WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2008; **8**:295–308
Published online 3 December 2007 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.578

Mitigating the impact of node mobility on ad hoc clustering

Rituparna Ghosh and Stefano Basagni*,†
Department of Electrical and Computer Engineering, 312 Dana Northeastern University, Boston, MA, U.S.A.

Summary

This paper explores the impact of node mobility on distributed and mobility adaptive clustering (DMAC), a typical clustering protocol for mobile ad hoc networks. In particular, in this paper we evaluate the cost of maintaining the DMAC clustering structures when the nodes move according to three different mobility models, namely, the random way point model, the Brownian motion and the Manhattan mobility model. Via ns2-based simulations, we have observed that the mobility models have different impact on protocol performance. The general trend, however, appears to be the same for networks of increasing size. The second contribution of this paper concerns investigating ways of mitigating the impact of mobility on the clustering structure and hence over the overall network performance. We consider a generalization of DMAC (GDMAC) where rules are established to decrease the number of cluster updates. Via simulation we have observed that GDMAC is effective in reducing the clustering overhead imposed by mobility, and the corresponding maintenance cost. Copyright © 2007 John Wiley & Sons, Ltd.

KEY WORDS: wireless ad hoc networks; clustering protocols; mobility management

1. Introduction

A mobile ad hoc network is an infrastructure-less multi-hop wireless network where each node acts as a router, thus relaying packets of other nodes [1]. These kinds of network are suitable when an instant communication infrastructure is needed and it is not viable to build *ex novo*, or repair, a wired communication system. Limited bandwidth, scarcity of resources (e.g., battery power), and unpredictable and rapid node movements are some of the characteristics of ad hoc network. Therefore, among the major challenges in designing ad hoc protocols there is the requirement of taking into account the dynamic nature of the network topology, where links between neighboring nodes (i.e., nodes that are in each other transmission range) are added and removed all the time.

Among the crucial challenges posed by the ad hoc architecture, there is that of defining scalable protocols. As in the case of wired networks, hierarchical approaches have been proposed for routing that make possible to 'simplify' the overall network topology now seen as a set of *clusters*. Nodes are partitioned into groups, each with a *clusterhead* that coordinates the cluster formation process and some *ordinary nodes* which rely on the clusterhead for inter-cluster communications. The idea common to basically all clustering protocols is to cluster together nodes that are in physical proximity, thereby providing the network with a hierarchical organization which is smaller in scale, and hence simpler to manage.

Clustering for ad hoc networks has been widely investigated and a host of solutions are available. Thorough surveys of ad hoc clustering protocols, as well as

*Correspondence to: Stefano Basagni, Department of Electrical and Computer Engineering, 312 Dana Northeastern University, Boston, MA, U.S.A.
†E-mail: basagni@ece.neu.edu

a performance-based comparison among some of the most representative solutions, can be found in References [2] and [3]. The vast majority of clustering protocols deal with static or quasi-static networks, or assume no node mobility while clustering is taking place. Mobile clustering is still largely uncharted territory.

With this paper, we provide a first account of the effect of different models of mobility on a typical ad hoc clustering organization. In particular, we consider a basic clustering algorithm, the distributed and mobility adaptive clustering (DMAC) [4] protocol, and we evaluate its performance when the network nodes move according to three different mobility patterns, which capture quite a variety of possible movement in realistic situations, some of which are listed in Reference [5]. More specifically, we consider the random way point mobility model [6], the random walk mobility model (Brownian motion) [7], and the Manhattan mobility model [8]. Among the variety of ad hoc clustering protocols, DMAC is chosen for its simplicity and efficiency in building the clusters and in maintaining the cluster structure in the face of node mobility.

The contribution of this paper goes beyond the assessment of the impact of mobility on an ad hoc clustering organization. Based on our observation, we have proceeded to define methods for limiting the effect of mobility on clustering. In particular, we have enhanced DMAC with primitives that decrease the number of the nodes that are elected to be clusterheads (elections) and of the nodes that switch cluster (re-affiliations). We termed the resulting protocol *generalized DMAC* (GDMAC).

