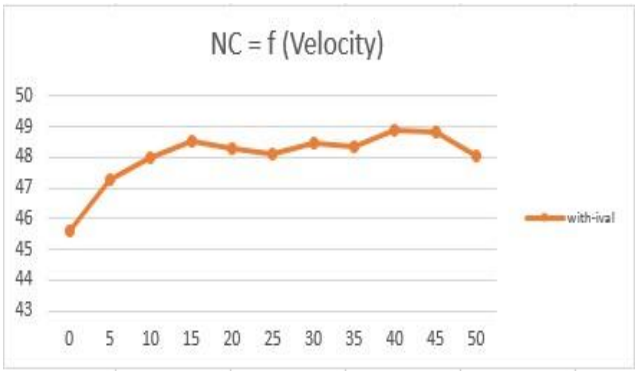

Fig. 10. Number of Clusters = f(velocity),nodes = 70

f. Average cluster duration based on the nodes velocity

Fig. 11. Cluster duration = f (velocity) 70 nodes

Figure11 shows the behavior of the average time during which a cluster is built based on the maximum speed of the 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 turns around 0.01 ms in the case where the interval clustering is not used, when it is around 7 ms in the case where the interval of clustering is used.

VI. THRESHOLD OF PERFORMANCE

We Notice that the system becomes a little bit stable after the application of the interval of clustering. However, even if we apply the interval of clustering, a node can change its status or its cluster if the performance of the received message is bigger than its own performance while it always have the adequate performances to play the role of the clusterhead. To resolve this problem we suggest applying a threshold of performance. Therefore, even if the node receives a bigger performance than its own performance, it is going to keep its status until it reaches the threshold of performance

A. Number of clusters formed based on the number of nodes in the network

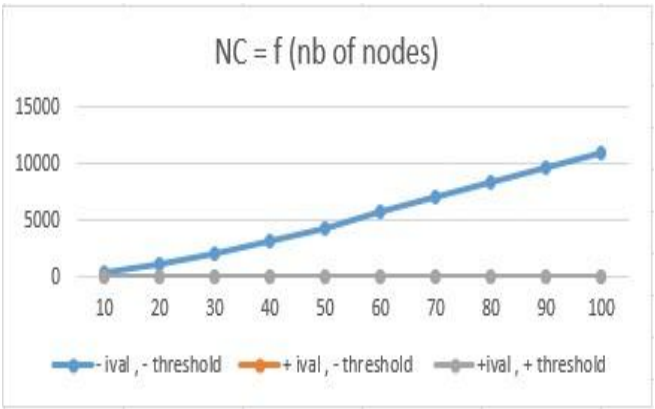

Fig. 12. Number of Clusters = f(nbr nodes), V= 10m/s

We notice a great improvement with the use of the performance threshold. The number of clusters varies:

- between 362 and 11076 : when the clustering interval is not used
- between 30 and 41 : when the clustering interval is used
- between 2 and 19 : when the performance threshold is used with the clustering interval

This figure shows the same information in figure 10 but at a different scale.

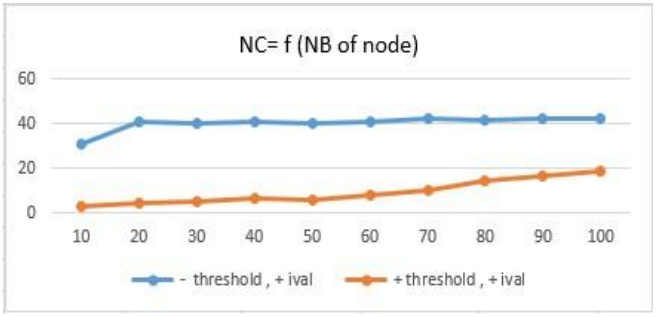

Fig. 13. Number of Clusters = f(nbr nodes), V = 10m/s

B. Number of clusters formed based on the nodes velocity

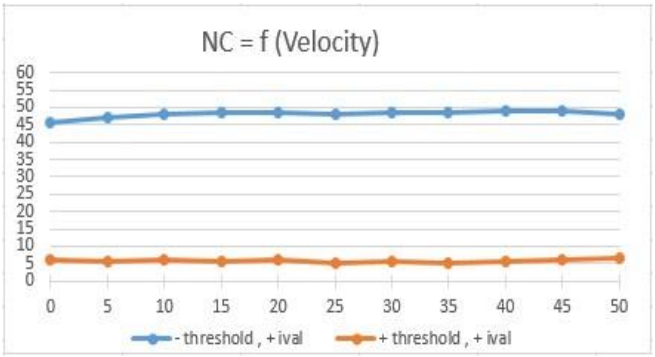

Fig. 14. Number of Clusters= f(velocity), nodes=70

Figure 12 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 the performance threshold. The number of clusters turns around:

- 48 when clustering interval is used
- 6 when the performance threshold is used with clustering interval

C. Average cluster duration based on the number of nodes in the network

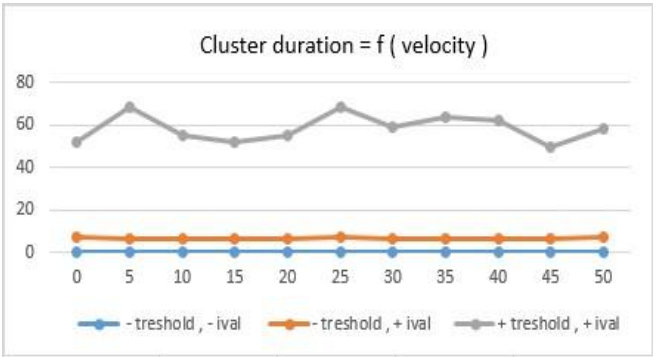

Fig. 15. Cluster duration = f (velocity), noeuds=70

Figure 13 shows the behavior of the average time during which a cluster is built based on the maximum speed of the nodes in the network. The number of nodes in the network is 70. We notice a significant improvement given by the performance threshold . The average turns around:

- 0.01 ms when the interval clustering is not used.
- 7 ms when the interval clustering is used.

- 65 ms when the performance threshold is used with the clustering interval

VII. CLUSTERING ALGORITHM BASED ON TRUST & PERFORMANCE (CATP)

Clustering is one of the main techniques that are used to increase the scalability of MANETs, but without any security considerations clustering is prone to various security attacks[23]. The selfishness is one of the attacks that threaten the functioning of the network, the goal of the malicious node is to use their resources only for their own benefit. In terms of resource consumption, data transmission is the most expensive function in the MANET environment. To send a bit over 10 or 100 m distance, MANETs nodes consume resources that can perform thousands to millions of arithmetic operations [24]. Thus, it may not forward others' packets and simply discard them on purpose. Or they may excessively reduce transmission power to save energy, resulting in network partitioning. Any such feature of behaviour is called selfishness [26]. Currently, most clustering algorithms assume that the network environment is reliable and has no threats [27]. Clusterheads are the key nodes in hierarchical ad hoc networks. If they are attacked, the network performance must decrease seriously. Unfortunately, the first part of our algorithm does not take into consideration the confidence of nodes, and therefore a
malicious node can be elected as clusterhead, thing that will disrupt the functioning of the network. To elect the suitable node to be a cluster head we propose to add the trust metric in our calculation, our aim is to elect the trustworthy, the densest and the most performance node in the network to act as clusterhead.

A. Trust evaluation:

In ad hoc networks, the node process routing control messages and data messages. To calculate the trust metric of a node, our algorithm use several types of messages, including: Hello message, TC message and data messages routed through a node. To determine the weight associated with each type of message we use the Rank Order Centroïde method (ROC)

Step 1: Sort criteria in descending order of importance:

Routed message > TC message > Hello message

Step 2: Fill the matrix

	Routed msg	**TC msg**	**HELLO msg**	**Control**
R1	1,00	0,00	0,00	1,00
R2	0,50	0,50	0,00	1,00
R3	0,333	0,333	0,333	1,00
AVG	0,61	0,28	0,11	1,00
				1,00

Step 3: Provide weights

	Routed msg	**TC msg**	**Hello msg**	**Control**
Weight	0,611111	0,28	0,11	1,00

The column control ensures that all weights are normalized (sum of weights = 1)

After this work, the formula becomes:

RTRST = 0.61 * ROUTEDmsg + 0.28 * TCmsg + 0.11 * HELLOmsg

B. The overall performance computation:

After evaluating the trust metric, we can improve the security of our algorithm described earlier by adding the new metric in the computation of the overall performance of a node.View the importance of trust metric, we will place it in the first rank when calculating the overall performance of a node:

Step 1: Sort criteria in descending order of importance:

RTRST > RAUT > RDENS > RRAM > RPRO > RHDD

Step 2: Fill the matrix

	RTRST	**RAUT**	**RDENS**	**RRAM**	**RPRO**	**RHDD**	**C**
R1	1,00	0,00	0,00	0,00	0,00	0,00	1, 0
R2	0,50	0,50	0,00	0,00	0,00	0,00	1, 0
R3	0,33	0,33	0,33	0,00	0,00	0,00	1, 0
R4	0,25	0,25	0,25	0,25	0,00	0,00	1, 0
R5	0,2	0,2	0,2	0,2	0,2	0,00	1, 0
R6	0,16	0,16	0,16	0,16	0,16	0,16	1, 0
AV	0,41	0,24	0,16	0,10	0,06	0,03	1, 0
							1, 0

Step 3: Provide weights

	RTRST	RAUT	RDENS	RRAM	RPRO	RHD	C
W	0.40	0.24	0.16	0.10	0.06	0.03	1

After this work, the formula becomes:**RPERF = 0,408*RTRST + 0,24 * RAUT + 0,16 * RDEN + 0,10 * RRAM + 0,06 * RPRO + 0,03 * RHDD**

C. OLSR Clustering algorithm:

In this chapter, we propose an improved version of the algorithm presented previously, which takes into account the trust metric of the MANETs node.

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 performance. 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 performance compared to its symmetric neighbors, and is part of the cluster members.

Each node calculates:

- Its neighbor trust
- Its overall performance.

These two information are carried in Hello message.

After receiving Hello message, the node gets its trust metric. Then it can calculate its overall performance. After computing the overall performance the node sends it through the broadcasted Hello message.When the node receives its neighbor nodes' Hello messages, it updates the related nodes' trust value and update its overall performance :Each node in the network calculate its neighbors trust and send it through the hello message.After receiving the hello messages, each node can have a vision of other nodes trust by computing the confidence average of each node.

• If (OPerf < OPerfi), the node goes to state 1 (cluster head) because its performance OPerfi is greater than OPerf 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 (OPerf > OPerfi), the node goes to state 2 (member) because its performance OPerfi 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 state2), and it receives a HELLO message with (OPerf < OPerfi) (respectively (OPerf > OPerfi)), 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 state2), and it receives a HELLO message with (Operf > OPerfi) (respectively (OPerf < OPerfi)), it moves to state 2 (respectively move to state 1) because its condition has to change.

D. Simulation results:

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 We performed simulations with, and without the selfishness attack and we have recorded the average number of clusters built (which we note NC) and the average time during which a cluster is maintained and the average of overall performance of clusterhead.

a. Performance of Cluster Head based on the number of node:

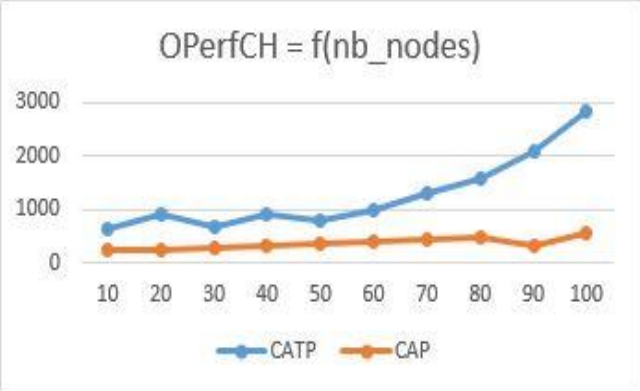

Fig. 16. Overall Performance of CH = f (nb nodes) , V = 10 m/s

We notice that the performances of the clusterhead in CATP are much more important than in the algorithm CAP

The performance of clusterhead varies between 649 and 2831 in the case where the trust metric is not used, when this number varies between 238 and 536 with the use of the trust metric for a network with 100 nodes.

