Performance Improvement of Clustered Mobile Ad hoc Networks using a CDMA Single Channel and Based on Admission Control Approach

ZOUHAIR EL-BAZZAL, MICHEL KADOCH, BASILE AGBA AND FRANCOIS GAGNON
Department of electrical engineering
University of Quebec, Ecole de technologie superieure
1100, notre dame west, Montreal, H3C 1K3
CANADA
zouhair.el-bazzal@etsmtl.ca

Abstract: - Clustering has been proven to be a promising approach for mimicking the operation of the fixed infrastructure and managing the resources in multi-hop networks. In this paper, we propose an Efficient Clustering Algorithm (ECA) in Mobile Ad hoc Networks based on the quality of service's (QoS) parameters (cluster throughput/delay, packet loss rate). The goals are yielding low number of clusters, maintaining stable clusters, and minimizing the number of invocations for the algorithm. The performance changes greatly for small and large clusters and depends strongly on the formation and maintenance procedures of clusters which should operate with minimum overhead, allowing mobile nodes to join and leave without perturbing the membership of the cluster and preserving current cluster structure as much as possible. In this manner, while QoS does not perform well under high traffic load conditions, admission control becomes necessary in order to provide and support the QoS of existing members. Based on the results from the proposed analytical model, we implement a new admission control algorithm that provides the desired throughput and access delay performance in order to determine the number of members inside an ECA cluster that can be accommodated while satisfying the constraints imposed by the current applications. Through numerical analysis and simulations, we have studied the performance of our model and compared it with that of WCA. The results demonstrate the superior performance of the proposed model.

Key-Words: - Clustering, Weight, Election, Throughput, Delay

1 Introduction

In recent years, most research is focusing on clustering approaches for multi-hop networks because of its effectiveness in building a virtual backbone formed by a set of suitable clusterheads to guarantee the communications across clusters. An example of multi-hop networks is an ad hoc network characterized by a collection of wireless nodes which communicate with each other using high-frequency radio waves. These nodes arbitrarily and randomly change their locations and capabilities without the aid of any fixed infrastructure. The main objective of clustering is to elect suitable nodes' representatives, i.e. clusterheads (CHs) and to store minimum topology information by reducing the propagation of routing information which facilitates the spatial reuse of resource and increase the system capacity. Each CH will act as a temporary base station within its zone or cluster and communicates with other CHs. Thus, packets for route finding and acknowledgement may only spread among CHs instead of flooding among all nodes. On the other hand, the topology change information caused by movement of some nodes is limited in adjacent clusters, not in all networks. Therefore, any clustering scheme should be adaptive to such changes with minimum clustering management overhead incurred by changes in the network topology.

To establish a cluster, traditional clustering algorithms suggest CH election exclusively based on nodes' IDs or location information and involve frequent broadcasting of control packets, even when network topology remains unchanged. Most recent work takes into account additional metrics (such as energy and mobility) and optimizes initial clustering without taking into consideration the QoS parameters like "cluster achievable throughput, cluster delay and transmissions' number per packet". In many situations, re-clustering procedure is hardly ever invoked; hence initially elected CHs soon waste their batteries due to serving the other members for longer periods of time.

In addition, a topology control mechanism is required to mitigate the vulnerability of such clusters due to node joining/leaving and link failures. It aims to reduce interference and energy consumption, to increase the effective network capacity, and to reduce the end to end delay.

As election of optimal clusterheads is an NP-hard problem [1], many heuristic mechanisms have been proposed. Centralized algorithms rely on the assumption that the elected CH is responsible of the cluster's maintenance. However, these algorithms suffer from single point (CH) of bottleneck especially in highly mobile environments; hence initially elected CHs have to collect excessive amounts of information and soon reach battery exhaustion. On the other hand, distributed algorithms are more adaptive to mobility due to the fact that the maintenance is done in collaboration between all the nodes where each node relies on the local information collected from the nearby nodes. Although the distributed manner is preferred for MANET, it lacks a major drawback in achieving and guarantying a strong connectivity between the nodes.

As the cluster structure is used for our routing purpose, it is not encouraging that the throughput available to each member approaches zero as the number of members increases inside a cluster. Therefore, the size of a cluster in terms of number of members competing on the channel seems to be an important factor which strongly affects the local available throughput, the cluster delay and therefore the whole performance of the clustered network. In this matter, the estimated knowledge of the number of members sharing an 802.11 cluster might effectively drive congestion avoidance on the CHs and inter-clusters load-balancing to achieve better network resource utilization. In order to simplify the maintenance, especially in high mobility scenarios, we investigate an algorithm that generates one-hop clusters.

In this way, the goals of this paper are to maintain stable clusters with a lowest number of clusterheads, to minimize the number of invocations for the clustering formation/maintenance and to maximize the lifetime of mobile nodes in the system. To achieve these goals, we propose an Efficient Clustering Algorithm (ECA) which utilizes factors like the node degree, remaining battery power, transmission power, and node mobility for the clusterheads' election. Our approach differs from others in that it is based on the clusters' capacity and it uses the link lifetime instead of the node mobility for the maintenance procedure. We refer this to the fact that the node mobility metric does not affect the election of a CH as much as the link stability metric does. It also provide an efficient technique to estimate and derivate the number of members inside a cluster with respect to the local allowed saturation throughput and delay when the protocol DCF (Distribution Coordination Function) is used to access the channel and when the members are homogenous in traffic generation. The estimation methodology builds on the existence of a mathematical relationship between the number of competing members, the packet collision probability encountered on the shared medium and the packet arrival rate at each member. The obtained results will help us to readjust the used parameters of the clustering algorithm in order to provide better quality of service guarantees depending on the used applications.

Once the cluster size is calculated with respect to the quality of service required, the simulations results show that the proposed clustering model provides better performance in terms of number of formed clusters, number of re-affiliations, average number of transition (state change) on CHs and number of clusterheads changes when compared to that of other weight based algorithms such as WCA [2].

The paper is organized as follows. In Section 2, we review several approaches proposed previously. Section 3 presents the proposed clustering model. Section 4 presents the proposed admission control model for providing the quality of service's parameters. Section 5 presents the performance analysis of the model. Finally, Section 6 concludes the paper.

2 Overview of Existing Approaches

A large number of approaches have been proposed for the election of clusterheads in mobile ad hoc networks. The Highest-Degree [3] uses the degree of a node as a metric for the selection of clusterheads. The degree of a node is the number of neighbors each node has. The node with maximum degree is chosen as a clusterhead; since the degree of a node changes very frequently, the CHs are not likely to play their role as clusterheads for very long. In addition, as the number of ordinary nodes in a cluster is increased, the throughput drops and system performance degrades. The Lowest-Identifier (LID) [4, 5, 6] chooses the node with the lowest ID as a clusterhead, the system performance is better than Highest-Degree in terms of throughput. However, those CHs with smaller IDs suffer from the battery drainage, resulting short lifetime of the system.

The Distributed Clustering Algorithm (DCA) [7] and Distributed Mobility Adaptive clustering algorithm (DMAC) [8] are enhanced versions of

LID; each node has a unique weight instead of just the node's ID, these weights are used for the selection of CHs. A node is chosen to be a clusterhead if its weight is higher than any of its neighbor's weight; otherwise, it joins a neighboring clusterhead. The DCA makes an assumption that the network topology does not change during the execution of the algorithm. Thus, it is proven to be useful for static networks when the nodes either do not move or move very slowly. The DMAC algorithm, on the other hand, adapts itself to the network topology changes and therefore can be used for any mobile networks. However, the assignment of weights has not been discussed in the both algorithms and there are no optimizations on the system parameters such as throughput and power control.

