Simulation Modelling Practice and Theory 48 (2014) 1–23

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A simulation-based performance evaluation of a randomized MIS-based clustering algorithm for ad hoc networks

Dimitrios Papakostas, Dimitrios Katsaros *

Electrical & Computer Engineering Dept., University of Thessaly, Volos, Greece

ARTICLE INFO

Article history:
Received 17 April 2014
Received in revised form 23 June 2014
Accepted 24 June 2014

Keywords:
Dominating sets
Maximum independent sets
Backbone formation
Clustering
Ad hoc networks

ABSTRACT

Ad-hoc networks represent distributed systems that comprise wireless nodes which can dynamically self-organize into arbitrary and temporary network topologies, without relying on pre-existing infrastructure, and thus network hierarchy formation via clustering is vital for them. The present article conducts a comprehensive simulation-based evaluation of the performance achieved by a recently proposed, biology-inspired, clustering algorithm used in wireless ad hoc networks, namely the Randomized Beep Based Maximum Independent Set (*RanMIS*) (Afek et al., 2011). This is the first evaluation done for this high-performance algorithm. The evaluation is done for a set of metrics (measures for protocol cost, backbone description and robustness) some of which has not been used in earlier simulation studies and are developed here. Our study confirms the virtues (message complexity) and reveals the shortcomings of *RanMIS* (latency issues), and quantifies the impact of some of its administratively-tuned parameters. *RanMIS* is compared with two representative graph-theoretic node clustering methods and a new one developed here; the results confirm the message optimality of *RanMIS*, but reveal some shortcomings of it, basically related to the excessive number of rounds that needs to run in order to complete the network clustering.

1. Introduction

A wireless ad hoc network is a type of wireless network which eliminates the complexities of infrastructure setup and administration, by enabling existing nodes to create and join networks *"on the fly"*, anywhere, anytime, for virtually any application. Each node participates in routing by forwarding data for other nodes, but the determination of which nodes will finally forward data is made dynamically.

The decentralized nature of wireless ad hoc networks makes them suitable for a variety of applications where central nodes cannot be relied on, and may improve the scalability of wireless ad hoc networks, compared to wireless managed networks. The minimal required configuration, the quick deployment, and the presence of dynamic and adaptive routing protocols, which allows them to be formed quickly, make ad hoc networks suitable for situations like habitat monitoring, disaster relief, law enforcement operations, battle field communications, target tracking and so on.

The dynamic nature of wireless ad hoc networks though, requires that solutions for multi hop network protocols at all levels must be distributed. Of the solutions proposed for scaling down networks with large numbers of nodes, *network*

* Corresponding author.
E-mail addresses: jim.papakostas@gmail.com (D. Papakostas), dkatsar@inf.uth.gr (D. Katsaros).

http://dx.doi.org/10.1016/j.simpat.2014.06.012
1569-190X/

clustering is among the most investigated for mobile ad hoc networks [2–12], for sensor ad hoc networks [13,14], for wireless mesh networks [15] and for vehicular ad hoc networks [16–18]. Other solutions of network management such as topology control [19] are remotely related to this work.

The basic idea in clustering is that of grouping network nodes that are in physical proximity, thereby providing the *flat* network a *hierarchical* organization, which is smaller in size, and simpler to manage.

The subsequent *backbone* construction uses the induced hierarchy to form a communication infrastructure that is functional in providing desirable properties such as minimizing communication overhead, choosing data aggregation points, increasing the probability of aggregating redundant data, and consequently minimizing the overall power consumption.

In a clumpy approach, clustering algorithms can be divided in two major families. The first with its roots in *graph theory*, exploits the localized network structure for estimating dynamically the clusterheads (CHs), while the second provides mechanisms to ameliorate the fact that nodes belonging to the backbone are solely responsible for carrying out all communication, thus running out of energy very soon. Namely, the latter family addresses the energy consumption problem, that is essentially proposes ways to rotate the role of CH among nodes of clusters e.g., the SPAN [20], the LEACH [21] and the HEED [22] protocols. The proposed methods use the residual energy of each node in order to direct its decision about whether it will elect itself as a CH or not. However, this family's methods ignore topological features of the nodes.

The former family of protocols encompasses the most representative and successful solutions in the area of network management. This is because while it exploits the wealth of information extracted from the network topology particularities, it can also take into account application specific requirements, such as QoS. Moreover, algorithms of this family can easily be combined with a round robin CH rotation method as was described in [13], and thus exploit all the advantages the energy-efficient algorithms provide.

Recently, *RanMIS* [1,23] – which belongs to the first family – was proposed as a message-optimal algorithm; it employees a probabilistic technique for node clustering and enables every node to independently decide on its role in the clustered network. It ensures rapid convergence, while keeping the message overhead low. Its message complexity on CH selection was studied analytically in [23], but its performance is still in question with respect to the delay it incurs for the clustering protocol termination, for the 'stability' of the created clusters in case of node failures, and so on. The focus of this paper is exactly to fill this gap by conducting a detailed experimental evaluation of *RanMIS* by simulation.

RanMIS competitors were selected from two distinct protocol categories.[1] The first category is oriented on providing the network with a two-layer hierarchical organization comprised by groups of nodes, i.e., clusters. One 'special' node of each cluster (the CH) participates in the so-called backbone; the backbone nodes form a *dominating set (DS)* over the flat network topology. This means that, each node that is not in the backbone, has at least one backbone node as its neighbor. Backbone nodes are joined via *gateway* nodes. Apparently, if the flat network is connected, it is deterministically guaranteed, that the backbone is connected as well. Distributed Clustering Algorithm (*DCA*) is a high-performance representative of this category.

The second category retains the layered structure of the first category, and it is oriented on facilitating routing without the need of gateway nodes. The concept behind this category is that of building a *Connected Dominating Set (CDS)* directly, without firstly selecting CHs and then joining them. It was initially introduced by Das et al. in [24] when they proposed the concept of creating a *communication spine* inside a flat network topology in order to support unicast, multicast, and fault-tolerant routing within an ad hoc network. *WuLi* is a practical and robust representative of this category.

Contrary to *RanMIS*'s randomized nature, its competitors use iterative clustering techniques, meaning that a node waits for a specific event to occur or certain nodes to decide about their role before making a decision themselves.

1.1. Motivation and contributions

The problem of providing a hierarchical organization for ad hoc networks is of crucial importance due to the widespread deployment of such networks (e.g., mobile ad hoc networks, sensor networks, vehicular ad hoc networks) and their scalability problems. Despite the fact that a really large number of algorithms have been proposed during the last decade for performing clustering, the proposal and study of protocols which exhibit low message complexity, fast convergence, incremental backbone maintenance, resilience to hub node failures, connectivity preservation, and backbone stability is still in quest. The considered clustering algorithms seem to encompass all the aforementioned features and therefore their detailed, joint investigation is a significant task for the area of ad hoc network management.

In this context, the present article makes the following contributions:

- We investigate for the first time by detailed simulation the performance of *RanMIS*. In particular:

  - We confirm its message optimality for backbone construction across all examined network topologies.
  - We reveal the impact of parameter D on the number of rounds before *RanMIS* terminates; moreover, we prove its independence on each node's neighborhood size, contrary to *RanMIS*' original article suggestion.
  - We question the value of parameter M as suggested by the authors of *RanMIS*.

[1] More detailed arguments about the selection of competitors can be found at subSection 3.4.

  - We illustrate *RanMIS*'s performance lagging with respect to the number of rounds before termination, which means that it will incur significant delay for clustering.
  - We provide a characterization for the distribution of nodes in clusters for the competing protocols, and show *RanMIS*'s drawback in that it produces 'long' backbones, which implies increased delays for message routing.

- We propose an improvement to *WuLi*, namely *WuLi** appropriate for cluster rebuilding processes in case of node failure.
- We develop a rich comparison framework for the ad hoc network clustering protocols, employing three families of performance measures, namely for protocol cost, for backbone description and for robustness.
- We provide a detailed comparison of the algorithms for these families of measures for many network topologies (with different size and density).

The rest of this article is organized as follows: in Section 2, we briefly describe the operation of *RanMIS*; in Section 3 we firstly describe the operation of *RanMIS*'s competitors and then provide the detailed performance evaluation of *RanMIS* against them; finally, in Section 4 we conclude the article.

2. The Beep-based randomized MIS algorithm

RanMIS [23] considers the problem of computing a *maximal independent set (MIS)*, a fundamental distributed computing procedure, that seeks to elect a set of local leaders in a network.

It generates the maximal set of nodes in such a manner that, no two of them are neighbors, by using an extremely harsh broadcast model that relies only on carrier sensing. Since the set is maximal, every node in the network is either in the *MIS* or a neighbor of a node in the *MIS*.

The model consists of an anonymous broadcast network in which nodes have no knowledge about the topology of the network. It has its roots to the solution of a similar problem that evolves during the development of the fly's nervous system, when sensory organ precursor (SOP) cells are chosen [23].

According to the specified assumptions, nodes receive as input, an upper bound on the number of nodes in the network n, and an upper bound D, on the number of neighbors any node can have (if no such bound is known, then D is set to n). Furthermore, it is assumed that they all wake up together at the same synchronous round (and start executing the algorithm), also that they can detect collisions, and finally that no failures occur.

The algorithm proceeds in $log\ D$ phases, each consisting of $M\ log\ n$ steps, where M is a constant. Initially, all nodes are active. Each step in each phase i consists of two exchanges.

In the first exchange, each active node broadcasts a message to its neighbors with probability p_i. Such as in the biological model, the probability p_i increases with i. In the second exchange, a node that has broadcasted a message in the first exchange joins the *MIS* if none of its neighbors had broadcasted at the first exchange. Such node broadcasts again a message to its neighbors, telling them to become inactive, and exits the algorithm.

For the sake of completeness we give in Fig. 1 the pseudocode of the *RanMIS* algorithm [23] which is synchronously executed by all nodes.

3. Performance evaluation

The exhaustively investigated in the past competitor algorithms involved in our analysis, apart from the fact that they belong to different graph-theoretic classes, were chosen basically because of their distinct diversification in the degree of localization compared to *RanMIS*.

The first class, which enables *WuLi* [10], exhibit a high degree of localization. In such a class as soon as a node has collected information about its surrounding topology, it is able to decide whether it will be part of the backbone or not. The only information it needs to wait for, is the identity of the node in its (h hop) neighborhood. In the second class, the degree of localization shown by the algorithms is limited with respect to that of the first. Such protocols implement a distributed version of the heuristics for finding an *independent set* of nodes which is maximal, and a *dominating set* which is minimal. As a representative algorithm of this class we consider the *DCA* [4]. Being maximum, the *independent set* produced by *DCA* is also a minimal dominating set.