b. number of clusters formed based on the number of nodes in the network

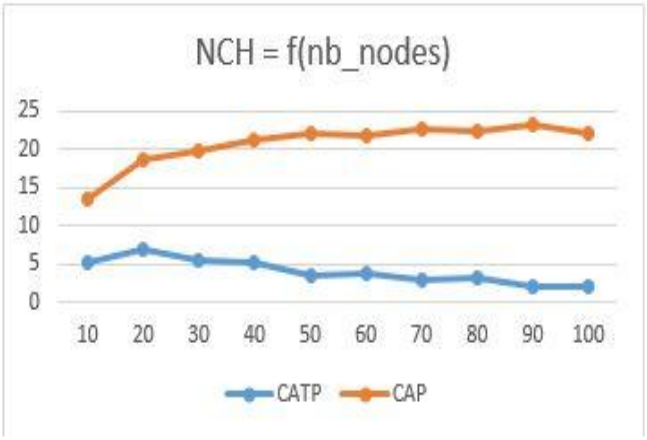

Fig. 17. Number of Clusters = f(nbr nodes), V= 10m/s

Figures 15 shows the evolution of the number of clusters in relation to the number of nodes in the network We notice a great improvement with the use of the trust metric . The number of clusters varies between 13 and 22 in the case where the trust metric is not used, when this number varies between 5 and 2 with the use of the trust metric for a network with 100 nodes.

c. Average cluster duration based on the number of nodes in the network

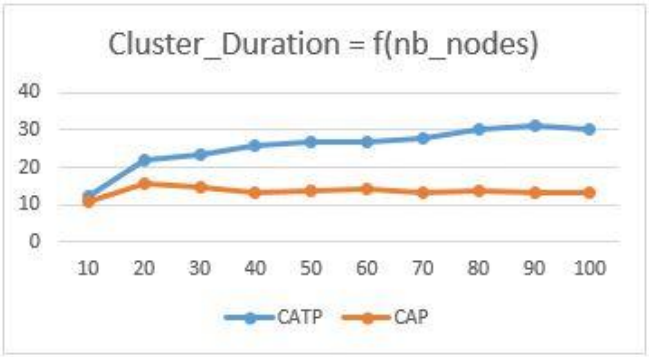

Fig. 18. Cluster duration = f(nbr nodes) , V = 10m/s

Figure 18 shows the behavior of the average time during which a cluster is built based on the number of nodes in the network. We notice that the cluster life is affected by the selfishness attack, but with the use of the trust metric the CATP algorithm is much more resistant to the selfishness attack. The average duration of clusters varies between 10.89 ms and 13.17 ms in the CAP algorithm, when this number varies between 12.44 ms and 30.01 in the CATP algorithm for a network with 100 nodes

VIII. CONCLUSION AND PERSPECTIVES

Clustering is an important research topic for (MANETs) because clustering makes it possible to guarantee basic levels of system performance. A large variety of approaches for ad hoc clustering has been presented. In this work we introduce an algorithm for efficient clustering of mobile ad-hoc networks. Its contributions, compared to existing solutions, are summarized in the following: it 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. It works transparently with the OLSR standard protocol. Clusters are formed around the nodes with the highest performance resource and the densest environment; in other words, the node that has the best material resources and the largest number of symmetric neighbors is selected as the cluster head. Our algorithm takes into account the node reputation by including in our calculation the trust metric. To make our algorithm more stable, we added the concept of the threshold of performance, which represents the performance at which each node can act as cluster head. According to the results of simulations that we made, we notice a great improvement and better system stability with the adopted solution. As perspective to this work, we intend to use the clustering solution to manage cryptographic key in MANETs.