Instead of static weights, MOBIC [9] uses a new mobility metric; Aggregate Local Mobility (ALM) to elect CH. ALM is computed as the ratio of received power levels of successive transmissions (periodic Hello messages) between a pair of nodes, which means the relative mobility between neighboring nodes.

Least Clusterhead Change Algorithm (LCC) [10] allows minimizing clusterhead changes that occur when two CHs come into direct contact. In such a case, one of them will give up its role and some of the nodes in one cluster may not be members of the other CH's cluster. Therefore, some nodes must become CH while causing a lot of re-elections because of the propagation of such changes across the entire network. Maximum Connectivity Clustering (MCC) [11] is based on the degree of connectivity. A node is elected as CH if it is the highest connected node. This is not suitable in dynamic network topologies where the degree of connectivity changes rapidly.

The Weighted Clustering Algorithm (WCA) [2] is based on the use of a combined weight metric that takes into account several parameters like the node-degree, distances with all its neighbors, node speed and the time spent as a clusterhead. Although WCA has proved better performance than all the previous algorithms, it lacks a drawback in knowing the weights of all the nodes before starting the clustering process and in draining the CHs rapidly. On the other hand, different aspects of the problem of capacity of ad hoc networks were examined in [12, 13, 14, 15], they show that performance is very sensitive to the number of nodes that are simultaneously trying to send a packet on the shared medium. Thus, the throughput of each node declines rapidly while the number of nodes increases.

3 Cluster Formation And Maintenance Procedure

Before discussing the clusters' formation, we state our environmental assumptions:

1. There is a common signaling CDMA code named "sig_code" used by all stations (new arrival nodes, existing nodes and clusterheads) for signaling messages (i.e. node desires to join/leave a cluster, clusterhead admits a node, clusterhead rejects a node, etc.)
2. There is a common CDMA code named "intercluster_code" for intercluster communications. Clusterheads will communicate between them using this unique code.
3. Clusterhead assigns a CDMA code exclusive to the cluster named "intracluster_code". All stations that are accepted as belonging to the cluster must use this code to communicate with the clusterhead. No other stations can interfere with the throughput of these stations. Other stations that will be rejected by the clusterhead must align themselves with other clusterheads or create a new cluster.
4. We assume that the total number of "intracluster_codes" is large, so that neighboring clusters are not assigned the same "intracluster_code". With a large number of codes, it is unlikely for a node to have an interfering node which is on the same channel but associated to another CH and using a different "intracluster_code". Clusterheads also maintain a list of the intracluster_codes used in the neighboring clusters in order to avoid conflicts.

For cluster formation, we allocate IDs for the nodes and the clusterheads. We use the MAC Address as the node ID in order to avoid the conflicts between IDs in the zone. Hence, the node ID (My_MAC) is unique within a cluster; the CH ID is the node ID of the CH (CH_MAC) in the cluster. The CH ID appended with the node ID forms a unique identifier for every node in the ad hoc network. Every node in the cluster will have information about its CH so that it can communicate across the cluster. Finally, the weight parameter is periodically calculated by each node in order to indicate the suitability of a node for playing clusterhead's role.

3.1 Setup procedure and network design

Every cluster has a limited number of nodes which defines its size. It also has a CH for communication across the cluster. The nodes collaborate to select the best CH. A CH must be able to manage its members, to accept or to refuse the adhesion of new arrivals based on its capacity without perturbing the functionality of the other members. A 'Counter' is maintained by each node in order to count the number of nodes inside a cluster and to guarantee that the cluster size does not exceed a parameter "N" in terms of number of nodes by cluster.

The problem is how to estimate "N" with respect to a certain level of QoS, this estimation is invoked and detailed in section 4.2 where we derivate a relationship between "N", the cluster throughput and delay. For now, we can suppose that each node N_i (member or CH) is identified by a state such as: N_i (id_{node}, id_{CH}, intracluster_code, Weight, Counter, N), it also has to maintain a 'node_table' wherein the information of the local members is stored. However, the CHs maintain another clusterhead information table 'CH_table' wherein the information about the other CHs is stored.

The format of these tables is defined as: node_table (id_{node}, Weight) and CH_table (id_{CH}, Weight). In complex networks, the nodes must coordinate between each other to update their tables. The Hello messages are used to complete this role. A Hello contains the state of the node; it is periodically exchanged either between CHs or between each CH and its members in order to update the 'CH_tables' and the 'node_tables' respectively. Before invoking the maintenance procedure, it is important to describe how each node is able to compute its weight and the several metrics taken into consideration. The clusterheads' election is based on the weight values of the nodes. Each node computes its weight value based on the following parameters:

1. The degree difference: defined as the difference between the cluster's size "N" and the actual number of neighbors. It allows estimating the remaining number of nodes that each node can still handle.
2. The actual transmission power of the node.
3. The average speed of the node.
4. The remaining battery power of the node.

These parameters are inspired from those used in WCA [2], except the actual transmission power 'P_i' and the remaining battery power 'E_i'. We focus on P_i instead of the sum of distance used in WCA in order to elect the node which can cover the largest range, thus, minimizing the number of clusters generated. In addition, the E_i factor is a better measure than the cumulative time during which the node acts as a CH that is used in WCA, because it allows to extend the lifetime of nodes by relinquish the role as a CH in case of insufficient battery power. Algorithm 1 shows the steps to calculate the weight value.

- Find the degree d_i of the node i by counting its neighbours;
- Compute the degree difference $\Delta_i = |d_i - N|$ for the node i, where N is a threshold for the cluster's size in terms of number of members;
- Compute the remaining battery power E_i for the node i;
- Compute the actual transmission power P_i of the node i;
- Compute the average speed S_i of the node i until the current time T;

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

  where (X_t, Y_t) is the coordinate of node i at time t;
- Calculate the combined weight W_i of the node i :
  $W_i = a*\Delta_i + b*E_i + c*P_i + d*S_i$
  where a, b, c et d are the weighting factors;

Algorithm 1. Procedure for calculating the weight of node i

The basic idea is to combine each of the above system parameters with certain weighing factors depending on the system needs. The flexibility of changing the weighting factors helps us apply our algorithm to various networks. We suppose that the nodes have the same needs. For example, in low mobility environment, we can privilege the remaining battery parameter, thus the factor 'b' can be made smaller for all the nodes.

On the other hand, we believe that the relative mobility M_i is a better factor than the average speed S_i. In the first stage, we still use S_i instead of M_i because it is impossible to estimate M_i when the node is alone in the zone without any other reference point (neighbors). In the second stage, the re-election is more sophisticated. Therefore, we use the link lifetime metric which seems more realistic than the relative mobility metric to see whether it is worth to re-elect or to continue with the old CH. Initially, each node broadcasts its state (Join with id_{node} = My_MAC) to notify its presence to the neighbors. Each node builds its neighbors' list based on the received states. After that, the election procedure is executed once the topology is stabled, and the node having the lowest weight is chosen as CH.