WuLi is a very simple distributed procedure consisting of a few local rules, the execution of which creates the desired *CDS*. Every node v exchanges its neighbor list with all its neighbors. A node set itself has a dominating node if it has at least two unconnected neighbors. In order to reduce the size of a *CDS*, the original protocol presents two pruning rules. According to the first rule, a node deletes itself from the *CDS* when its close neighbor set, which includes all of its direct neighbors as well as itself, is completely included in the neighbor set of a neighboring dominating node and it has smaller ID than the neighboring dominating node. According to the second rule, a node deletes itself from the *CDS* when its open neighbor set, which includes all of its direct neighbors, is completely included in the neighbor sets of two connected neighboring dominating nodes and has the smallest ID.

```
Algorithm: RanMIS (n, D) at node u
// n: number of participating nodes in network topology
// u: node under consideration
// D: upper bound on the number of neighbors
// B: 1 bit message
// M: constant

1 . For(i = 0 : log D)
2 .   For (j = 0 : M log n)
3 .       * exchange 1 *
4 .       v = 0;
5 .       With probability 1/2^(log D - i),
              broadcast B to neighbors;
              set v = 1;
6 .       If received message from neighbor, then
              v = 0;
7 .       * exchange 2 *
8 .       If v = 1 then
9.            Broadcast B;
              join MIS;
              exit the algorithm.
10.       Else
11.           If received message B in this exchange, then
                  mark node u inactive;
                  exit the algorithm.
12.       End.
13.   End.
14. End.
```

Fig. 1. The randomized MIS clustering algorithm.

The *DCA* protocol assumes quasi-stationary nodes with real-valued weights, which are initially *UNDECIDED*. Node weights induce a total ordering of the nodes without any ties because node weights reflect real numbers. During the execution of the algorithm each node will decide to be *IN* or to be *OUT*, of the *independent set*. A node decides when all its neighbors with larger weight have decided. When the time comes, a node decides to be *OUT* if one of its neighbors is *IN*. Otherwise it decides to be *IN*. The *IN* nodes form the minimal dominating set required for clustering. The execution time depends on possible chains of dependency between the nodes, a negative consequence of the reduced localization *DCA* presents. The node with the smallest weight has to wait for all other nodes in the chain.

In Section 3.1 we briefly introduce the graph model used in our simulation. Then in Section 3.2 we present the simulation platform that produced all the network topologies used in the performance analysis. In Section 3.3 we give the performance metrics of the protocols comparison and finally in Section 3.4, we display the simulation results.

3.1. Simulation model

As in most studies in multihop wireless networks, we used unit disk random graphs to represent the network topologies used in the performance analysis. A unit disk graph is determined by node positions and a fixed *communication range R*, for all nodes. The produced topologies which more precisely are described as *Constrained – Connected Random Unit Graphs (C-CRUG)*, because of the constrains the participating nodes are obliged to adhere, were constructed in MATLAB by taking in to consideration two crucial independent variables, concerning the final network connectivity, that is the network density, which is perfectly expressed by the degree d of each node, and the node population n.

3.2. Simulation platform

The graphs were normally generated by placing nodes randomly and independently from each other with the help of a modified version of *Minimum degree proximity algorithm (MIN-DPA)* [25], a *C-CRUG* generator, which aims to distribute node degrees more uniformly while maintaining connectivity.

The idea is to place each new node in the vicinity of the node that has the smallest number of neighbors, while preserving a minimum distance to increase the probability of achieving a connected graph. The modified version of it though, involves the notion that the average degree of the topology and the transmission range of a node, were predefined.

Table 1
L parameter values.

Nodes	Degree#4	Degree#7	Degree#10	Degree#15
100	500	400	250	200
500	2000	1500	750	500
1000	5000	4000	1000	700
5000	7000	5000	2000	1500

The simulations refer to scenarios in which n static wireless nodes with maximum transmission range R are randomly and uniformly scattered in a geographic area of side L. We make the assumption that two nodes are neighbors if and only if their Euclidean distance is less than R.

We emphasize that nodes start the protocol execution at the same time and run the clustering and backbone formation algorithms to form a hierarchical multi-hop ad hoc network.

In our simulations R has been set to 50 m, the number of nodes n has been assigned the values $100, 500, 1000$ and 5000 while L has been set accordingly (see Table 1) to influence an average node degree of $4, 7, 10$ and 15. This allowed us to test the protocols on increasingly dense networks, from (moderately) sparse to dense networks.

In order to diminish any statistical errors all tests were repeated for 1000 times and the data used in our graphs are all average values of the results. A short description of the simulation parameters is given in Table 2.

3.3. Performance metrics

The assessment of the competing protocols, which was also implemented in MATLAB, were done along three dimensions: the first dimension portrays the cost (in delay and energy consumption) incurred by the protocols, the second concerns the characteristics of the resulting backbone, and the third one deals with the resilience of the resulting spanner to node failures.

Metrics for protocol cost. This family of measures, includes the *protocol duration* which is a direct measure of the delay incurred until the network spanner is established, and the *total number of messages exchanged* among nodes which is a measure of the energy consumed by the network, especially crucial for assessing the usefulness of clustering to networks with energy constrained nodes.

Protocol duration. It is the number of rounds required by the protocol to complete the procedure of backbone formation.

Total number of messages transmitted. We deal with two different situations (cases) where message exchange takes place among network nodes. The first situation concerns the backbone construction operation, i.e., the selection of network nodes which will take on the role of a CH, and the declaration of 'attachment' of non-CH nodes to a CH. The second situation occurs when a backbone node fails. We do not examine failures of non-CH nodes[2] because message exchange in such scenarios is unrelated to the clustering protocol running. It is obvious that, if the number of messages exchanged for backbone maintenance is high, then local reclustering may impose a non-negligible burden to the network, and a backbone recalculation should take place. To uncouple our measurements from MAC particularities and focus only to the pure clustering protocol performance, we assume an ideal MAC layer with no provision for retransmissions in case of collisions exist.

Metrics for backbone description. This family of measures assess the backbone's 'layout'.

Backbone size. It is the number of the network nodes that comprise the backbone. A smaller backbone is desirable for minimizing the routing overhead. For instance, in sensor networks with nodes that can turn off their radio interface (energy saving sleep mode), the backbone size is a measure of how many nodes need to stay awake for data transport. Usually, the nodes in the backbone stay up or have a higher duty cycle for guaranteeing routes to the sink, and hence the smaller the backbone the more nodes can be in energy conserving mode. On the other hand, a very small backbone could result in situations where CH-to-member communication would require either excessive amount of energy to reach the node and vice versa, or multi-hop communication within the cluster.

Cluster cardinality distribution. It is the distribution of the number of members for each generated cluster.

Route length. If we consider the subgraph G_b of the network induced by the backbone construction (G_b is the graph where there are no links between ordinary nodes), then this metric gives a measure of how well a routing protocol can perform over the backbone.

Inter-clusterhead distance distribution. This distribution records the distances among neighboring CHs. The larger this distance is the better for the performance of the clustering protocol, since, for instance, a three-hop distance among clusterheads would avoid hidden terminal problems and guarantees the 'independence' of clusterhead transmission.

Metrics for robustness. This evaluation is useful in cases where node failures are anticipated. There are various definitions for evaluating the robustness of a network [26,27]. In general, the "robustness" of a network measures its resilience (in terms of connectivity) to the removal of edges or vertices. Robustness can be formalized in a variety of ways, depending on

[2] Either gateway nodes which are used by *RanMIS* and *DCA* in order to achieve backbone connectivity or plain nodes.

Table 2
Interpretation of used symbols.

Symbol	Usage	Default value
n	Number of nodes in network	
D	Intended number of neighbors	24
d	Average node degree	
M	Constant	32
R	Node transmission range	
L	Axis length (in meters) of *AOR* (Area Of Responsibility)	

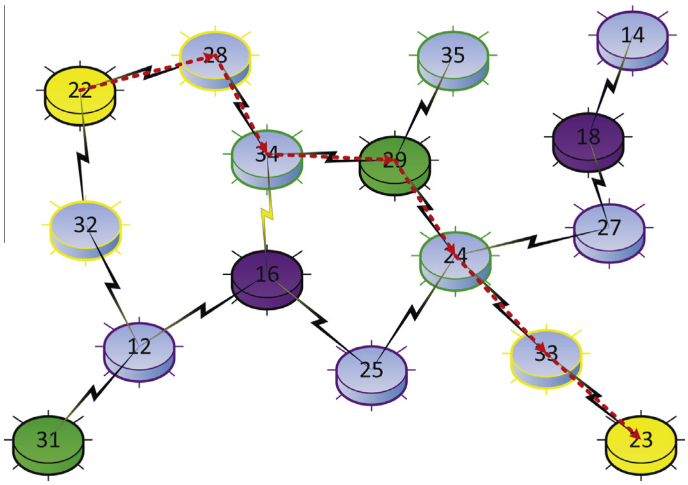

Fig. 2. Shortest path transmission when all nodes are in working condition.

how connectivity is measured, whether edges or vertices are removed, and how the edges or vertices to be removed are chosen.

In our experimentations, we consider a wireless ad hoc network where nodes are removed randomly until backbone connectivity is lost. Fig. 2, depicts a section of a larger wireless network where a message is transmitted from $node_{22}$ to $node_{23}$. It is obvious that the scenario under consideration deals with an MIS network topology where gateway nodes are used to forward any transmitted messages. Adjacent CHs differ in their fill in colors and any clustermembers have a border line color which is in consistency with their CH fill in color. A red dotted line is used to represent the shortest path for the current transmission. We emphasize that in our tests whenever a clustermember faces the dilemma of choosing between more than one CH candidates then by default it selects the one with the smaller ID.

We assume that nodes have an inherent capability to identify in their neighborhood either a failed CH as clustermembers or a failed gateway node as CHs. Taken this assumption in to consideration, as soon as a node failure is perceived, each competitor protocol strives to solve locally any network inefficiencies so that the connectivity is maintained and no uncovered nodes exist. The metric considers an upper bound of 3-hop message retransmission in order to establish connection between CHs. We examined two different scenarios for robustness depending on the role of the failed node in the network.