We have evaluated GDMAC and compared it to DMAC through extensive ns2-based simulations. Our observations confirm that GDMAC is effective in limiting the detrimental effect of mobility on the DMAC-based clustering organization. In particular, and independently of the speed of the nodes, GDMAC outperforms DMAC in all investigated metrics, hence being able to reduce the time the network is 'blocked for re-clustering' and at the same time the cost associated to re-clustering itself.

The aim of this paper is not a comparative study or a performance evaluation of different clustering techniques with respect to mobility. Rather, we aim at evaluating how much mobility affects the hierarchical organization of ad hoc networks and we propose simple and efficient techniques for limiting the impact. In our investigation, we take into account realistic network conditions such as packet loss (due to collisions), increasing network density, unpredictable link failures, etc. All the performed experiments confirm the intuitions and expectations behind the basic concept that was proposed in Reference [9]. The GDMAC protocol is indeed able to stabilize the DMAC cluster structure under different network dynamics.

The metrics we considered in our study concern the part of clustering performance that is mostly affected by the mobility of the nodes. More specifically, we investigate the following metrics (all averages): *cluster density*, that is, the number of clusters. *Cluster stability*, measured in terms of cluster lifetime (how long a node is a clusterhead), affiliation time of an ordinary node to a particular cluster (residence time), and number of status changes per node (election to clusterhead and re-affiliation to another clusterhead). *Message complexity per node*. This is the total number of clustering-related messages sent by a node.

The remainder of the paper is organized as follows. In the next section, we describe related work in the area of mobile ad hoc clustering. Section 3 introduces DMAC and describes GDMAC in details. Section 4 shows the comparative performance evaluation of DMAC and GDMAC in networks where nodes move at different speeds according to the three selected mobility models. Finally, Section 5 concludes the paper.

2. Related Work

Clustering protocols for ad hoc networks are roughly divisible into two main classes, *node centric* and *cluster centric*. In node-centric solutions, clusters are created around the clusterheads. The clusterheads form a *dominating set* of the network nodes, and are responsible for providing basic network function on behalf of and for their ordinary nodes. Typical functions are controlling channel access, performing power measurements, guaranteeing bandwidth for real-time traffic, and general coordination of intra and inter-cluster communications. This usually induces on the clusterhead increased overhead with respect to the overhead imposed on the ordinary nodes. A generalized load-balancing solution for resolving this asymmetry is given in Reference [10], where a circular queue is maintained that distributes the responsibility of acting as clusterheads evenly among all the cluster nodes (clusterhead rotation). A number of algorithms have been proposed for dealing with cluster formation in ad hoc networks which belong to node centric clustering. Most of the algorithms that have been proposed are based on two fundamental algorithms, lowest ID [11] and highest degree [12]. A generalization of this idea, where each node is at most d

DOI: 10.1002/wcm

hops from a clusterhead is proposed in Reference [13], where a protocol is given that efficiently builds disjoint clusters in which each node is at most $d \geq 1$ hops away from its clusterhead. The network is clustered in a number of rounds which is proportional to d, which favorably compares to most of the previous solutions when d is small. Lin *et al*. [14] present a *node*-ID-based algorithm which is adapted from the early LCA algorithm of Reference [11], as well as on the degree-based algorithm from Reference [15]. In Reference [14] the node with the lowest ID is selected as a clusterhead, and the cluster is formed by that node and all its neighbors. The same procedure is repeated among the remaining nodes until each node is assigned a cluster. In Reference [15], the same procedure is performed where instead of considering a node ID a clustered is selected considering nodal degree (the number of a node's neighbors). Basagni proposed the weight-based algorithms DMAC evaluated in this paper [4]. The generalization we define here is also suggested in a following paper of his [9], where the need for better 'control' of mobility in a hierarchical environment is shown. These algorithms assume only local (1-hop) information and are quite fast and message efficient compared to those requiring global topology information [16]. In ARC [17], a cluster change only occurs when one cluster becomes a subset of another, which helps in improving the stability of the clusters. Denko [18] uses mobile agents for this purpose. More specifically, mobile agents are used for cluster maintenance and distribution of routing information at each node. Mobile agents also help in cluster size adjustment, re-clustering, and continuous cluster state monitoring. In other solutions [19–22], the mobility pattern of the nodes is considered for forming the clusters. In Reference [21], the mobile nodes are organized in variable-sized non-overlapping clusters, while in References [19,20] the clusters are of variable diameter. The algorithm proposed in Reference [21] uses a combination of both physical and logical partitions of the network for the clustering, while similar moving nodes are grouped together in References [19,20]. Basu *et al*. [22], similarly to the lowest ID solution, propose a weight-based clustering algorithm where *aggregated local mobility* (ALM) is used to elect a clusterhead. The ratio between received power levels of successive transmissions between a pair of nodes is used to compute the relative mobility between neighboring nodes.