3.2 New arrival nodes mechanism

Once a wireless node is activated, its id_{CH} field is equal to NULL. Since it does not belong to any cluster. The node continuously monitors the channel until it figures out that there is some activity in its neighborhood. This is due to the ability to receive the signals from other present nodes in the network. The node still has no stable state, thus its state is not full identified. In this case, it broadcasts a

Join_Request using the sig_code in order to join the most powerful clusterhead. Thus, it waits either for a welcome_ACK or for a welcome_NACK. When the entry node does receive neither welcome_ACK nor welcome_NACK, it may increase its transmission power in order to broadcast another Join_Request that may reach the farthest clusterheads. If this persists for certain number of attempts, the node declares itself as an isolated node, readjusts its transmission power and restarts by broadcasting a new Join_Request after a period of time. We note that just the CHs may response by a welcome_ACK or welcome_NACK; the ordinary members have to ignore any Join_Request received even if they are in the transmission range of the new entry node.

In the case where the node receives a response (welcome_ACK or welcome_NACK), it does not take immediately any decision, this allows the node to be certain that it has received all the responses from all the neighboring CHs. The welcome_ACK and welcome_NACK messages do not indicate that the CH has added the node to its table; they just signify that the CH is waiting for a Join_Accept in order to add the node to its table and to assign the used intracluster_code. When the node receives multiple welcome_ACKs, it selects the one which has the lowest weight. After that, it sends a Join_Accept using the sig_code to the chosen clusterhead and waits for CH_ACK from this CH. The CH_ACK has to contain a confirmation that the id_{node} has been added to the CH_table, and the used intracluster_code. Thus the node can fully-define its state. The reason that we use four ways to confirm the joining procedure is to prevent other CHs that they can serve the entry node to add this node to their tables and cause conflicts.

In the case where the node was just receiving welcome_NACKs, it considers these responses as rejection messages from the CHs. This may occur when the CHs are saturated and decide to reject the adhesion of new nodes. Hence, the node may increase its transmission power and count the number of attempts it tries to reach any CH. When the number of attempts reaches a certain value, the node prefers not to stay isolated, thus it declares itself as CH; in this case, the node must search for the welcome_NACK which has the lowest weight, thus it communicates via a CH_Request message with the CH which is the source of the chosen welcome_NACK and waits for a CH_Response.

Table 1 summarizes messages used in the proposed algorithm.

3.3 Clusterhead nodes mechanism

A CH has an id_{node} field is equal to id_{CH} field. As a CH, the node calculates periodically its weight, thus it sends periodically Hello messages to its members and to the neighboring CHs in order to update the node_tables and CH_tables respectively. The CH must monitor the channel for Leave, Hello and Join_Request messages.

When the CH receives a Leave message, it updates the node_table and broadcasts a Hello message to its members and to its neighboring CHs to inform them that its previous 'Counter' was decremented.

When the CH receives a Hello from a neighboring CH, it updates the CH_table. If the Hello's source is a node member, the CH updates the node_table and verifies its weight. In the case of a lowest weight, the CH must invoke the re-election procedure. We restrict this procedure to the CHs in order to simplify the maintenance and the complexity of the cluster management. The re-election does not necessarily mean that a new CH must be elected even if there is a member node having a lowest weight, we will explain in details this procedure in algorithm 3.

When the CH receives a Join_Request (id_{CH} =NULL) from a new arrival node or a Join_Request (full state) from a node which belongs to another cluster, the CH must invoke the merging procedure explained in algorithm 2 in order to admit or to reject the request basing on its capacities, the link lifetime and the available resources. This procedure gives more flexibility to the members by allowing them to leave a weak CH and join another one which seems stronger than the current CH. It may not be possible for all the clusters to reach the cluster size N. We have tried to reduce the number of clusters by merging those that have not attained their cluster size limit.

However, in order not to rapidly drain the clusterhead's power by accepting a lot of new nodes, we define thresholds which allow the clusterhead to control the number of nodes inside its cluster. When the CH receives a Join_Request, it verifies its capacity in terms of maximum number of nodes, then it verifies the ratio of power levels of the successive Join_Request messages received from the requester member, which allows getting good knowledge about the link lifetime metric between the CH and the requester node.

Hence, the clusterhead does not definitively accept the merging until it is certain that the power level of the last received messages from the member is greater than the power level of the first received messages. In this way, the CH is sure that the member is moving closer to it. If not, the CH realizes that the link is going to break and it is no need to add this node in the node_table because it is going to leave soon. Finally, when a CH receives a CH_Request from a node which desires to be CH, it must accept the request by adding the node as a new arrival CH in the topology, send a CH_Response to the node, update CH_table and broadcast a Hello message to the neighboring CHs.

Algorithm 2 shows the merging procedure used to join or to merge multiples nodes within a cluster.

```
Define the threshold of the link durability while storing a historic about the last
Join_Request messages received;
A = The historic indicates that the power signal of the Join_Request decrements rapidely;
B = Perform admission control;
if (A or B is false)
  Send welcome_NACK to the Join request's source;
else
  if (Join_Request is received from a new arrival node) // the state is not identified
    Send welcome_ACK to the source of Join_Request;
    Wait during a time interval for Join_Accept from the new arrival node;
    if (no Join accept)
      Ignore the request;
    end
  end
  //Join_Request is received from an identified member node or a new arrival node
  Counter = Counter + 1;
  Add the node to the node_table;
  Send a CH_ACK to the new added member;
  Update node_table and CH_table;
  Broadcast hello message to the members and the neighboring CHs;
  Perform re-election procedure;
end
```

Algorithm 2. Merging procedure

It is favorable when the CH stays in the cluster for a longer time, as time need not be spent in re-election of a CH frequently. The re-election is not periodically invoked; it is performed just in case of a lowest received weight, it allows minimizing the generated overhead encountered in previous works.

As we explained above, the re-election may not result a new CH, it depends on the stability of the new node for playing the CH's role. In the case where a new CH must be elected, the procedure should be soft and flexible in order not to perturb the clusters while to copying the databases from the old CH to the new CH. We limit the execution of the algorithm where there is a CH or a network change in order not to impact the whole ad hoc topology. Thus the furthest nodes are not affected by any problem which occurs in other clusters; therefore they are up to date about the size's changes of any cluster in the network.

Algorithm 3 shows the re-election procedure used in order to decide whether to elect or not a CH.

```
Define the threshold of the link durability while storing a historic about the last
Join_Request messages received;
A = The historic indicates that the power signal of the Join_Request decrements rapidely;
B = Perform admission control;
Verify periodically the hellos received weights from neighbors;
if (there is a new Weight < Weight (CH))
  if (A or B is false)
    Don't perform the re-election and the CH continues its role;
  else // Prepare to the re-election;
    The old CH stays CH until the reception of a confirmation;
    Send database_info(old CH) to the new elected node;
    Wait during a time interval for a database_ACK() from the new elected CH;
    if (no database_ACK)
      Don't perform the re-election and the CH continues playing its role;
    else
      Store in the id CH field the MAC Address of the new elected node;
      Update node_table and CH_table;
      Send CH_info() to the elected node;
      Broadcast CH_change() to the members and the neighboring CHs;
    end
  end
else
  Don't perform the re-election and the CH continues playing its role;
end
```

Algorithm 3. Re-election procedure

3.4 Member nodes mechanism

Note that after joining a cluster, the node declares itself as a member of this cluster. Hence, it calculates periodically its weight and sends periodically Hello messages to its CH. As a member, this node should just handle the Hello, the welcome_NACK, the CH_info and the database_info messages received from the clusterhead nodes (see table 1). This allows optimizing the resources (bandwidth, battery, etc) and minimizing the job of the nodes.