Network substratum Robustness is defined as the number of non-CH nodes whose removal (because of failure or energy depletion) causes backbone disconnection. This metric is directly related to *RanMIS* and *DCA* protocols who build an MIS network topology and use as *gateways* non-CH nodes to maintain backbone connectivity. *WuLi* who builds CDS and is presented also in the diagram illustrates the total number of non-CH nodes in the network because failure of non-CH nodes has negligible effects to backbone connectivity. Intuitively network performance is connected to the cardinality of the clusters and it is straightforward that a populated cluster is more desirable than a sparse occupied.

In Fig. 3, we see the impact of a gateway failure to the shortest path construction. If the failure concerns a non-gateway node then network connectivity will not be affected at all.

We addressed network substratum failures as follows:

- Initially, we run Dijkstra's routing algorithm on the whole network topology to identify if backbone connectivity is maintained after node failure.

- If so, which means that the gateway role on the cluster who suffered the failure was assigned to another node without any further consequences to the backbone connectivity, we repeat the scenario with another node until backbone connectivity is lost.
- If not, which means that node failure had catastrophic consequences to the backbone connectivity, we use the total number of the failed nodes that each protocol could bear prior to loss of backbone connectivity to describe the network resilience to failures of such kind w.r.t. network size and its density.

Backbone Robustness, is defined as the number of backbone nodes whose removal (because of failure or energy depletion) causes backbone disconnection or the uncovering of ordinary nodes. This metric provides an indirect measure on how long the network will be operational before requiring a backbone recomputation.

We use the network topologies presented in Figs. 2, 4, and 5 to describe the way we address CH node failures with *RanMIS*, *DCA*, *WuLi* protocols, respectively.

In Fig. 6, we see the impact of an *RanMIS* CH failure to the shortest path construction.

We addressed *RanMIS* CH failures as follows:

- We locate any resultant orphan clustermembers due to CH failure.
- If any, we attempt to associate them with a neighboring CH.
- Remaining orphan clustermembers become CH candidates, and rerun the protocol.

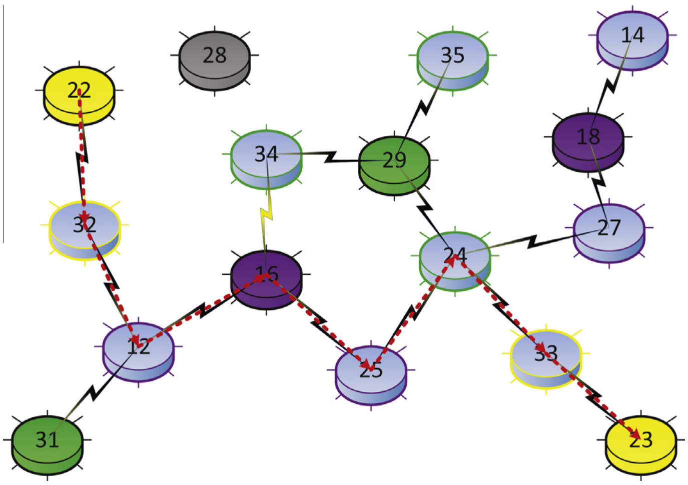

Fig. 3. Shortest path recalculation as a result to gateway node failure.

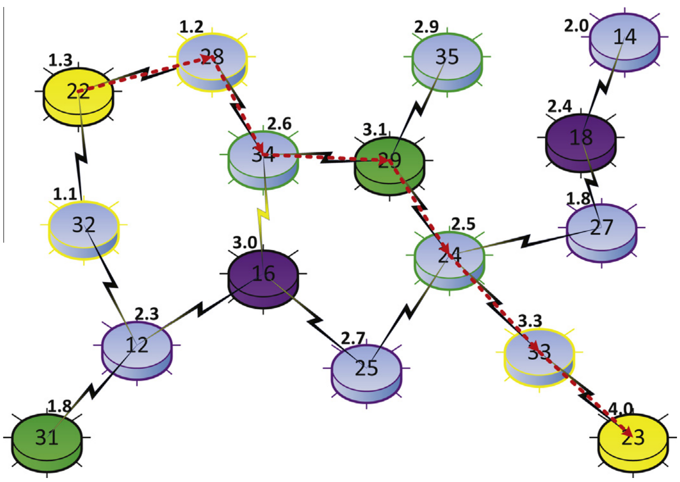

Fig. 4. Shortest path for a DCA network topology.

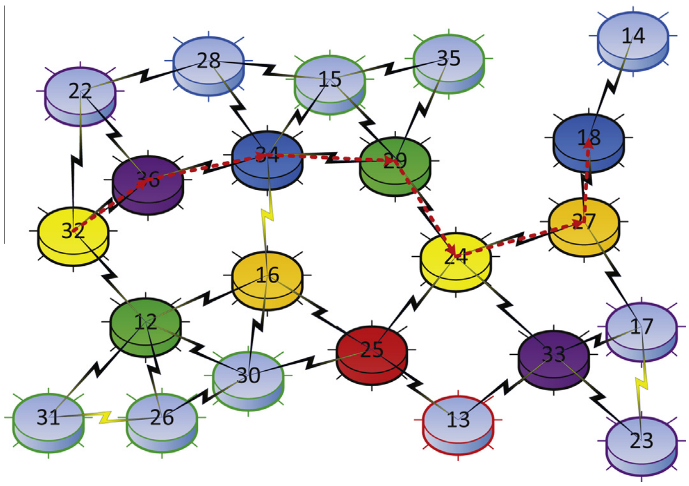

Fig. 5. Shortest path for a WuLi network topology.

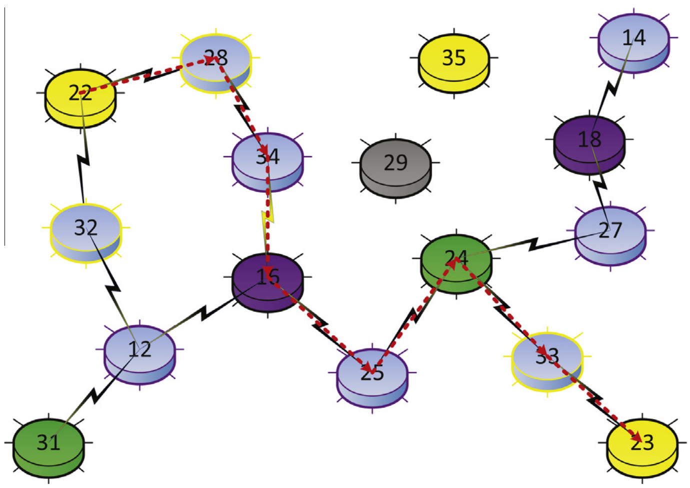

Fig. 6. Shortest path recalculation as a result to *RanMIS* CH node failure.

- Then we run Dijkstra's routing algorithm on the whole network topology to identify if backbone connectivity is maintained.
- If so, we repeat the scenario with another one CH until backbone connectivity is lost.
- If not, we use the total number of the removed nodes that each protocol could bear prior to loss of backbone connectivity to describe the network resilience to failures of such kind w.r.t. network size and its density.

In Fig. 7, we see the impact of a *DCA* CH failure to the shortest path construction.
We addressed *DCA* CH failures as follows:

- We locate any resultant orphan clustermembers due to CH failure.
- If any, their status will be determined in sequence behind any undecided neighboring node with larger weight.
- When appropriate we associate orphan clustermembers with a CH whose weight is the larger in the neighborhood.
- Remaining orphan clustermembers become CH candidates, rerun the protocol and finally are announced CHs.
- Then we run Dijkstra's routing algorithm on the whole network topology to identify if backbone connectivity is maintained.
- If so, we repeat the scenario with another one CH until backbone connectivity is lost.
- If not, we use the total number of the removed nodes that each protocol could bear prior to loss of backbone connectivity to describe the network resilience to failures of such kind w.r.t. network size and its density.

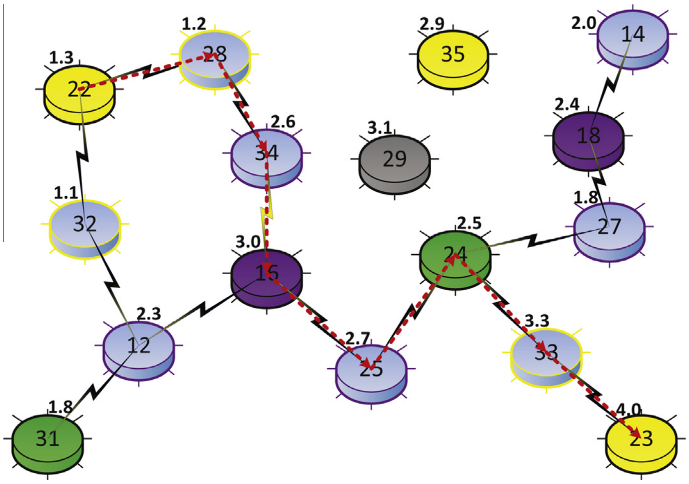

Fig. 7. Shortest path recalculation as a result to *DCA* CH node failure.

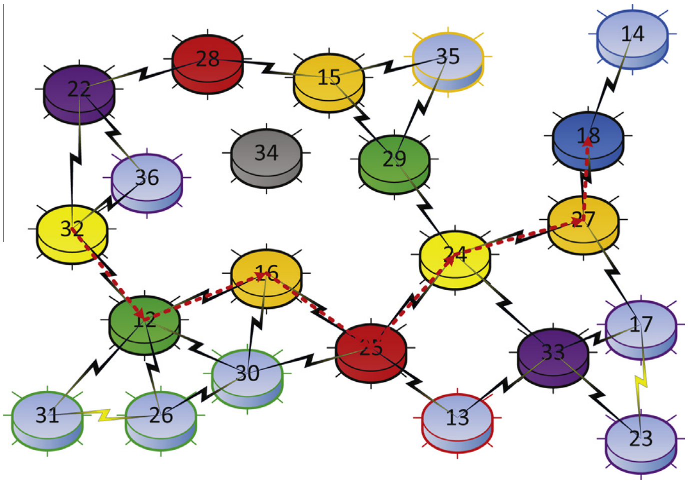

Fig. 8. Shortest path recalculation as a result to *WuLi* CH node failure.

In Fig. 8, we see the impact of a *WuLi* CH failure to the shortest path construction.
We addressed *WuLi* CH failures as follows:

- Initially, we locate any CH whose role was determined by the presence of the failed node, when it was in working condition, and we mark it as an orphan clustermember.
- Any clustermembers that the former CH had, are marked as orphan clustermembers also.
- Afterwards, we locate any resultant orphan clustermembers due to CH failure.
- Subsequently, we attempt to associate orphan clustermembers with a neighboring CH.
- Then we run Dijkstra's routing algorithm on the whole network topology to identify if backbone connectivity is maintained.
- If so, we repeat the scenario with another one CH until backbone connectivity is lost.
- If not, we use the total number of the removed nodes that each protocol could bear, prior to loss of backbone connectivity, to describe the network resilience to failures of such kind w.r.t. network size and its density.
- It is worth to mention that, uncovered clustermembers are taken into account and cause the termination of protocol execution when discovered.