Different from the node-centric solutions in ad hoc networks are the cluster-centric ones, also known as *k-clustering*. A k-cluster is made up of a group of nodes that are mutually reachable by a path of length $k \geq 1$. In this approach, the network is decomposed into clusters of nodes with no specific node designated as clusterhead. In Reference [23], an algorithm for achieving *clique* partitioning (where $k = 1$) and the maintenance of the cliques in face of various network occurrences is presented. The (α, t)-cluster approach is proposed in Reference [24], where dynamically organizing mobile nodes group themselves into clusters in such a way that the probability of path availability between the nodes of the clusters can be bounded.

Clustering solutions for unit disk graphs (UDG, a typical model for describing ad hoc networks mathematically) are presented in Reference [25]. The two phase distributed approximation solution has polynomial time and message complexity. In the first phase, a spanning tree of the network is constructed. In the second phase, this spanning tree is partitioned into subtrees with bounded diameter. One main problem with this approach is that the nodes are unrealistically assumed to remain stationary during clustering.

There are also some other algorithms that have implicit constraints on the cluster diameter [24,26,27]. In Reference [26], a distributed implementation of a centralized clustering algorithm is proposed. It requires constructing a global spanning tree to generate clusters that satisfy certain constraints on the number of nodes in each cluster and the number of hierarchical levels. The priority is to create clusters of size between k and $2k - 1$, for a given k. The algorithm first creates a rooted spanning tree which covers the entire network. The cluster formation is then run bottom-up, where subtrees are made into clusters that meet the size requirements. Wang *et al*. [28] discuss the problem of cluster maintenance at length. The protocol is based on the properties of diameter-2 graphs and makes use of a spanning tree maintained at some nodes. For further references on ad hoc clustering, the reader is referred to Reference [3].

3. DMAC and GDMAC

In this section, we describe the operations of the DMAC protocol. We further generalize DMAC to GDMAC by giving the idea of how GDMAC helps in improving the performance of DMAC by limiting the impact of mobility.

Upon starting the protocol, a DMAC node computes its *weight*, that is, a real number > 0 which indicates how good that node is for serving as a clusterhead. For instance, the weight could be computed based on

DOI: 10.1002/wcm

the node's residual energy or on its current velocity. This also implies that a node's weight changes in time, reflecting the changes in the node's status. The node then acquires knowledge of its neighbors' identity and weights and depending on the weights it decides whether to be a clusterhead or not. This process is performed by having each node periodically send out 'hello' packets that carry the node's identity, its current weight, and status (either a *clusterhead* or an *ordinary node*; clearly at the start of the algorithm the status of a node is undecided). In particular, the node with the bigger weight in its (one hop) neighborhood will declare itself a clusterhead. Consequently, all its neighbors become ordinary nodes. This prevents two clusterhead to become neighbors, thus obtaining a well-spread set of clusterheads throughout the network.

According to the DMAC specifications [4], an ordinary node always affiliates with the neighboring clusterhead with the biggest weight. If in time a 'bigger weighted' clusterhead comes along, the node switches to the new one. This operation is called a re-affiliation. Another instance of re-affiliation happens when two clusterheads come into the hearing range of each other, only the 'heavier' one (in terms of weight) stays clusterhead, while the other one *resigns* its current role and becomes an ordinary node of the biggest neighboring clusterhead. If the clusterhead of an ordinary node moves away, the orphan (re-)affiliate with the biggest clusterhead around. If none is in range, the node becomes a clusterhead itself (this role change is called an *election*).[‡] This is the way DMAC accommodates network dynamics due to node mobility and to the arrival of new nodes in the network. This feature is made possible by the continuous 'monitoring' of a node's surroundings to determine the presence of new nodes; as soon as a new node is detected, relevant informations (identity, weight, etc.) are exchanged among the nodes, and suitable procedures are triggered to re-organize the clusters to include the newly arrived.