When the node receives a Hello from its CH, the node has to update its node_table. When the node receives Hellos from the neighboring CHs, the node has the possibility to migrate to another CH if there is a Hello which has a smaller weight than the current CH's weight, it sends a Join_Request to the CH which is Hello's source and continues as a member of the current CH until the reception of CH_ACK. In this case, the node can send a Leave_Request to the last CH. This method allows us to minimize the number of the formed clusters in the network.

When the node member receives a CH_info message as a result of the re-election procedure, thus it realizes that it is going to become the new CH in the cluster. When a node member does not receive any message from its CH, it considers that the CH has gone brusquely down; in this case, the nodes have no choice and must restart the clustering setup procedure.

4 An Admission Control Method for Providing Desired Throughput and Delay Performance

While QoS does not perform well under high traffic load conditions, admission control becomes necessary in order to provide and support the QoS of existing members. We first analyze the cluster throughput and delay performance of DCF based cluster using a unique CDMA code. Specifically, we investigate the impact of active nodes and wireless channel collisions on the performance of the DCF. Based on the results from this analysis, the cluster will be able to admit or reject the nodes by restricting the input traffic being admitted to the system in order to maintain the QoS of already admitted members.

4.1 The Proposed analytical model

We are interested in studying the performance of the system in the worse cases where the clusterhead is serving all nodes that are in its transmission range. Such that, we can derivate the maximum number of members that a clusterhead can serve, thus the cluster size “N” which will be used for the formation of the cluster with respect to the QoS parameters (throughput and delay).

To model the exponential backoff schema implemented in DCF 802.11 MAC layer, we know that for each packet transmission, a node initializes a backoff time which is a random integer uniformly distributed over the interval $(0, W-1)$. The value of W is called contention window, and depends on the number of failed transmissions of the packet. At the first transmission attempt, the value of W is equal to CW_{min} called the minimum contention window.

Let p be the probability that the transmitted packet faces a collision in the channel due to two or more nodes transmitting simultaneously in the same slot. In this case, after each unsuccessful transmission, the value of W is doubled, up to a maximum value $CW_{max} = 2^m CW_{min}$ where m represents the number of unsuccessful attempts for this packet, i.e., the maximum backoff stage. Once W reaches CW_{max}, it keeps this value until it is reset to CW_{min}.

Now, we can derive the general probability that the contention window W that a node chooses and is given by:

$$P\{\mathrm{Window} = W\} = \begin{cases} p^{m-1}(1-p) \text{ for } W = 2^{m-1}CW_{min} \\ \\ p^m \text{ for } W = CW_{max} \end{cases} \tag{1}$$

Tay and all [15] derived the collision probability p for the case of saturated network where a transmitting node always has a queue of packets to send, so each incoming packet is immediately backlogged, i.e. it is preceded by a backoff. At saturation, each packet is backlogged immediately. The average backoff window in the saturated case is given by:

$$(1-p)\frac{W}{2} + p(1-p)\frac{2W}{2} + \cdots + p^m(1-p)\frac{2^m W}{2} + p^{m+1}\frac{2^m W}{2} = \frac{1-p-p(2p)^m}{1-2p}\frac{W}{2} \tag{2}$$

In this paper, we extend the model proposed in [15] to obtain an approximate expression for collision probabilities in case of non-saturated arrival rates. Therefore, we use Poisson data source instead of saturation data source. We consider a cluster with "N" members operating in discrete time where each member could be represented as an M/G/1 queuing system with an infinite storage, the packet arrival process is a Poisson memoryless processes with a rate given by λ packets while the packet service process of the network has general distribution with first moment μ which will be explained in section 4.2.

For such M/G/1 system, the probability that the packets' interface queue is empty could be approximated by the following equation:

$$\pi_0(\mathrm{node}) = 1 - \frac{\lambda}{\mu} \tag{3}$$

A packet is backlogged on arrival if the system is non-empty at the instant of arrival. When an arbitrary arrival occurs, we approximate the probability that the cluster (N members in steady state) is empty by:

$$\pi_0(\mathrm{network}) = \left(1 - \frac{\lambda}{\mu}\right)^N \tag{4}$$

Then, the backoff window is 0 for any arbitrary packet with probability $\pi_0(\mathrm{network})$, and it is backlogged with probability $1 - \pi_0(\mathrm{network})$. Therefore, the average backoff window size for general (non-saturated) arrival rates is given by:

$$\overline{W}_{\mathrm{general\ case}} = \left[1 - \left(1 - \frac{\lambda}{\mu}\right)^N\right]\left[\frac{1-p-p(2p)^m}{1-2p}\frac{W}{2}\right] \tag{5}$$

4.2 Estimation of the optimal number of members and the correspondent throughput

Assuming that at an instant t, an arrival to an idle node will not be backlogged even if there are some other backlogged nodes having non empty queues.

Now, following the arguments explained in [15] and considering the fact that the cluster contains N members and only those with a nonempty queue can actually collide with packets from other nodes. The packet collision probability can be obtained by solving:

$$p = 1 - \left[\left(1-\pi_0(\text{node})\right)\left(1-\frac{1}{\overline{W}_{\text{general case}}}\right)\right]^{N-1}$$
$$= 1-\left(\frac{\lambda}{\mu}\right)^{N-1}\left[1-\frac{2(1-2p)}{(1-p-p(2p)^m W)\left(1-\left(1-\frac{\lambda}{\mu}\right)^N\right)}\right]^{N-1} \quad (6)$$

On the other hand, assuming q the probability that a node transmits in a randomly chosen slot, the probability p that a transmitted packet faces a collision on the channel in a given slot will be equivalent to the probability that at least one of the $(N-1)$ remaining nodes transmits in the same time slot. In other words, if we assume that each remaining node transmits a packet with probability q in the same time slot, thus the probability p that a collision occurs is given by:
$1 - P\,\{\text{none of the } (N-1) \text{ remaining nodes transmits}\}$.
Therefore: $p = 1-(1-q)^{N-1}$.

Let S be the saturation throughput inside the cluster, defined as the expected time needed to transmit the data payload with respect to idle, collision and header transmission time, during a cycle of frame exchange.
$S = \frac{\text{Probability(success tx in a slot)}\times\text{Payload\_size}}{\text{Duration of a cycle of frame exchange}}$.
A cycle of frame exchange consists of several collision cycles and one successful data frame transmission plus header transmission and idle times.

$$S = \frac{Nq(1-q)^{N-1}\times L(\text{DATA})}{\text{Success}_{\text{cycle}}+\text{Collision}_{\text{cycle}}+\text{Idle}_{\text{cycle}}} \quad (7)$$

Where
$$\begin{cases} \text{Success}_{\text{cycle}} = \alpha \times \text{Prob(success transmission)} \\ \quad = \alpha \times Nq(1-q)^{N-1} \\ \text{Collision}_{\text{cycle}} = \beta \times \text{Prob(collision)} \\ \quad = \beta \times [1-(1-q)^N - Nq(1-q)^{N-1}] \\ \text{idle}_{\text{cycle}} = \gamma \times \text{Prob(idle)} = \gamma \times (1-q)^N \end{cases}$$

α represents the time that the channel is captured with a successful node transmission, β represents the collision duration, i.e., the time that the channel is captured by the node with a collision and γ represents the duration time of a time slot. The value of the parameters α and β differs depending on the access model. Assuming that the packets are data fragment only, that means that there is no fragmentation. Thus, for basic access mechanism:

$$\begin{cases} \alpha = \text{DIFS} + \overline{\text{PHY}_{\text{hdr}}} + \frac{\text{MAC}_{\text{hdr}}+\text{DATA}}{\vartheta} + \text{SIFS} + \frac{\text{ACK}}{\vartheta} + 2\varepsilon \\ \beta = \text{DIFS} + \overline{\text{PHY}_{\text{hdr}}} + \frac{\text{MAC}_{\text{hdr}}+\text{DATA}}{\vartheta} + \varepsilon \end{cases} \quad (8)$$