The way *WuLi* addresses CH failures which was proposed by its authors in [10] has been proven inefficient concerning the overall energy consumption to backbone maintenance. It is our belief that once a network structure has been established the

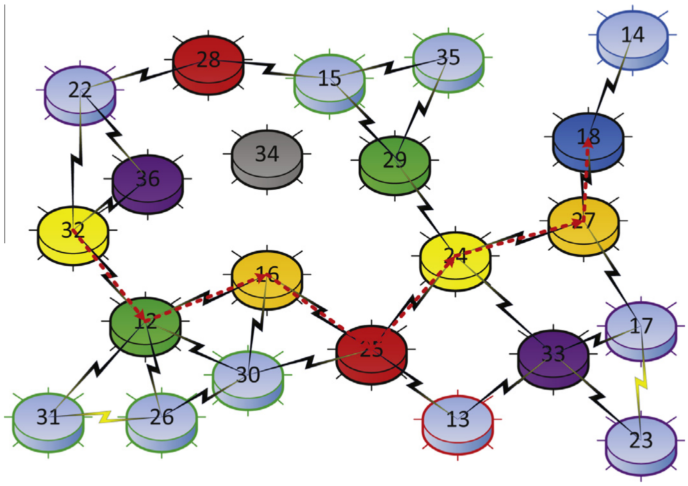

Fig. 9. Shortest path recalculation as a result to *WuLi*★ CH node failure.

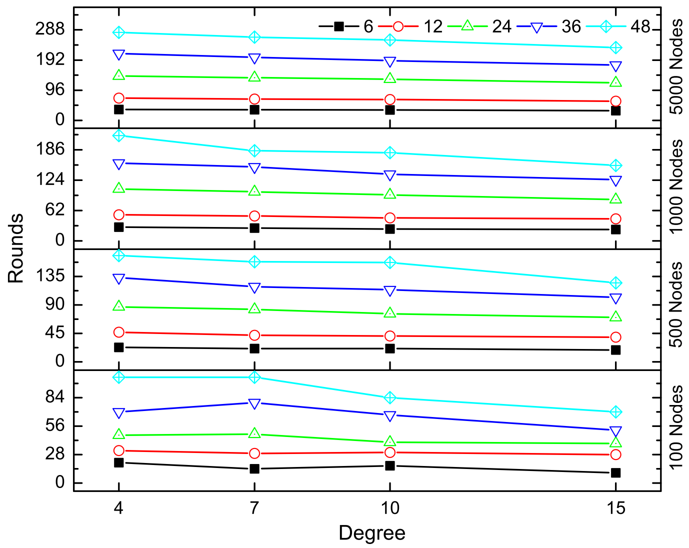

Fig. 10. Impact of *D* on the number of rounds required by *RanMIS* to complete backbone construction w.r.t. network size and its density.

effort should be to preserve it as long as the connectivity is maintained. We were motivated by this assumption to present a modified version of *WuLi* concerning the way it deals with CH node failures. In Fig. 9, we see the impact of a *WuLi* CH failure to the shortest path construction when using the Proposed *WuLi* protocol.

In our proposed version of *WuLi*★ we address CH failures as follows:

- Initially, we deal with substratum clustermembers to identify whether they can be assigned a new role due to CH failure or not.
- Subsequently, we attempt to associate orphan clustermembers with a neighboring CH.
- Then we run Dijkstra's routing algorithm on the whole network topology to identify if backbone connectivity is maintained.
- If so, we repeat the scenario with another one CH until backbone connectivity is lost.
- If not, we use the total number of the removed nodes that each protocol could bear, prior to loss of backbone connectivity, to describe the network resilience to failures of such kind w.r.t. network size and its density.
- Uncovered clustermembers are taken into account also and cause the termination of protocol execution when discovered.

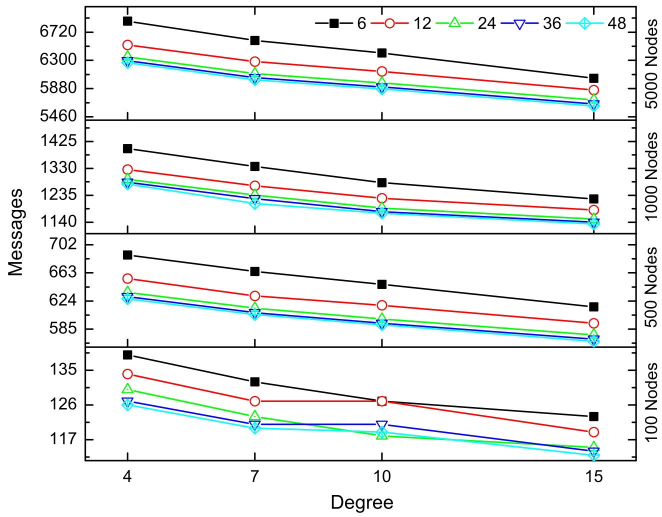

Fig. 11. Impact of *D* on the number of messages exchanged by *RanMIS* for backbone construction w.r.t. network size and its density.

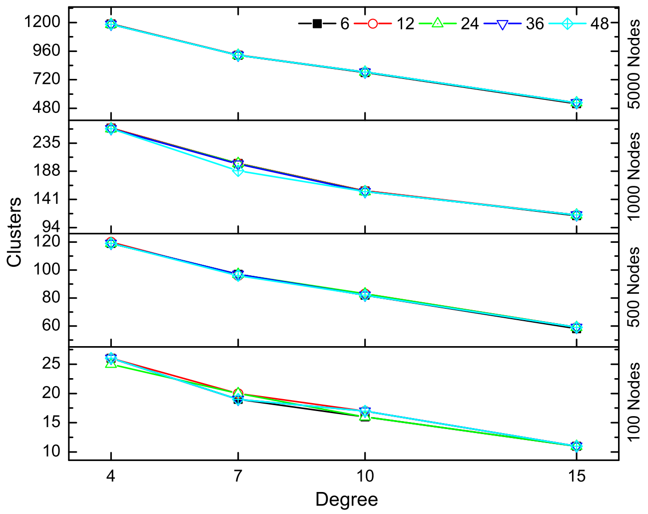

Fig. 12. Impact of *D* on the number of clusters produced by *RanMIS* for backbone construction w.r.t. network size and its density.

3.4. Simulation results

In order to evaluate the performance of *RanMIS*, we selected two graph-theoretic ad hoc networks clustering algorithms, namely the *WuLi* and *DCA*, both of which create dominating sets, the former produces a connected dominating set and the latter produces a maximum independent set. These algorithms are simple, practical, extremely popular in the clustering community[3] and comprise the base for the development of many similar clustering algorithms [28]. We performed a series of experiments to compare the performance of *RanMIS* against these algorithms. The first set of experiments concerns the tuning of *RanMIS* with respect to its parameters, and then it follows its comparison against its competitors.

3.4.1. Tuning of RanMIS

The original article which introduced *RanMIS* [23], described two parameters that control its operation and affect its overall performance. The first of these parameters, *D*, concerns the estimated size of neighborhood of each node, while

[3] They both have an excessive number of citations. See http://scholar.google.com/scholar?cites=7672109277762563412&as_sdt=2005&sciodt=0,5&hl=en and http://scholar.google.com/scholar?cites=13992908027776879246&as_sdt=2005&sciodt=0,5&hl=en.

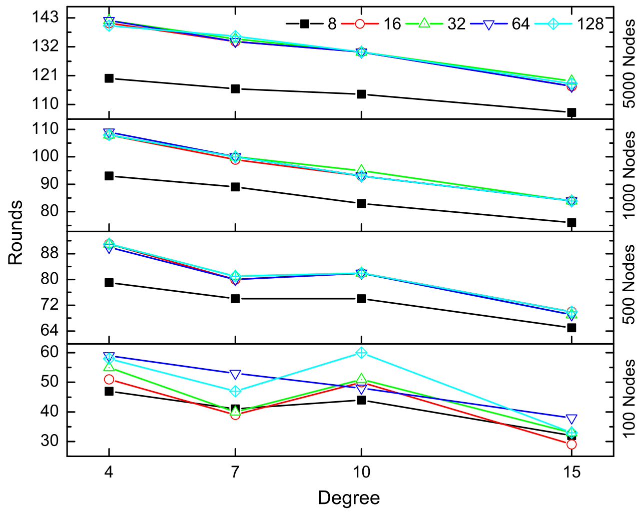

Fig. 13. Impact of *M* on the number of rounds required by *RanMIS* to complete backbone construction w.r.t. network size and its density.

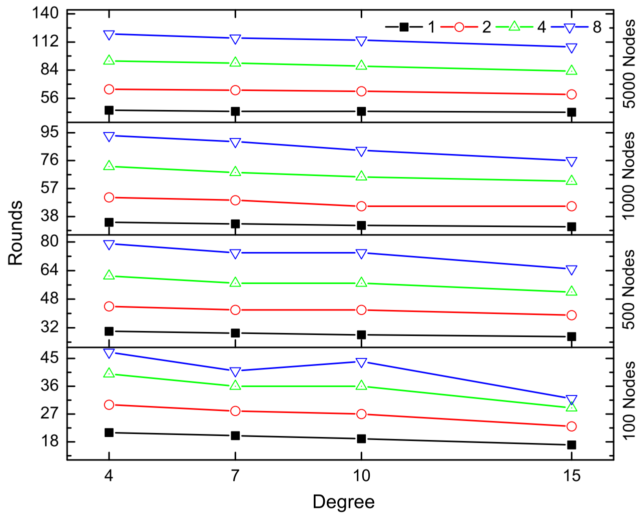

Fig. 14. Impact of *M* on the number of rounds required by *RanMIS* to complete backbone construction w.r.t. network size and its density.

the second, *M*, is a constant related to the number of rounds required for the algorithm to complete the backbone construction. These parameters are independent and thus we have to make an initial choice for one of them (e.g., for *D*) and investigate the other (e.g., *M*), and vice versa.