It is clear that, depending on the degree of mobility in the network and on the particular mobility pattern followed by the nodes, there can be frequent cluster re-organizations in the form of elections and re-affiliations. There are cases when even a single topological change might trigger numerous role changes among nodes. This 'ripple effect' of DMAC-like protocols has been studied in Reference [29] which describes how the introduction of a single new node can cause a change of role reversals among the other nodes along a path of a tree when certain conditions exist. Such undesirable phenomena are detrimental for network performance, as they produce extra protocol overhead, decrease the data throughput and increase data latency (when a cluster is re-organizing, data cannot be efficiently sent).

In order to decrease the number of mobility-dependent changes (election and re-affiliations) we can (a) relax the quite strict DMAC requirement that forbids neighboring clusterheads, and (b) allow an ordinary node to stay with its current clusterhead even if a bigger one has come by. The *generalized DMAC* protocol, or GDMAC in short, was defined with these ideas in mind. GDMAC operations are based on the value of two parameters H and K that are introduced with the aim of mitigating the DMAC possible 'chain reaction.' More specifically:

H. Parameter H implements the idea that a cluster re-organization is needed only when the new clusterhead is really better than the current one. In other words, an ordinary node switches to a newly arrived clusterhead only when the weight of the new clusterhead exceeds the weight of its current clusterhead by a quantity H. The higher is the value of H, the less likely a node will switch to a new neighboring clusterhead.

K. Parameter K controls the spatial density of the clusterheads. Up to $K \geq 0$, clusterheads are now allowed to be neighbors. Having $K = 0$ ensures that no two clusterheads can be neighbors (DMAC). Setting $K > 0$ helps in mitigating cluster re-organization since now a clusterhead is not forced to give up its leadership when up to $K - 1$ clusterhead with bigger weights become its neighbors.

It is clear that GDMAC is indeed a generalization of DMAC. The DMAC protocol is obtained when both K and H are set to 0, which can be meaningful in static or quasi-static networks.

In the remainder of this section, we describe the GDMAC procedures executed by a generic node v when it starts the protocol operations.

Except for the initial procedure, the algorithm is message driven; a specific procedure will be executed at a node depending on the reception of the corresponding message. We use three types of messages that are exchanged among the nodes. More specifically, the message $\textsc{Ch}(v)$ is used by a node v to make its neighbors aware that it is going to be a clusterhead; the message

[‡] There are also re-affiliations and elections due to the dynamic changes in nodal weights. Those are not taken into account in the experiments in this paper.

DOI: 10.1002/wcm

JOIN(v, u) is sent by node v to communicate to its neighbors that it will be part of the cluster whose clusterhead is node u. The message RESIGN(w) is used to require the resignation from the role of clusterhead of any receiving clusterheads whose weight is $\leq w$.

In the following description, we use the following notation and common assumptions.

- v, the generic node executing the algorithm (from now on we will assume that v encodes not only the node's ID but also its weight w_v). The set of all nodes is indicated by V and the set of v's one hop neighbors is denoted by $\Gamma(v)$.
- *Cluster*(v), the set of nodes in v's cluster. It is initialized to $\emptyset$, and it is updated only if v is a clusterhead.
- *Clusterhead*, the variable in which every node records the (ID of the) clusterhead that it joins.
- $Ch(-)$, boolean variables. Node v sets $Ch(u)$, $u \in \{v\} \cup \Gamma(v)$, to **true** when either it sends a CH(v) message ($v = u$) or it receives a CH(u) message from u ($u \neq v, u \in \Gamma(v)$).
- Every node is made aware of the failure of a link, or of the presence of a new link by a service of a lower level (this will trigger the execution of the corresponding procedure).
- The GDMAC procedures (G-procedures, for short) are 'atomic,' that is, they are not interruptible.
- At clustering set up or when a node is added to the network, its variables *Clusterhead*, $Ch(-)$, and *Cluster*$(-)$ are initialized to **nil**, **false**, and $\emptyset$, respectively.