Where $\overline{\text{PHY}_{\text{hdr}}}$ represents the synchronization time between the source node and the destination node, i.e., the transmission time of the PLCP preamble and PLCP header which defines the physical header of a 802.11 packet. ϑ represents the channel bit rate and ε represents the average propagation delay of any frame on the channel. All these parameters are defined in the IEEE 802.11 standard [16]. By resolving equation 7, we obtain the relationship between N and S which is:

$$S = \frac{Nq(1-q)^{N-1}\times L(\text{DATA})}{\alpha Nq(1-q)^{N-1}+\beta[1-(1-q)^N-Nq(1-q)^{N-1}]+\gamma(1-q)^N} \quad (9)$$

4.3 Cluster delay analysis

The cluster delay $\overline{D}$ is defined by the average time from the point when a packet becomes head of the node's queue to the point when it is successfully received by the destination, i.e., when a positive acknowledgment is received. We model this delay without taking into account the waiting time in the packets' interface queue before transmitting. $\overline{D} = \left(\gamma\overline{W}_{\text{general case}} + \alpha\right) + \frac{1}{\mu}$. The service rate μ is expressed via the average time required for successful packet transmission, thus:
$\mu = \frac{1}{\left(\gamma\overline{W}_{\text{general case}}+\beta+\Delta\right)\overline{N}_{cp}}$.

Δ represents the time that a node has to wait when its packet transmission collides, before sensing the channel again. Therefore $\Delta = \text{SIFS} + \text{ACK\_Timeout}$. Owing to the assumption of independence at each retransmission, we can calculate $\overline{N_{cp}}$ which represents the average number of collisions of a packet until it is received successfully, and approximate the average rate of successful transmission per packet $\overline{N_{sp}}$ which follows a geometric distribution having the Esperance $\frac{1}{1-p}$.

Knowing that $\overline{N_{sp}}$ defines the average number of node attempts to successfully transmit its packet, i.e., the node has been exposed to $(\overline{N_{sp}}-1)$ collisions before to successfully transmit its packet; therefore: $\overline{N}_{cp} = \frac{1}{1-p} - 1$. Thus:

$$\overline{D} = \left(\gamma\overline{W}_{\text{general case}} + \alpha\right) + \frac{p\left(\gamma\overline{W}_{\text{general case}}+\beta+\Delta\right)}{1-p} \quad (10)$$

4.4 Numerical analysis and discussion

Numerical values are computed in this section based on the analytical model presented in previous section. The goals are to present the theoretical throughput and delay of each cluster while varying the number of its members and with respect to the parameters specified in IEEE 802.11b specifications [16] which define the frame sizes of the MAC layer and the Frequency Hopping Spread Spectrum (FHSS) physical layer. In the rest of the paper, the channel bit rate ϑ has been assumed equal to 1 Mbps without RTS/CTS; we also vary the number of members "N" inside the cluster between 10 and 50 in order to study the impact on the whole performance.

Figure 1 shows that the throughput strongly depends on the number of members inside the cluster. We realize that, in most cases, the greater is the number of the active members, the lower is the throughput. This condition remains valid until the value of the window "W" is approximately equal to 500. We see that an higher value of 'W' tends to give better throughput performance in the case of large members' number in the cluster, while it drastically penalizes the throughput in the case of small members' number. On the other hand, we neglect the probability of having great values of W in case of small number of members, since the probability of collisions in this case is very small in comparison to the case of large number of members.

This behavior is also seen in figure 2, the amount of channel time wasted in collisions is extremely large for a small value W and a large number of members. Large value of W may, in fact, increase the delay; because for small W, the amount of idle slot times per packet transmission is very low. This value becomes significant only when W gets greater and the number of members is small. When the number of members is large, the throughputs of large windows are very close, but the cluster delay deteriorates much more severely for smaller contention windows.

On the other hand, figure 3 shows that the number of transmissions per packet significantly increases as the window size decreases. This effect is much more significant for large number of members. Therefore, we can settle the fact that the clustering of an ad hoc network cannot be concluded by the randomly execution of an algorithm regardless the specifications of the applications used inside the cluster. By using a sophisticated algorithm like ours (ECA), we can adjust the parameters of this algorithm in order to generate a certain number of clusters which fulfill the requirement of each member in terms of the required throughput, the tolerated delay and the allowed transmission number per packet.

Figure 4 plots the achievable throughput versus the packet size for three different cluster sizes (N=10, 30 and 50). We see that when the maximum channel bit rate ϑ is equal to 1Mbps, the throughput efficiency increases (approaches to ϑ) as the packet size increases. The situation is explained by considering that the time spent for frame transmission is decreased as the data rate increases but the time overhead spent on DIFS, SIFS and the backoff delay remains the same. We can conclude that the choice of a smaller number of members performs better in the whole cluster. We fixed that number to 15 which will be used in the next section to define the cluster size "N" with respect to the QoS parameters.

5 Performance Analysis

5.1 Simulation environment and parameters

Generating appropriate scenario with realistic mobility pattern is very challenging when designing ad hoc networks. Agba and al. [17] have developed a simulation tool including a scenarios generator and a propagation modeler. It allows a complete description of environmental parameters, mobility model parameters and other simulation settings.

The simulations scenarios used in this paper were randomly generated based on 2D - Random Walk model (2D-RWM) using the described simulation tool. Some keys parameters are the number of nodes, the percentage of mobile nodes, the simulation area limits, the minimum and maximum speeds, hello interval, pause times and simulation duration. For the physical layer, a semi-deterministic channel model that takes into account the path-loss calculation with respect of 3D environment parameters is used.

The model also allows defining the maximum radio range as the radius of a mobile when operating at full transmission power and having an effective communication range. This range is approximately 300 meters which is a design parameter of some IEEE 802.11 products. All the nodes follow the 2D-RWM used in the scenario generator with speed ranging from 3 Km/h to 10 Km/h.

5.2 Simulation results and discussion

The network size is 500m x 500m. The number of nodes used in the whole network varies between 20 and 100. The simulations were run for 300 seconds. The cluster size was fixed at N=15. We depict some statistics on the formed clusters for different transmission ranges. In the first set of simulations, the scalability of the algorithm is measured in terms of nodes density and transmission range.

Figure 5 shows that for small ranges, most of nodes remain out of each other's transmission range, thus the number of clusters is relatively high and the network may become disconnected because there are no other choices. When transmission range increases, more nodes can hear each other. The average number of clusters formed decreases and the clusters become larger in size. When the transmission range is very small, most of nodes form one node cluster which only consists of itself.

Due to our algorithm design, which requires one-node clusters to attempt to merge with neighboring clusters with less number of nodes whenever possible, clusterheads will switch their status to non-clustered state in order to merge with their neighbors (if any). This causes the high rate of transitions in disconnected networks. However, we argue that this will not affect network performance as this will only occur when the network is disconnected (A disconnected network is unable to function too).

On the other hand, when the transmission range begins to be larger, mobile nodes tend to remain in the range of their neighbors and the number of transitions decreases. Therefore, clusters are less dynamic and the number of CH changes (number of changes occurred on the CHs during the entire simulation) also decreases as depicted in figure 6. We also compare the performance of our approach with the corresponding performance of the WCA algorithm while the nodes are moving under the same conditions.