Impact of parameter ***D*****.** To evaluate the impact of parameter *D* on *RanMIS*'s performance, we conducted a series of experiments for various values of $D, 6, 12, 24$ 36, and 48 and for various performance measures. *D* controls the maximum number of rounds *RanMIS* has in order to complete the backbone construction and it is consistent with the size of each node's neighborhood. That is because in a large neighborhood the intuition is that the likelihood to have a simultaneous transmission and a resultant collision between two adjacent transmitting nodes is increased, and consequently more protocol rounds will be required by the affected nodes in order to decide about their status.

In Fig. 10, we see the number of rounds required by *RanMIS* to complete backbone construction for various values of *D* w.r.t. network size and its density.

The first observation is that, in accordance with our intuition, when the value of *D* increases, the number of rounds required by *RanMIS* to complete backbone construction increases for every network size. This consequence derives from the fact that there is a direct relation between the probability a node has in order to be chosen as a CH candidate and the setting of parameter *D* (cf. Fig. 1). The higher the setting of *D* the lesser the probability for a node to become a CH and if

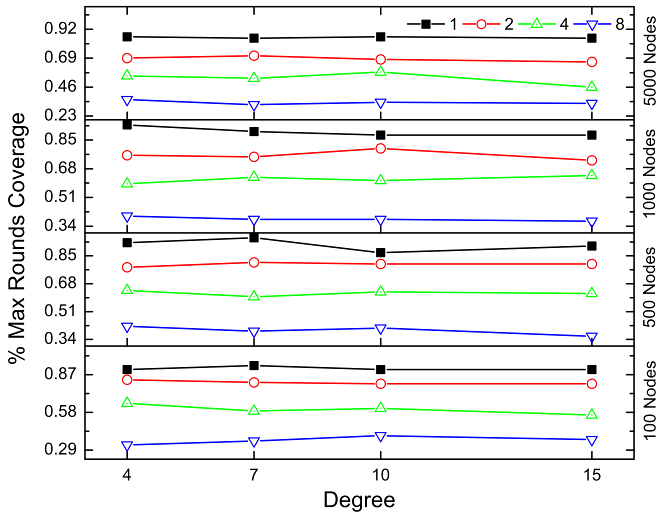

Fig. 15. % percentage coverage of Max number of rounds available to *RanMIS* in order to converge w.r.t. *M* parameter setting.

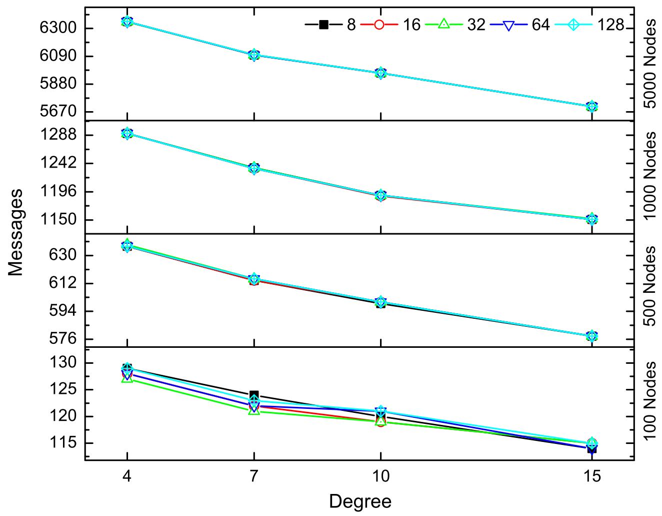

Fig. 16. Impact of *M* on the number of messages exchanged by *RanMIS* for backbone construction w.r.t. network size and its density.

so, the probability for a neighboring node to become a CH is also small (thus, we avoid any unwanted message retransmissions due to collisions). The second observation is that while the number of rounds is almost irrelevant to network density for a given network size, the number of rounds increases linearly to the network size. This is due to the fact that more clusters will be developed and therefore the competition for becoming a clusterhead increases. It is obvious that the less the required rounds to complete the backbone is, the best for the protocol is (namely in Fig. 10 this observation favors *D* setting of 6 and 12 to the other). Though, to conclude which is the best choice for parameter *D*, we must examine also the number of transmitted messages and the number of clusters created for the various choices of *D* (cf. Figs. 11 and 12). Thus, we do not conclude at this point that the best value for *D* is 6.

In Fig. 11, we see the number of messages exchanged by *RanMIS* for backbone construction for various values of *D* w.r.t. network size and its density.

The situation here is reversed with respect to what we saw in Fig. 10. In general, the number of messages required by *RanMIS* to complete the backbone construction process decreases for increasing values of *D*. Therefore, so far the best choice for *D* is the one that achieves a balance between the number of rounds and the messages transmitted, i.e., $D = 24$ (we will revisit this issue when commenting the next figure). Additionally, examining the figure, we observe a decreasing number of transmitted messages for increasing network density (for every network size). This is because when a node declares itself as a CH, then all of its neighbors (and there are too many nodes in dense networks) immediately attach themselves to this node.

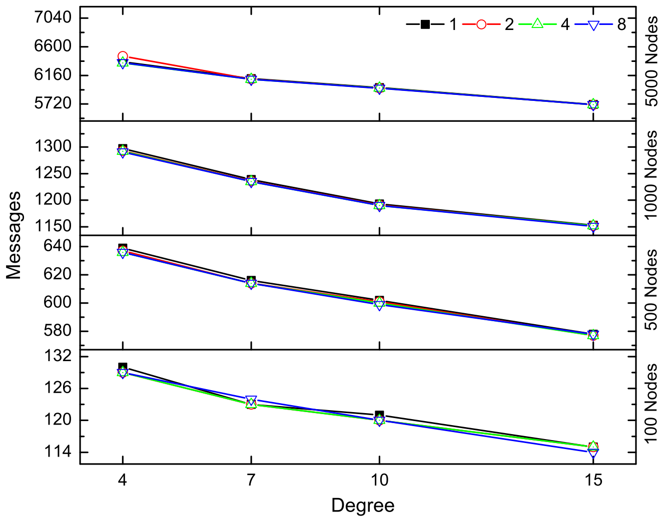

Fig. 17. Impact of *M* on the number of messages exchanged by *RanMIS* for backbone construction w.r.t. network size and its density.

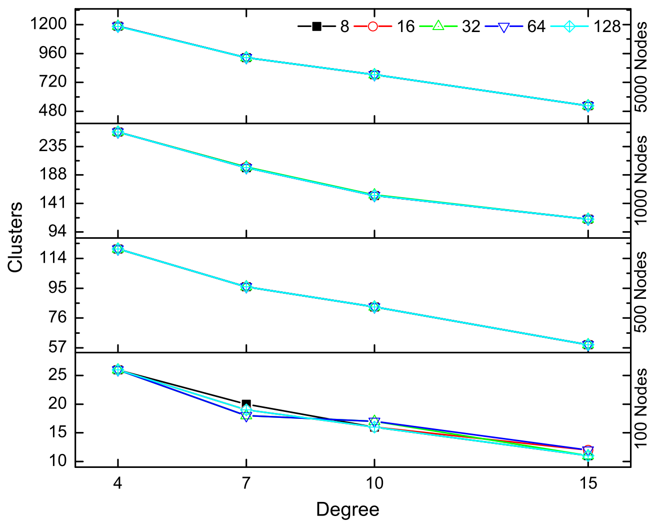

Fig. 18. Impact of *M* on the number of clusters produced by *RanMIS* w.r.t. network size and its density.

The final observation is that the relative ordering of the performance curves for various values of *D* is maintained across different densities (with some statistically insignificant variation for very small networks).

In Fig. 12, we see clusters produced by *RanMIS* for backbone construction w.r.t. network size and its density for various values of *D*.

The main observation is that the number of clusters produced is not affected by parameter *D*. However, protocol execution drives to decreasing number of produced clusters w.r.t. increasing network density for every network size, and the explanation for that is similar to the one for the previous figure.

Overall, *a value of* $D = 24$ *is the most appropriate choice* since it achieves a good tradeoff between network backbone construction delay (due to many rounds) and the transmitted messages.

Impact of parameter ***M*****.** To evaluate the impact of constant *M* on *RanMIS*'s performance, we conducted a series of experiments for various values of $M = 2^x$ ($x = 0$–7), and for various performance measures. For these sets of experiments, the value of *D* was set equal to 24 (cf. subsection *Impact of parameter D*.)

Initially, we examined the impact of *M* on the number of rounds required by *RanMIS* to complete the backbone construction. In Fig. 13, it is clear that an $M = 8$ setting supersedes any other for practically all network instances (except perhaps for 100 nodes), basically because less work in the inner loop of *RanMIS* (see Fig. 1), signifies less residency in a low probability

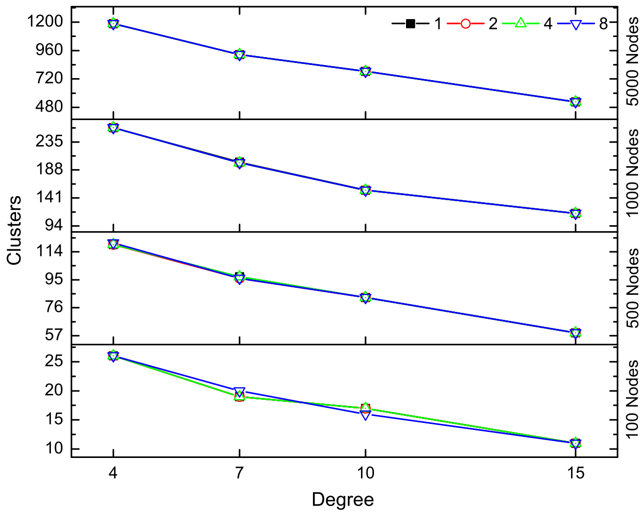

Fig. 19. Impact of *M* on the number of clusters produced by *RanMIS* w.r.t. network size and its density.

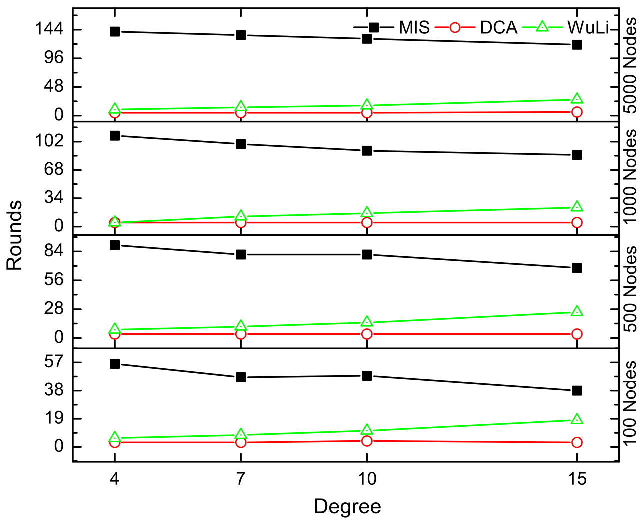

Fig. 20. Rounds required by each competitor algorithm to complete backbone construction w.r.t. network size and its density.

state for a node to become a CH. Practically, with small settings of *M*, we bias the system to converge faster. The same observation holds in Fig. 14, for even smaller values of *M* (namely 1–8).