3.1. Init

At clustering set up, or when a node v is added to the network, it executes the procedure *Init* in order to determine its own role. If among its neighbors there is at least a clusterhead with bigger weight, then v will join it. Otherwise it will be a clusterhead. In this case, the new clusterhead v has to check the number of its neighbors that are already clusterheads. If they exceed K, then a RESIGN message is also transmitted, carrying the weight of the first clusterhead (namely, the one with the $(K+1)$th biggest weight) that violates the K-neighborhood condition (this weight is determined by the operator $\min_K$). On receiving a message RESIGN(w), every clusterhead u such that $w_u \leq w$ will resign. Notice that a neighbor with a bigger weight that has not decided its role yet (this may happen at the clustering set up, or when two or more nodes are added to the network at the same time), will eventually send a message (every node executes the *Init* procedure). If this message is a CH message, then v could possibly resign (after receiving the corresponding RESIGN message) or affiliate with the new clusterhead.

```
PROCEDURE Init;
begin
  if {z ∈ Γ(v) : w_z > w_v ∧ Ch(z)} ≠ ∅
    then begin
           x := max_{w_z > w_v}{z : Ch(z)};
           send JOIN(v,x);
           Clusterhead := x
         end
    else begin
           send CH(v);
           Ch(v) := true;
           Clusterhead := v;
           Cluster(v) := {v};
           if |{z ∈ Γ(v) : Ch(z)}| > K then
           send RESIGN(min_K{w_z : z ∈ Γ(v) ∧ Ch(z)})
         end
end;
```

3.2. Link_Failure

Whenever made aware of the failure of the link with a node u, node v checks if its own role is clusterhead and if u used to belong to its cluster. If this is the case, v removes u from *Cluster*(v). If v is an ordinary node, and u was its own clusterhead, then it is necessary to determine a new role for v. To this aim, v checks if there exists at least a clusterhead $z \in \Gamma(v)$ such that $w_z > w_v$. If this is the case, then v joins the clusterhead with the bigger weight, otherwise it becomes a clusterhead. As in the case of the *Init* procedure, a test on the number of the neighboring clusterheads is now needed, with the possible resigning of some of them.

```
PROCEDURE Link_failure (u);
begin
  if Ch(v) and (u ∈ Cluster (v))
    then Cluster(v) := Cluster(v) \ {u}
    else if Clusterhead = u then
           if {z ∈ Γ(v) : w_z > w_v ∧ Ch(z)} ≠ ∅
             then begin
                    x := max_{w_z > w_v}{z : Ch(z)};
                    send JOIN(v,x);
                    Clusterhead := x
                  end
             else begin
                    send CH(v);
                    Ch(v) := true;
                    Clusterhead := v;
                    Cluster(v) := {v};
                    if |{z ∈ Γ(v) : Ch(z)}| > K then
                    send RESIGN(min_K{w_z : z ∈ Γ(v) ∧ Ch(z)})
                  end
end;
```


DOI: 10.1002/wcm

3.3. New_Link

When node v is made aware of the presence of a new neighbor u, it checks if u is a clusterhead. If this is the case, and if w_u is bigger than the weight of v's current clusterhead plus the threshold H, then, independently of its own role, v affiliates with u. Otherwise, if v itself is a clusterhead, and the number of its current neighboring clusterheads is $> K$ then the operator $\min_K$ is used to determine the weight of the clusterhead x that violates the K-neighborhood condition. If $w_v > w_x$, then node x has to resign, otherwise, if no clusterhead x exists with a weight smaller than v's weight, v can no longer be a clusterhead, and it will join the neighboring clusterhead with the biggest weight.

```
PROCEDURE New_link (u);
begin
  if Ch(u) then
    if (w_u > w_Clusterhead + H)
      then begin
             send JOIN(v,u);
             Clusterhead := u;
             if Ch(v) then Ch(v) := false
           end
      else if Ch(v) and |{z ∈ Γ(v) : Ch(z)}| > K then
             begin
               w := min_K{w_z : z ∈ Γ(v) ∧ Ch(z)};
               if w_v > w then send RESIGN(w)
                          else begin
                                 x := max_{w_z>w_v}{z : Ch(z)};
                                 send JOIN(v,x);
                                 Clusterhead := x;
                                 Ch(v) := false
                               end
             end
end;
```

The following procedures are triggered by the reception of the corresponding message.