In figure 7, we note that the performance difference is small between WCA and ECA with respect to the average number of clusters. This is because both algorithms are variations of a local weight based clustering technique that forms one-hop clusters. For high transmission range (more than 250 m), WCA generates less CH than ECA but to the detriment of a large number of transition on each CH (the number of times an elected CH changes its state from CH to a node member) (figure 8), where the stability is one of the important criteria in clustering because the frequent changes of CH adversely affect the performance of the clustering algorithm.

As shown in figure 8, with 100 nodes in the ad hoc network and for a transmission range equal to 180m, the proposed algorithm produced about 50.0% to 83.3% less transitions on each CH than WCA. In the WCA algorithm, WCA will keep changing with changes in topology. The CH of WCA algorithm relinquishes its position when another node having lower weight joins the cluster. In our algorithm, the CH has to verify the suitability of a new election even if a new node having lower weight has joined the cluster. As a result, our algorithm gives better performance in terms of stability when the node density in the network is high.

The result of the average number of re-affiliations (the number of different clusters a node joins during the time simulation) due to increasing node density is depicted in figure 9. For a transmission range of 120 meters, the number of re-affiliations increased when varying the number of nodes in the network for both our algorithm and WCA. As the number of nodes increased, the increasing rate of re-affiliations slowed down in ECA, which was not the case in WCA. For a node speed varying between 3 and 10 km/h, when there were 20 nodes in the network and for the same transmission range (120 m), the proposed algorithm produced 61.5% less re-affiliations than WCA. When the number of nodes was increased to 100, our algorithm gave 66.5% less re-affiliations than WCA for the same node speed.

6 Conclusion and Future Work

This paper has presented an Efficient Clustering Algorithm in Mobile Ad hoc Networks based on the quality of service's parameters (cluster throughput/delay, packet loss rate). It has the flexibility of assigning different weights and takes into account a combined metrics to form clusters automatically.

The effect of the number of competing cluster members, the contention window, the backoff stage and the length of transmitted packets are examined in order to estimate the achievable throughput and the cluster delay under traffic conditions that correspond to the general load that the network can support in stable conditions. We have concluded that the cluster performance strongly depends on the number of members.

Limiting the number of nodes inside a cluster allows restricting the number of nodes catered by a clusterhead so that it does not degrade the MAC functioning. A clusterhead with constrained energy may drain its battery quickly due to heavy utilization; in order to spread the energy usage over the network and achieve a better load balancing among clusterheads, re-election of the clusterheads may be a useful strategy; the algorithm is executed only when there is a demand. Also, if a node is moving away from the clusterhead, then the algorithm is flexible and cheap enough to be applied iteratively as the network configuration changes. Therefore, such approach provides a reliable method of cluster organization for wireless ad hoc networks. Simulation results indicated that the model agrees well with the behavior of the algorithm.

Eventually, the route between two nodes changes constantly as the clusterhead set changes. We are planning to study the overhead generated by ECA in order to evaluate its impact on the network, to allow a load balancing between the clusters. This might effectively drive congestion avoidance on the clusterheads and inter-clusters load-balancing to achieve better network resource utilization.

References:

[1] Das B., Bharghavan V., "Routing in ad-hoc networks using minimum connected dominating sets," IEEE International Conference on Communications (ICC Montreal 97), vol. 1, Jun. 1997, pp. 376-380.

[2] Chatterjee M., Das S.K., Turgut D., "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks," Cluster Computing Journal, vol. 5, no. 2, Apr. 2002, pp. 193-204.

[3] Gerla M., Tsai J. T. C., "Multicluster, Mobile, Multimedia Radio Network," ACM/Baltzer Wireless Networks Journal 95, vol. 1, Oct. 1995, pp. 255-265.

[4] Baker D.J., Ephremides A., "A distributed algorithm for organizing mobile radio telecommunication networks," Proceedings of the 2nd International Conference on Distributed Computer Systems, Apr. 1981, pp. 476-483.

[5] Baker D.J., Ephremides A., "The architectural organization of a mobile radio network via a distributed algorithm," IEEE Transactions on Communications, Nov. 1981, pp. 1694-1701.

[6] Ephremides A., Wieselthier J. E., Baker D. J., "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," Proceedings of the IEEE, vol. 75, no. 1, Jan. 1987, pp. 56-73.

[7] Basagni S., "Distributed clustering for ad hoc networks," Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks, Jun. 1999, pp. 310-315.

[8] Basagni S., "Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks," Proceedings of Vehicular Technology Conference, VTC, vol. 2, fall 1999, pp. 889-893.

[9] Basu P., Kham N., Little T. D. C., "A mobility based metric for clustering in mobile ad hoc networks," International Conference on Distributed Computing Systems Workshop, Apr. 2001.

[10] Chiang C. C., Wu H. K., Liu W., Gerla M., "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," IEEE SICON, Singapore, Apr. 1997.

[11] Parekh A.K., "Selecting routers in ad-hoc wireless networks," Proceedings of the SBT/IEEE International Telecommunications Symposium, Aug. 1994.

[12] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function", IEEE Journal on Selected Areas in Communications, Vol. 18, No. 3, 2000, pp. 535-547.

[13] G. Bianchi, L. Fratta, and M. Oliveri, "Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs", PIMRC'96., Vol. 2, 1996, pp. 392 -396.

[14] G. Bianchi, "IEEE 802.11-saturation throughput analysis", IEEE Communications Letters, Vol. 2, No. 12, 1998.

[15] Y. Tay and K. Chua, "A capacity analysis for the IEEE 802.11 MAC protocol", Wireless Networks, Vol. 7, No. 2, 2001, pp. 159-171.

[16] IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[17] Agba L., Gagnon F., Kouki A., "Scenarios generator for ad hoc networks," International Symposium on Industrial Electronics, Montreal, Canada, Jul. 2006.