The disadvantage though, of a small *M* setting is that it might create severe competition for the role of CH among neighboring nodes, and since *M* affects also the maximum number of rounds that are available to the protocol for backbone construction, it is possible to drive it not to converge. Fig. 15 reflects our thoughts, where for small settings of *M* (e.g 1) more than 90% of the available rounds are used by *RanMIS* in order to converge.

In Figs. 16 and 17, we see the impact of *M* on the number of messages exchanged by *RanMIS* to complete the backbone. The observation is that for practically all network instances the number of transmitted messages is unrelated to *M*.

In Fig. 18 and 19, we see the impact of *M* on the number of clusters produced by *RanMIS*. Similarly, the number of clusters produced is not affected by *M*.

Overall, it seems that *the authors' suggestion of* $M = 34$ *is not the best choice*; it is only marginally good for small networks (100 nodes) when we want to minimize the number of transmitted messages – the difference though is not really significant. We concluded though, that we should be conservative with the setting of *M*, especially when we have to deal with real life network topologies. Nevertheless, in the rest of this article, we follow the *RanMIS*'s creators suggestion and set the value of *M* equal to 32.

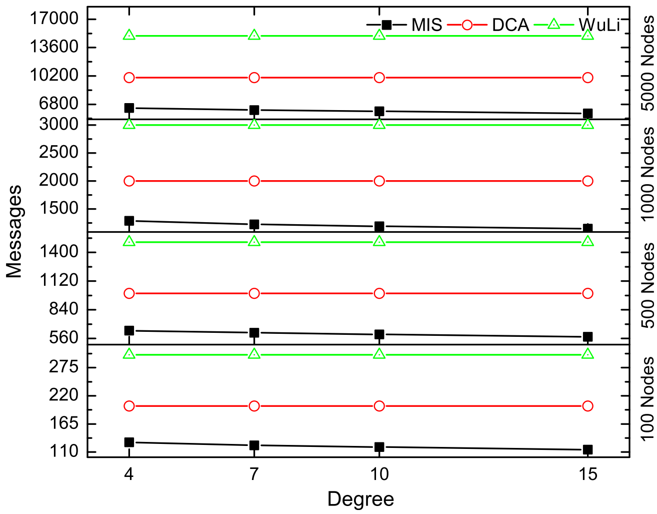

Fig. 21. Messages required by each competitor algorithm to complete backbone construction w.r.t. network size and its density.

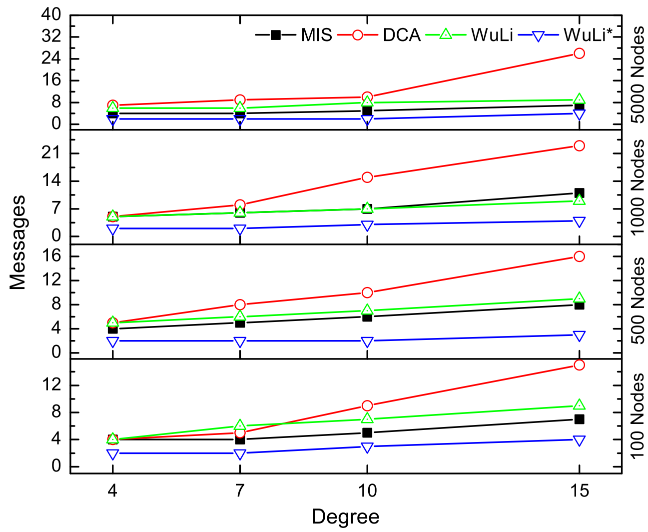

Fig. 22. Impact of network size and its density on messages required to rebuild locally the broken backbone in case of a CH failure.

3.4.2. Comparison of competing clustering protocols

The next paragraphs unveil the nature of the competing protocols concerning their operational cost, their network topology uniqueness and their backbone robustness.

Results concerning the protocol cost. In this series of experiments, we measured the number of rounds required by each protocol to complete and also the total number of messages transmitted. In Figs. 20 and 21, we see the relative comparison of the algorithms.

As far as the number of protocol rounds is concerned, we observe that *DCA* is the clear winner for all network instances. The performance of the algorithms is consistent with the what the theory predicts for them; thus we dot not comment further on this issue.

As far as the number of transmitted messages for backbone construction is concerned, we observe that *RanMIS* is now the clear winner confirming its theoretical behavior. Its performance gap from the other competitors is stable across all network topologies. *DCA* sends around 40% more messages than *RanMIS* does, and *WuLi* sends around 60% more messages than *RanMIS*. Therefore, even though it takes more time to *RanMIS* to construct the backbone due to its probabilistic nature, it does this with far less messages than its competitors.

As far as the number of transmitted messages for backbone reconstruction when a node dies, are presented in Fig. 22. In this figure, apart from the three competitors we present also the proposed improvement to *WuLi* which was described in

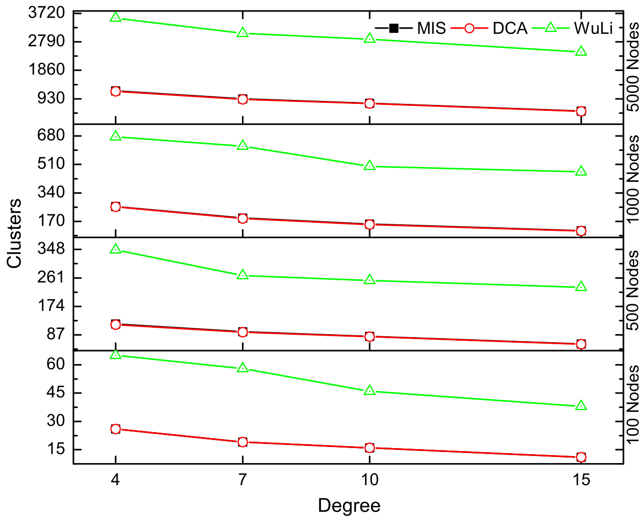

Fig. 23. Clusters produced by each competitor algorithm w.r.t. network size and its density.

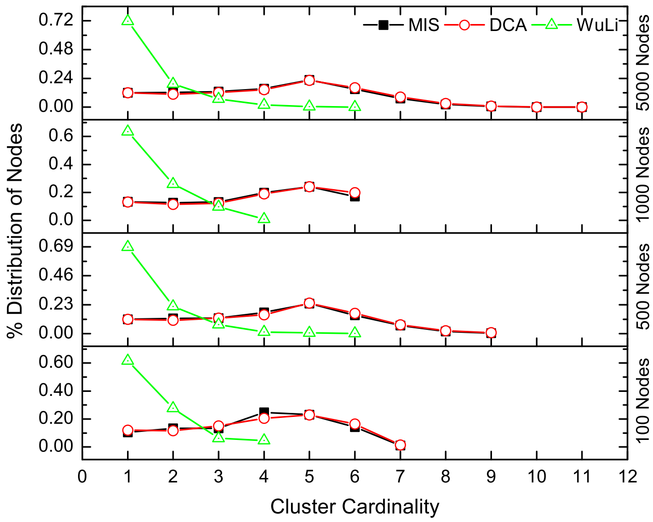

Fig. 24. Distribution of nodes in clusters after backbone construction w.r.t. network size and a density of $D = 4$.

page 16. The first observation is that this proposed improvement is the best performing algorithm which is expected, since the central idea behind its development was how to least modify the existing network backbone, and therefore transmit as few messages as possible. The second best performing algorithm is *RanMIS*, which is also expected since it is message optimal. The performance of all algorithms deteriorates for denser networks, because a single node's failure affects a lot of neighboring nodes, but their performance is not significantly affected by the network size.

Results concerning the backbone description. In Figs. 23–30 we present the performance of the protocols for the metrics belonging to the category of "backbone description".

The number of generated clusters by each protocol is presented in Fig. 23. The first observation is that *RanMIS* and *DCA* produce almost the same number of clusters; even though their difference is indistinguishable in the figure, their difference is less than 1% which can be attributed to statistical variation. The reason for this resemblance is that both algorithms are building maximum independent sets using either a weight for each node (assigned dynamically or statically) in the case of *DCA* or using randomization in choice in the case of *RanMIS*. On the other hand, *WuLi* produces too many clusters, since it is mandatory to create *connected* dominating sets, whereas the former two algorithms are building plain dominating sets. Finally, we observe that the number of generated clusters reduces with increasing density, since in dense networks more nodes are able to find a CH (i.e., dominator) in their 1-hop neighborhood.

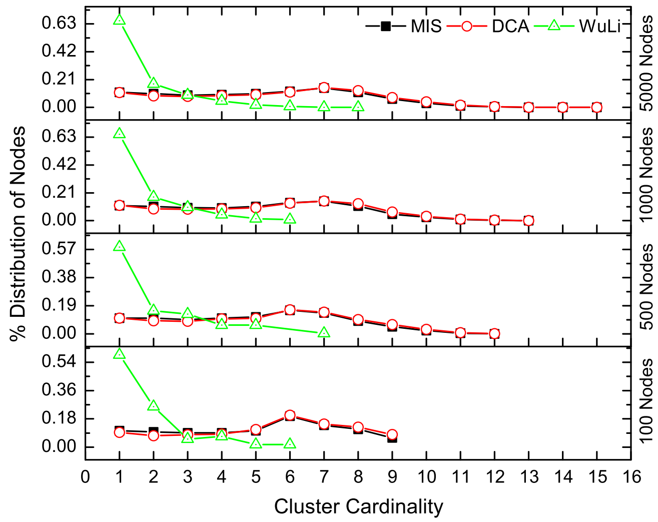

Fig. 25. Distribution of nodes in clusters after backbone construction w.r.t. network size and a density of $D = 7$.