3.4. On Receiving CH(u)

When a neighbor u becomes a clusterhead, on receiving the corresponding CH message, node v checks if it has to affiliate with u, that is, it checks whether w_u is bigger than the weight of v's clusterhead plus the threshold H or not. In this case, independently of its current role, v joins u's cluster. Otherwise, if v is a clusterhead with more than K neighbors which are clusterheads, as in the case of a new link, the weight of the clusterhead x that violates the K-neighborhood condition is determined, and correspondingly the clusterhead with the smallest weight will resign.

```
On receiving CH(u);
begin
  if (w_u > w_Clusterhead + H) then begin
                 send JOIN(v,u);
                 Clusterhead := u;
                 if Ch(v) then Ch(v) := false
               end
          else if Ch(v) and |{z ∈ Γ(v) : Ch(z)}| > K then
                 begin
                  w := min_K{w_z : z ∈ Γ(v) ∧ Ch(z)};
                  if w_v > w then send RESIGN(w)
                       else begin
                              x := max_{w_z>w_v}{z : Ch(z)};
                              send JOIN(v,x);
                              Clusterhead := x;
                              Ch(v) := false
                            end
                 end
  end;
```

3.5. On Receiving JOIN(u,z)

On receiving the message JOIN(u,z), the behavior of node v depends on whether it is a clusterhead or not. In the affirmative, v has to check if either u is joining its cluster ($z = v$. In this case, u is added to *Cluster*(v)) or if u belonged to its cluster and is now joining another cluster ($z \neq v$. In this case, u is removed from *Cluster*(v)). If v is not a clusterhead, it has to check if u was its clusterhead. Only if this is the case, v has to decide its role: it will join the biggest clusterhead x in its neighborhood such that $w_x > w_v$ if such a node exists. Otherwise, it will be a clusterhead. In this latter case, if the K-neighborhood condition is violated, a RESIGN message is transmitted in order for the clusterhead with the smallest weight in v's neighborhood to resign.

```
On receiving JOIN(u, z);
begin
  if Ch(v)
    then if z = v then Cluster(v) := Cluster(v) ∪ {u}
      else if u ∈ Cluster(v) then Cluster(v) := Cluster(v) \ {u}
    else if Clusterhead = u then
     if {z ∈ Γ(v) : w_z > w_v ∧ Ch(z)} ≠ ∅
      then begin
             x := max_{w_z>w_v}{z : Ch(z)};
             send JOIN(v,x);
             Clusterhead := x
           end
      else begin
             send CH(v);
             Ch(v) := true;
             Clusterhead := v;
             Cluster(v) := {v};
             if |{z ∈ Γ(v) : Ch(z)}| > K then
             send RESIGN(min_k{w_z : z ∈ Γ(v) ∧ Ch(z)})
           end
end;
```


DOI: 10.1002/wcm

3.6. On Receiving RESIGN(w)

On receiving the message RESIGN(w), node v checks if its weight is $\leq w$. In this case, it has to resign and it will join the neighboring clusterhead with the biggest weight. Notice that since the G-procedures are supposed to be not interruptible, and since v could have resigned already, it has also to check if it is still a clusterhead.

```
On receiving RESIGN(w);
begin
  if Ch(v) and w_v ≤ w then begin
                              x := max_{w_z > w_v}{z : Ch(z)};
                              send JOIN(v,x);
                              Clusterhead := x;
                              Ch(v) := false
                            end
end;
```

4. Performance Evaluation

4.1. Metrics of Interest and Simulation Scenarios

In order to assess the impact of mobility on the performance of GDMAC and compare it to that of DMAC, we have performed extensive simulation experiments. Both DMAC and its generalized version have been implemented in the VINT Project network simulator ns2 [30] with the CMU wireless extension [6,31] that implements the IEEE 802.11 MAC with the DCF. Our study aims at measuring the 'goodness' of the protocols in dealing with node mobility. In particular, we are interested in evaluating DMAC and GDMAC with respect to the following key performance metrics (all averages).