REFERENCES

1. http://www.ietf.org
2. T. CLAUSEN ET P. JACQUET. Optimized Link State Routing Protocol (OLSR).http://www.ietf.org/rfc/rfc3626.txt,RFC 3626
3. S. Sarkar, T. G. Basavaraju, and C. Puttamadappa, Ad Hoc Mobile Wireless Networks: Principles, Protocols and Applications New York: Auerbach Publications, 2007
4. Anju Sharma, Shini Agarwal and Ravindra Singh Rathore "Cluster Based Routing in Mobile Ad hoc Wireless Networks Using Neuro-Genetic Paradigm", International Journal of Scientific & Engineering Research Volume 3, Issue 7, July-2012
5. Dr. Nasib Singh Gill, Swati Atri ,Jaideep Atri "Clustering Approach Based on ant Colony Optimization" International Journal of Advanced Research in Computer Science and Software Engineering , Volume 4, Issue 2, February 2014
6. A. Hajami, K. Oudidi, M. Elkoutbi. "A Distributed Key Management Scheme based on Multi Hop Clustering Algorithm for MANET",IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010
7. Satu Elisa Virtanen and Pekka Nikander. Local clustering for hierarchical ad hoc networks. In Proceedings of WiOpt'04: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pages 404–405, Los Alamitos, CA, USA, 2004.
8. E. M. Belding-Royer, "Hierarchical Routing in Ad Hoc MobileNetworks," Wireless Commun. and Mobile Comp., vol. 2, no.5, 2002, pp. 515–32.
9. Anju Sharma, Shini Agarwal and Ravindra Singh Rathore "Cluster Based Routing in Mobile Ad hoc Wireless Networks Using Neuro-Genetic Paradigm", International Journal of Scientific & Engineering Research Volume 3, Issue 7, July-2012
10. I. Er and W. Seah, "Mobility-based d-hop clustering algorithm for mobile ad hoc networks," in Wireless Communications and Networking Conference, 2004.
11. Karamjeet Singh "energy efficiency in mobile ad -hoc networking using cluster head routing protocol" Vol. International Journal of Advanced Research in IT and Engineering 2 No. 5 May 2013
12. M. L. Jiang, J. Y. Li, and Y. C. Tay, " Cluster Based Routing Protocol (CBRP) Functional Specification." draft-ietfmanet- cbrp-spec-01.txt, Aug. 1999.
13. E. Baccelli. "OLSR Trees: A simple Clustering Mechanism for OLSR." Mediterranean Workshop on Ad-Hoc Networks (MED-HOC-NET), Porquerolles, France, June 2005.
14. Y. Lacharite, M. Wang, P. Minet, T. Clausen. " Hierarchical OLSR " draft-lacharite-manet-holsr-02.txt July 13, 2009
15. M. Gerla, J.T.C.Tsai, "Multicluster, mobile, multimediaradio network", Wireless Networks 1(3), pp. 255, 1995.
16. Roy, B. (2005). An overview of MCDA techniques today: paradigms and challenges. In: Figueira, J., Greco, S. and Ehrgott, M. (eds) Multiple criteria decision analysis: state of the art surveys.
17. Barron, F.H. 1992. Selecting a best multiattribute alternative with partial information about attribute weights.
18. http://www.ncsu.edu/nrli/decision-making/MCDA.php
19. Butler J, Olson DL. Comparison of centroid and simulation approaches for selection sensitivity analysis. Journal of Multi-Criteria Decision Analysis 1999;8:146–61.
20. Jia J, Fischer GW, Dyer JS. Attribute weighting method and decision quality in the presence of response error: a simulation study. Journal of Behavioral Decision Making 1998;11:85–105.
21. StillwellWG, Seaver DA, EdwardsW. A comparison of weight approximation techniques in multiattribute utility decision making. Organization Behavior and Human Decision Processes 1981;28:62–77.
22. Network Simulator NS2 http://www.isi.edu/nsnam/ns/
23. Bidaki, Moazam; Masdari, Mohammad "Reputation-Based Clustering Algorithms in Mobile Ad Hoc Networks." International Journal of Advanced Science & Technology . May2013, Vol. 54, p1-11
24. Sohail Abbas "A Survey of Reputation Based Schemes for MANET" PGNet 2010
25. younghwan yoo and dharma p.agrawal "why does it pay to be selfish in a manet ? " ieee wireless communications december 2006
26. Yao Yu+, Lincong Zhang "A Secure Clustering Algorithm in Mobile Ad Hoc Networks" IPCSIT vol. 29 2012

AUTHORS PROFILE

Mohamed DYABI
Received the Master degree in networks and systems in 2010 from Faculty of Science and Technology HASSAN I University Settat-Morocco.
Currently, he is a PhD Student in Computer Science.
Ongoing research interests:
Security in mobile ADHOC networks (MANETs) QoS in wireless networks Next Generation Networks

Prof.Abdelmajid HAJAMI.
PhD in informatics and telecommunications, Mohamed V-Souissi University Rabat-Morocco. 2011
Ex Trainer in Regional Centre in teaching and training
Assistant professor at the Faculty of Science and Technology of Settat in Morocco
Member of LAVETE Lab at Faculty of Science and Technology of Settat
Research interests:
Security and QoS in wireless networks
Radio Access Networks
Next Generation Networks
ILE : informatics Learning Environments eLearning

Hakim ALLALI
was born in Morocco on 1966. He received the Ph.D degree from Claude Bernard Lyon I University (France) in 1993 and the "Docteur d'Etat" degree from Hassan II-Mohamedia University, Casablanca (Morocco) in 1997. He is currently Professor at Faculty of Sciences and Technologies of Hassan 1st University of Settat (Morocco) and director of LAVETE Laboratory. He is executive manager and founder of IT Learning Campus. His research interests include technology enhanced learning, modeling, image processing, computer networking and GIS