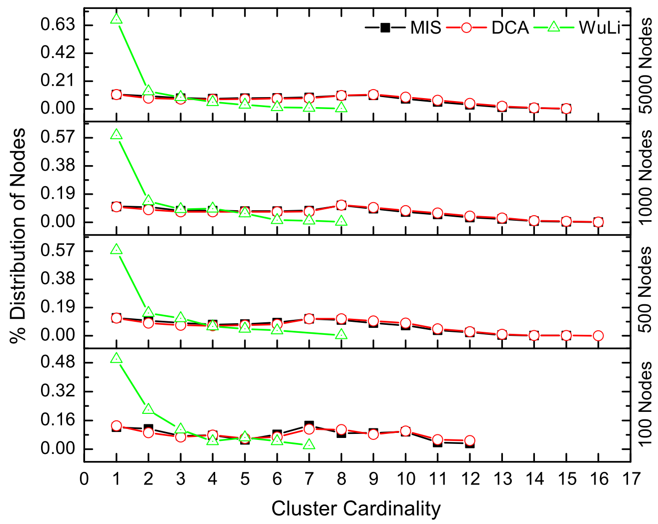

Fig. 26. Distribution of nodes in clusters after backbone construction w.r.t. network size and a density of $D = 10$.

Next, we evaluated the cluster cardinality distribution for various densities of the network. The results are illustrated in Figs. 24–27. At this point we should emphasize that this distribution should in general be 'bell-shaped' or look like any distribution with probability mass concentrated around the average node degree of the topology. In Figs. 24 and 25, we see that this property is achieved by *RanMIS* and *DCA* for sparse networks, but it is not achieved for dense networks (in Figs. 26 and 27). In these last two figures, it seems that a significant percentage of clusters – around 22% of them – contain very few nodes, i.e., 1 or 2, including the clusterhead. On the other hand, *WuLi* exhibits an undesirable pattern across all network topologies, since more than 70% of its clusters contain 1 or 2 nodes, which is also a consequence of the large numbers of clusters produced (see Fig. 23).

The diameter[4] length of the produced backbone by each protocol is presented in Fig. 28 (in meters) and in Fig. 29 (in hops). The common observation is that *WuLi* is the best performing algorithm which is due to the fact that it produces connected dominating sets and thus there are no gateway nodes intervening among CHs so as to increase the diameter. The second observation is that the diameter in general decreases with increasing network density, since in dense networks it easier to find short routes for any pair of nodes. In some cases where the topology is peculiar, the diameter increases in denser networks (e.g., network with 500 nodes with density equal to 7).

[4] The largest of the shortest paths for any pair of nodes.

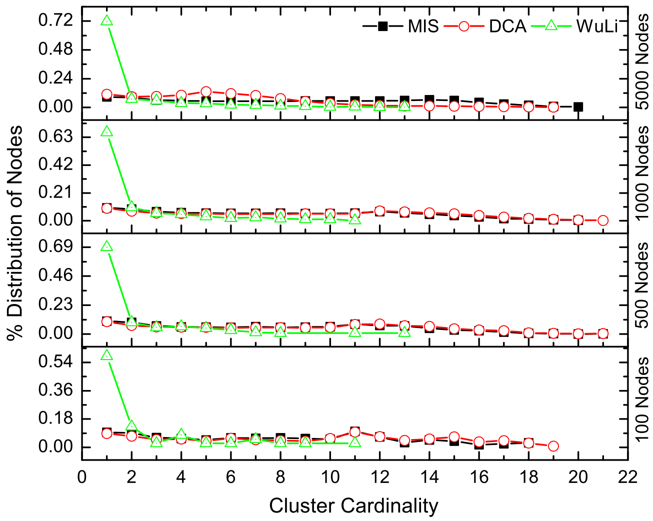

Fig. 27. Distribution of nodes in clusters after backbone construction w.r.t. network size and a density of $D = 15$.

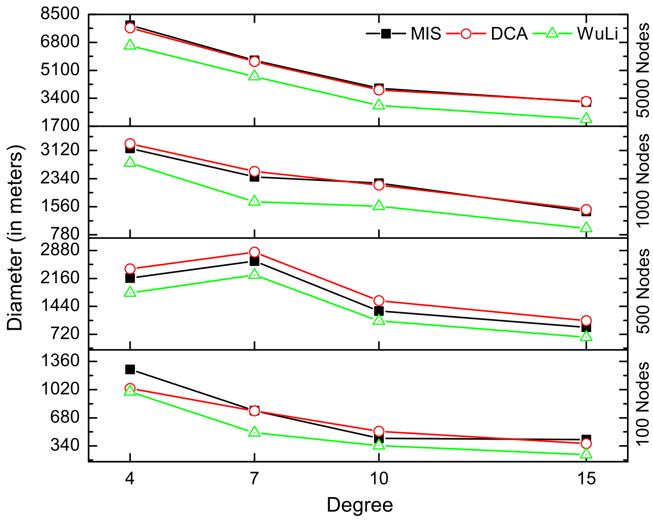

Fig. 28. Network diameter diversification (in meters) after backbone construction w.r.t. network size and its density.

In Fig. 30 we illustrate the percentage of adjacent CHs that reside at a distance of 3-hops from each other. Apparently, *WuLi* produces clusterheads at 1-hop distance from each other, and therefore we do not include it in the plot. For the other two algorithms, which produce maximum independent sets, it holds by the definition of the MIS, that any two adjacent CHs will be either at a distance of 2 hops or at a distance of 3 hops, with the latter being the preferred one (see paragraph on 'Interclusterhead distance distribution' definition, Section 3.3). We see that both algorithms achieve approximately the same performance – the gap among them is statistically insignificant. In particular, we see that the majority (more than 50%) of adjacent CHs in almost all network topologies, are 3-hops away. The only exception appears for very small and very sparse networks (i.e., 100 nodes with degree 4), where this percentage drops to 48%; this is due to the fact that there are not too many links among the nodes to establish 3-hop distance among neighboring CHs.

Robustness evaluation. Next, we evaluated the robustness of the resulting backbone for various densities of the network. The results are illustrated in Figs. 24–27. In Fig. 31 we illustrate the number of non-CHs nodes that must be removed to lose backbone connectivity. For *WuLi* which has no gateway nodes, the plot simply shows the total number of clusters members. For the other two algorithms, their performance is similar for all topologies, but for the large networks, where *RanMIS* exhibit a more robust behavior than *DCA*, with their performance gap widening for denser networks (namely 4–7 degree).

Finally, we examined the backbone robustness of the competing protocols and the results are illustrated in Figs. 32 and 22. For all but the very dense networks, *RanMIS* is the best performing algorithm when the number of backbone nodes removed or the number of transmitted messages is considered. Though, for very dense networks (density equal to 15),

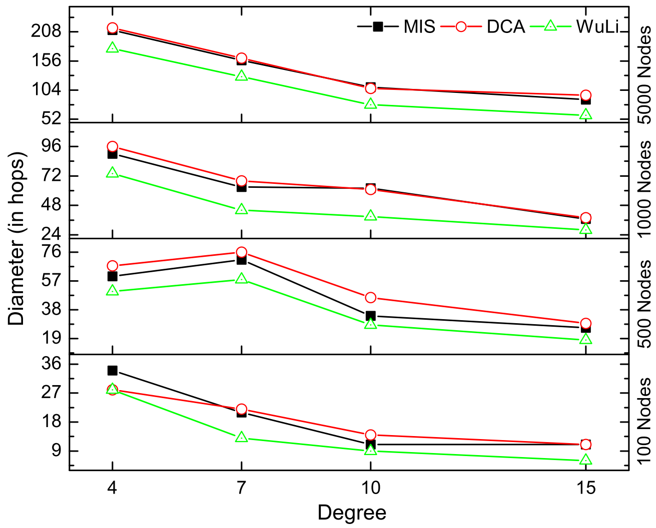

Fig. 29. Network diameter diversification (in Hops) after backbone construction w.r.t. network size and its density.

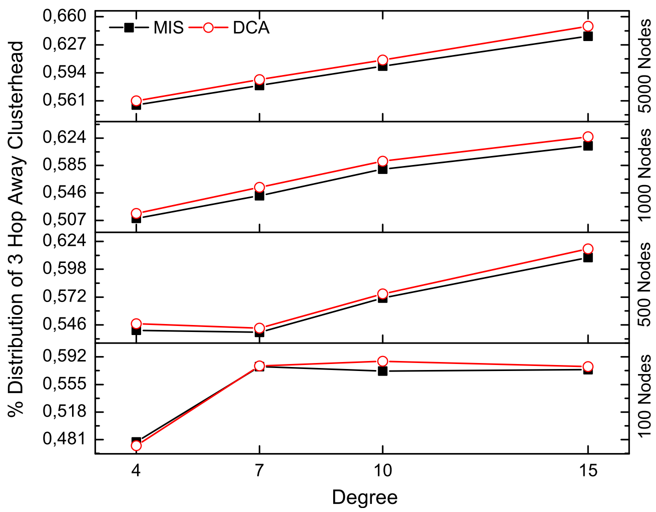

Fig. 30. CH distance distribution after backbone construction w.r.t. network size and its density.

RanMIS's performance is inferior to *WuLi* (and its variation) when considering the number of backbone nodes allowed to be removed. When examining the number of transmitted messages, *RanMIS* loses only, as expected, by *WuLi*'s variation whose goal (as mentioned before) is to violate *WuLi* principle of CHs selection in order to keep the message overhead as low as possible.

Overall comparison of the protocols. Here we briefly summarize the most significant conclusions from the experiments. The value of parameter *D* controls a tradeoff between the number of rounds needed to complete *RanMIS* and transmitted messages; a value equal to 24 seems to achieve a good balance. The value of parameter *M* does have an impact on the number of rounds, and it seems that a value equal to 8 should be the preferred choice. None of these parameters has an impact on the number of cluster produced. Concerning *RanMIS*'s comparison with the rest of the protocols, we found that *RanMIS* is the worst when the number of rounds is considered, but the best from the perspective of transmitted messages; its number of rounds and messages increases for larger networks, and drops for denser networks. *RanMIS* has a graceful distribution for the distribution of cluster cardinality, and for the intercluster distance, but it produces relatively 'long' backbones. Finally, *RanMIS* is quite robust when backbone nodes die being the most resilient protocol for sparse networks, but not the best one for very dense networks, staying behind *WuLi**.