FIGURES AND TABLES

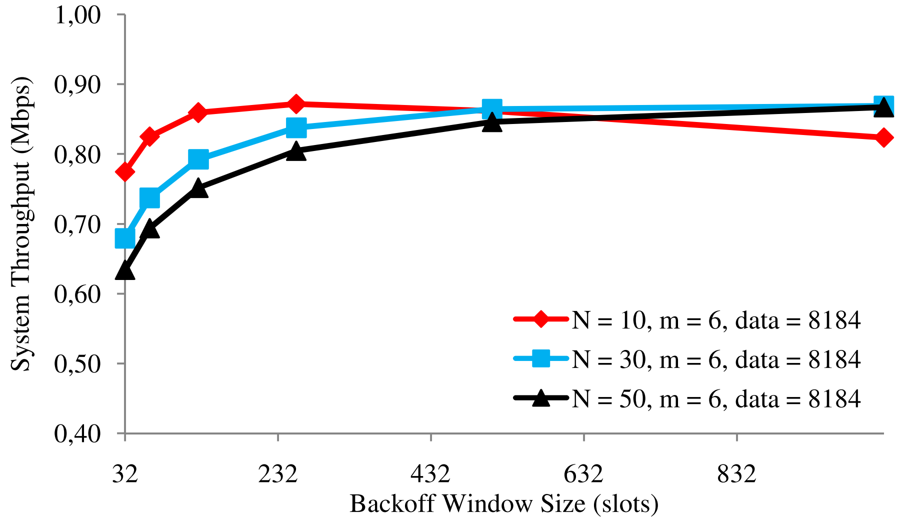

Fig. 1. Cluster throughput versus backoff window size

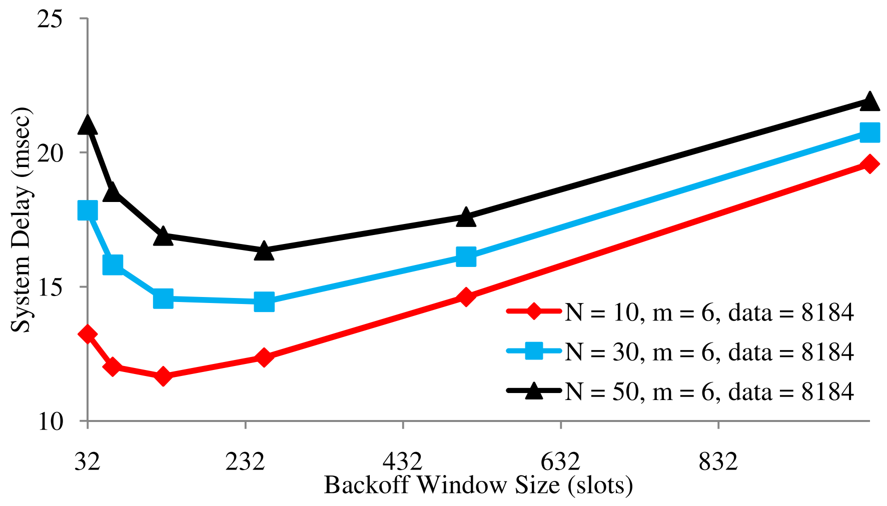

Fig. 2. Cluster delay versus backoff window size

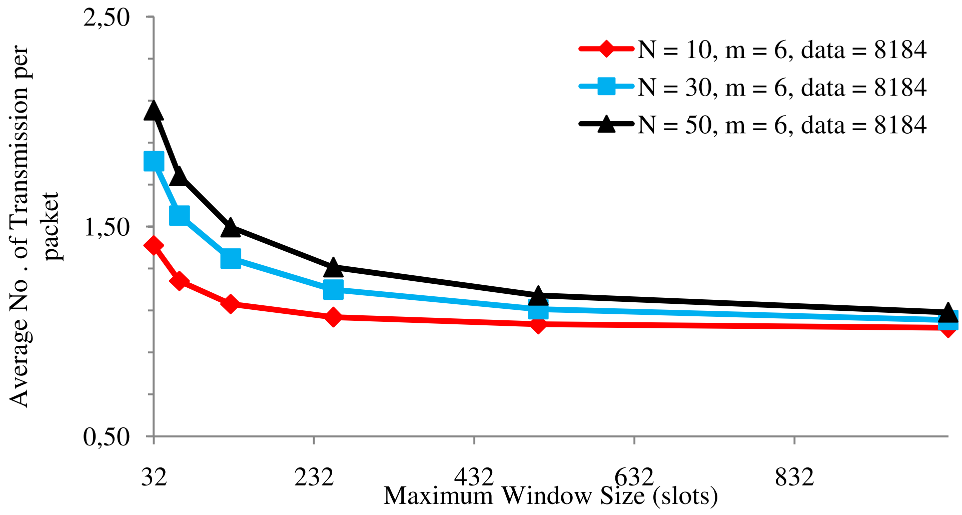

Fig. 3. Average no. of transmission per packet versus window size

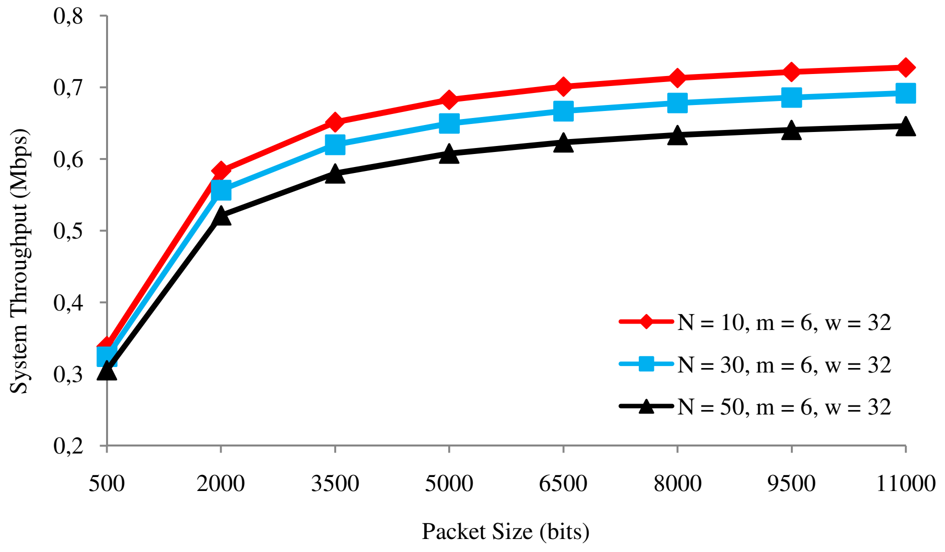

Fig. 4. Cluster throughput versus packet size

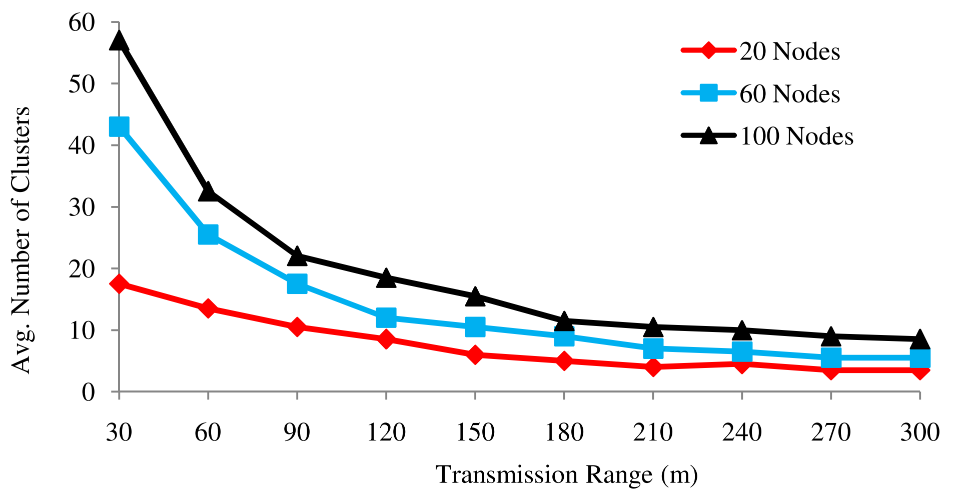

Fig. 5. Transmission range vs. Avg. Number of Clusters

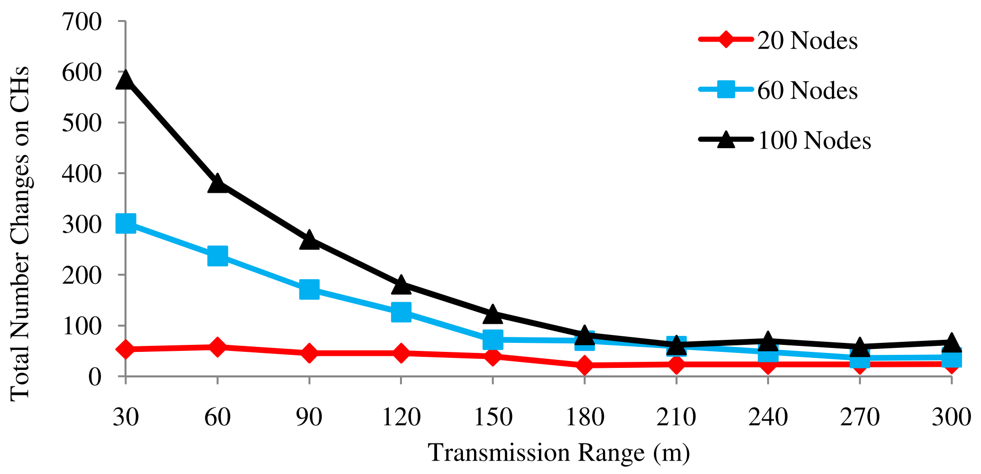

Fig. 6. Transmission range vs. Avg. Number of CH Changes

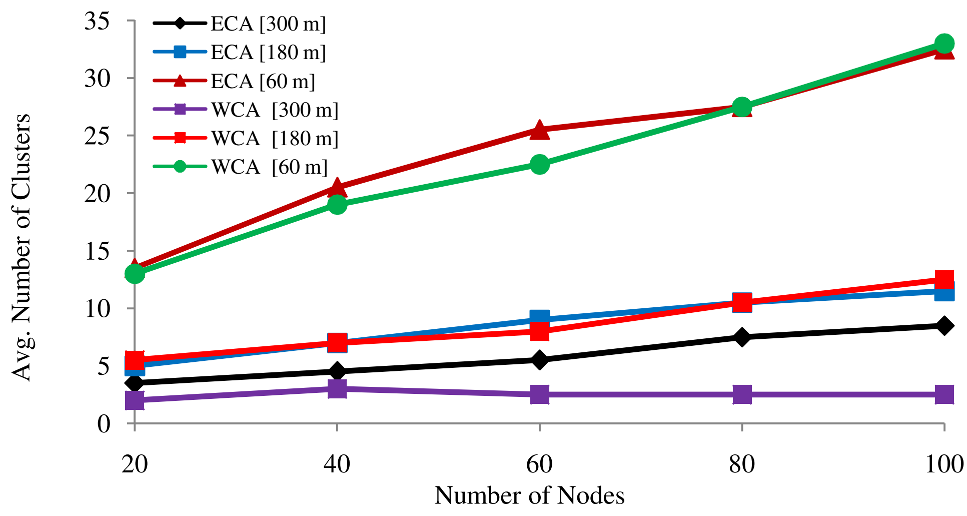

Fig. 7. Number of Nodes vs. Avg. Number of Clusters

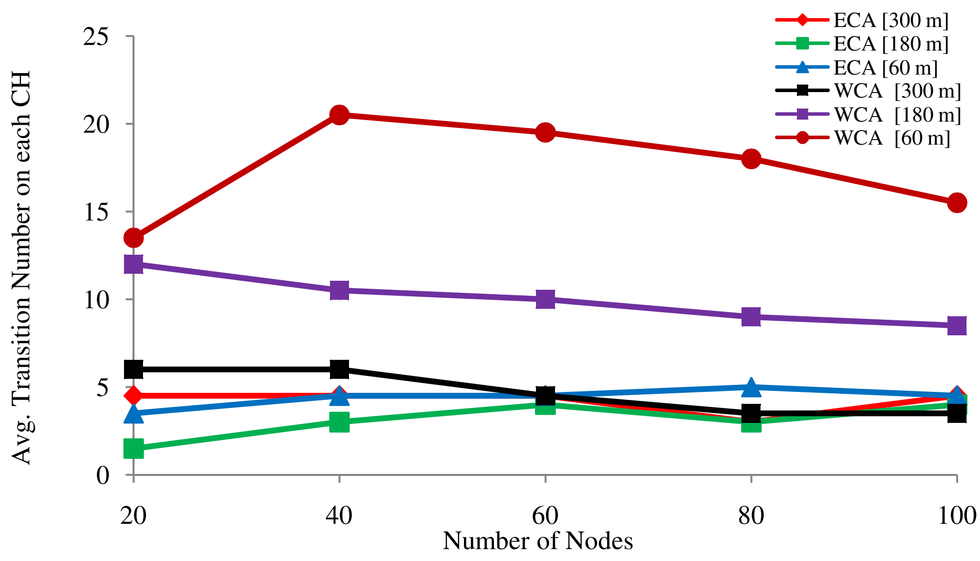

Fig. 8. Number of Nodes vs. Avg. Transition Number on each CH

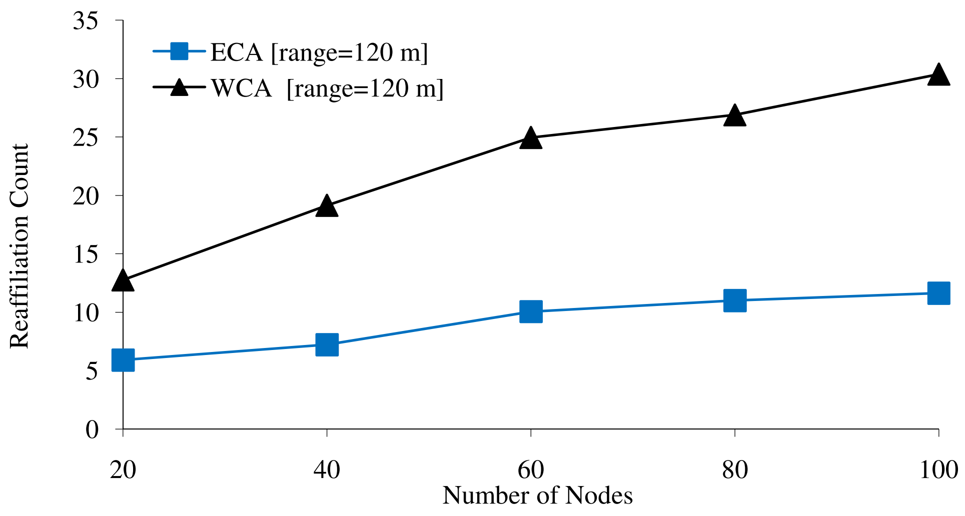

Fig. 9. Number of Nodes vs. Re-affiliation count

Table 1. Messages used for the clustering algorithm ECA

Message	**Description**
Hello(id_{node}, id_{CH}, Weight, Counter, N)	To update the tables of the nodes
Join_Request(id_{node}, id_{CH})	To affiliate a cluster
welcome_ACK(id_{node}, id_{CH}, Weight)	The CH accepts a Join_Request
welcome_NACK(id_{node}, id_{CH}, Weight)	The CH rejects a Join_Request
CH_Request(id_{node})	The node declares itself as CH
CH_Response(id_{node})	The CH accepts a CH_Request
Join_Accept(id_{node}, id_{CH}, Weight, Counter, N)	The node accepts the welcome_ACK
CH_ACK(id_{node}, id_{CH}, Weight, intracluster_code, Counter, N)	The CH adds the node as a member
Database_info(id_{node}, id_{CH}, Weight, Counter, N)	The current CH sends the database to a new elected CH
Database_ACK(id_{node} , id_{CH}, Weight, Counter, N)	The new elected CH accepts the received database
CH_change(id_{CH})	The CH notifies a CH change
CH_info(id_{node}, id_{CH}, Weight, Counter, N)	The CH accepts the presence of a new CH in the network
Leave_Request(id_{node} , id_{CH}, Weight, Counter, N)	The node leaves the cluster