RanMIS incorporates some pros that are beneficial to ad hoc networks. A *RanMIS* network topology would encompass considerable availability, because it is efficient in the message complexity that involves backbone construction and maintenance

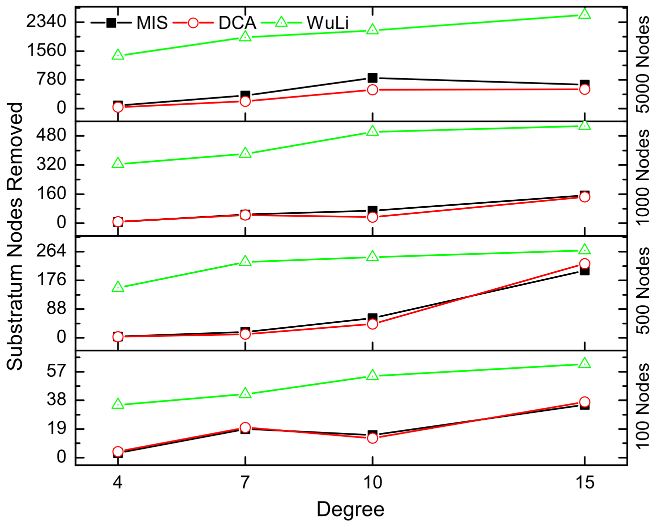

Fig. 31. Non-CH nodes (gateways) removed before backbone recalculation w.r.t. network size and its density.

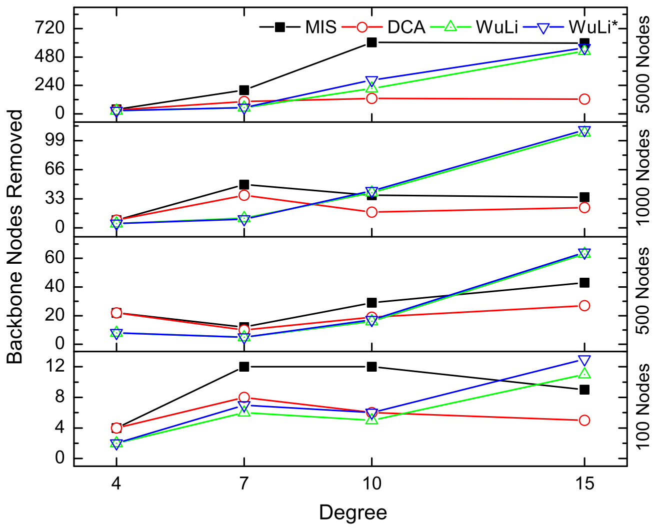

Fig. 32. CH nodes removed before backbone recalculation w.r.t. network size and its density.

(see Figs. 21 and 22). Additionally, it is sustainable, in the sense that it can continue operating for an extended amount of time, compared to its competitors, because of its resilience in node failures (see Fig. 32). The 'intracluster redundancy', as it is described by the network cluster cardinality (Figs. 24–27), is a very useful network attribute, because it allows for adopting a round-robin CH selection method, so as to put some nodes in a sleep mode and thus increase network longevity.

4. Conclusion

Ad hoc network node clustering is a very practical and useful topology management approach to reduce the communication overhead and exploit data aggregation in wireless ad hoc networks. Despite the very large literature on the subject, still there is room for the development of protocols that will achieve to run in very few rounds (and thus incur small delay) and at the same time transmit very few messages (thus addressing the broadcast storm problem). In this article, we investigated by detailed simulation the performance of *RanMIS* using a new exhaustive evaluation framework. The results confirmed the message-optimality of *RanMIS*, and exhibited its drawback when the number of rounds is concerned based on the original suggestions. Overall, we strongly believe that *RanMIS* could be a viable clustering option for wireless ad hoc networks.

The properties of a network topology constructed by *RanMIS* can be used by applications such as habitat monitoring [29], by disaster relief [30] or law enforcement operations management [31], or even military applications, such as the battle field communications, the early warning, the security surveillance, the identification, the shooter position estimation [32], the battlefield situational awareness [33], the battle damage assessment, the targeting [34], the nuclear-biological-chemical hazard awareness [35], where quasi stationary nodes are involved.

Acknowledgements

Research supported by the project "REDUCTION: Reducing Environmental Footprint based on Multi-Modal Fleet management System for Eco-Routing and Driver Behaviour Adaptation", funded by the EU.ICT program, Challenge ICT-2011.7.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.simpat.2014.06.012.

References

[1] Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, F. Kuhn, Beeping a maximal independent set, in: Proceedings of the International Conference on Distributed Computing (DISC), Lecture Notes on Computer Science, vol. 6950, 2011, pp. 32–50.
[2] A. Amis, R. Prakash, T. Vuong, D.T. Huynh, Max-min d-cluster formation in wireless ad hoc networks, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), 2000, pp. 32–41.
[3] V. Anitha, M.P. Sebastian, (k, r)-Dominating set-based, weighted and adaptive clustering algorithms for mobile ad hoc networks, IET Commun. 5 (13) (2011) 1836–1853.
[4] S. Basagni, Distributed clustering for ad hoc networks, in: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN), 1999, pp. 310–315.
[5] S. Basagni, M. Mastrogiovanni, A. Panconesi, C. Petrioli, Localized protocols for ad-hoc clustering and backbone formation: a performance comparison, IEEE Trans. Parallel Distrib. Syst. 17 (4) (2006) 292–306.
[6] M. Chatterjee, S.K. Das, D. Turgut, WCA: a weighted clustering algorithm for mobile ad-hoc networks, J. Cluster Comput. 5 (2002) 193–204.
[7] J. Ghaderi, L.-L. Xie, X. Shen, Hierarchical cooperation in ad hoc networks: optimal clustering and achievable throughput, IEEE Trans. Inform. Theory 55 (8) (2009) 3425–3436.
[8] S. Leng, Y. Zhang, H.-W. Chen, L. Zhang, K. Liu, A novel k-hop compound metric based clustering scheme for ad hoc wireless networks, IEEE Trans. Wireless Commun. 8 (1) (2009) 367–375.
[9] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, IEEE Trans. Parallel Distrib. Syst. 13 (1) (2002) 14–25.
[10] J. Wu, H. Li, A dominating-set-based routing scheme in ad-hoc wireless networks, Telecommun. Syst. 18 (2001) 13–36.
[11] C. Tselikis, S. Mitropoulos, N. Komninos, C. Douligeris, Degree-based clustering algorithms for wireless ad hoc networks under attack, IEEE Commun. Lett. 16 (5) (2012) 619–621.
[12] J.Y. Yu, P.H.J. Chong, A survey of clustering schemes for mobile ad hoc networks, IEEE Commun. Surv. Tutor. 7 (2005) 32–48.
[13] N. Dimokas, D. Katsaros, Y. Manolopoulos, Energy-efficient distributed clustering in wireless sensor networks, J. Parallel Distrib. Comput. 70 (4) (2010) 371–383.
[14] O. Younis, M. Krunz, S. Ramasubramanian, Node clustering in wireless sensor networks: recent developments and deployment challenges, IEEE Netw. Mag. 20 (3) (2006) 20–25.
[15] M. Almullaa, K. Abrouguib, A.A. Boukerche, LEADMesh: design and analysis of an efficient leader election protocol for wireless mesh networks, Simul. Modell. Pract. Theory 36 (2013) 22–32.
[16] L. Maglaras, D. Katsaros, Distributed clustering in vehicular networks, in: Proceedings of the International Workshop on Vehicular Communications and Networking (VECON), 2012, pp. 601–607.
[17] M. Ni, Z. Zhong, D. Zhao, MPBC: a mobility prediction-based clustering scheme for ad hoc networks, IEEE Trans. Vehic. Technol. 60 (9) (2011) 4549–4559.
[18] S. Vodopivec, J. Bester, A. Kos, A survey on clustering algorithms for vehicular ad-hoc networks, in: Proceedings of the International Conference on Telecommunications and Signal Processing (TSP), 2012, pp. 52–56.
[19] A. Cuzzocrea, A. Papadimitriou, D. Katsaros, Y. Manolopoulos, Edge betweenness centrality: a novel algorithm for QoS-based topology control over wireless sensor networks, J. Network Comput. Appl. 35 (4) (2012) 1210–1217.
[20] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, ACM Wireless Netw. 8 (5) (2002) 481–494.
[21] W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Trans. Wireless Commun. 1 (4) (2002) 660–670.
[22] O. Younis, S. Fahmy, HEED: a hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks, IEEE Trans. Mobile Comput. 3 (4) (2004) 366–379.
[23] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, Z. Bar-Joseph, A biological solution to a fundamental distributed computing problem, Science 331 (6014) (2011) 183–185.
[24] B. Das, V. Bharghavan, Routing in ad hoc networks using minimum connected dominating sets, in: Proceedings of the IEEE International Conference on Communications (ICC'97), 1997, pp. 376–380.
[25] F. Atay, I. Stojmenovic, H. Yanikomeroglu, Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks, in: Proceedings of the IEEE Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2008, pp. 597–615.
[26] P. Crucitti, V. Latora, M. Marchiori, A. Rapisarda, Error and attack tolerance of complex networks, Phys. A: Stat. Mech. Its Appl. 340 (1–3) (2004) 388–394.
[27] R.M. Salles, D.A.J. Donato, A. Marino, Strategies and metric for resilience in computer networks, The Comput. J. 55 (6) (2012) 728–739.
[28] K. Erciyes, O. Dagdeviren, D. Cokuslu, D. Ozsoyeller, Graph-theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks – survey, Appl. Comput. Math. 6 (2) (2007) 162–180.
[29] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, J. Anderson, Analysis of wireless sensor networks for habitat monitoring, in: C.S. Raghavendra, K.M. Sivalingam, T. Znati (Eds.), Wireless Sensor Networks, 2004, pp. 399–423.
[30] E. Cayirci, T. Coplu, SENDROM: sensor networks for disaster relief operations management, ACM Wireless Netw. 13 (2007) 409–423.

[31] C. Chung-Kuo, J. Huang, Video surveillance for hazardous conditions using sensor networks, in: Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC), 2004, pp. 1008–1013.
[32] P. Volgyesi, G. Balogh, A. Nadas, C.B. Nash, A. Ledeczi, Shooter localization and weapon classification with soldier-wearable networked sensors, in: Proceedings of the ACM International Conference on Mobile Systems, Applications and Services (MobiSys), 2007, pp. 113–126.
[33] G. Hua, Y.-X. Li, X.-M. Yan, Research on the wireless sensor networks applied in the battlefield situation awareness system, in: Advanced Research on Electronic Commerce, Web Application, and Communication, vol. 144, 2011, pp. 443–449.
[34] X. Ji, H. Zha, Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), 2004, pp. 2652–2661.
[35] J.K. Hart, K. Martinez, Environmental sensor networks: a revolution in the earth system science?, Earth-Sci Rev. 78 (3–4) (2006) 177–191.