- Cluster density, that is, the number of clusters formed during the simulation time.
- Cluster stability, which we define as how much a cluster remain the same ('stable') while the nodes move. The stability of a cluster is captured by the following metrics.
  - Cluster lifetime, that is, how long a node has the role of clusterhead;
  - residence time, which is the affiliation time of an ordinary node to a particular cluster, and
  - status changes, that is, the number of status changes per node (election and re-affiliations).
- Message complexity per node. This is the total number of cluster formation and maintenance control messages sent by a node.

Nodes move according to three different mobility models, namely, the *random way point* (RWP) model, the *random walk* (RW) model, and the *Manhattan* (MAN) model.

1. Since its introduction in Reference [6] the RWP model is used in many prominent simulation studies of ad hoc network protocols. According to this model, each node begins by pausing in one location for a certain period of time. After this time, a node chooses a random destination and a speed that is uniformly distributed between [0, maxspeed]. The node then travels toward the newly chosen destination in a straight line with the selected speed. Upon arrival, the node pauses for a specified period of time before starting traveling again.
2. The RW mobility model mimics erratic movement. In this model, a node moves from its current location to a new location by randomly choosing direction and speed. The new speed and direction are both chosen from predefined ranges [0, maxspeed] and [0, 360].
3. The MAN model, first introduced in Reference [8] (also studied in References [7,32] as 'city section mobility model'), describes a realistic city mobility model by stating how a node moves in a city surrounding. This model requires the simulation area to be divided in horizontal and vertical blocks (here 10) and nodes are forced to move along predefined paths between the blocks with predetermined speed (maxspeed). At the end of a block, a node pause (here for 5s) and then it decides whether to turn or continue to travel in the same direction with some probability (here set to 0.5).

We make the common assumption that two nodes are neighbors if and only if their Euclidean distance is $\leq$250 m. The simulations have realistic MAC and physical layer characteristics where packets might get lost due to the underlying channel conditions and due to possible collisions and need to be re-transmitted. The nodes beacon their presence periodically (in our experiments, this happens every half a second) and the drifting in of a new node is realized when its new neighbors hear its beacons. Similarly, when a node does not hear beacons from a known neighbor within a certain amount of time (here 3 s), it assumes the

DOI: 10.1002/wcm

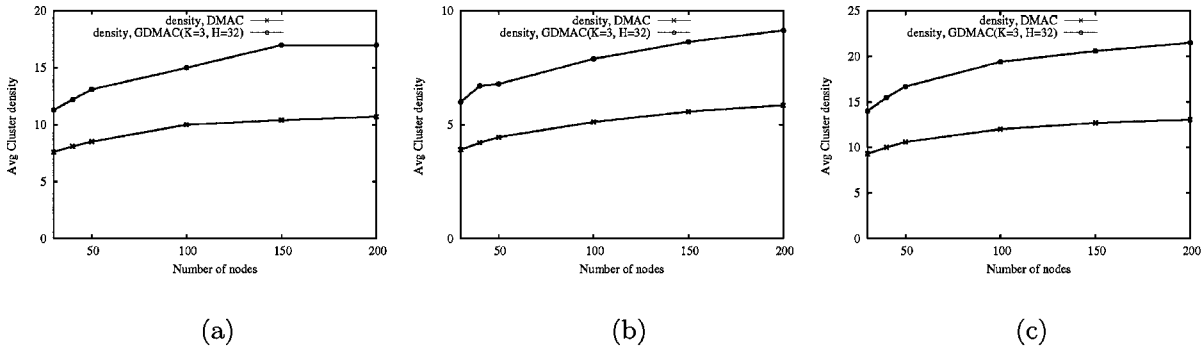

Fig. 1. Average cluster density. (a) RWP, (b) RW, (c) MAN.

neighbor to be either 'dead' or out of range due to mobility. In our simulations, the number of nodes n has been assigned the values 30, 40, 50, 100, 150, and 200, while L has been set to 1000 m. This allows us to test the protocols on sparse topologies (30 and 40 nodes) and on increasingly dense ones. All results are obtained on 100 topologies and have been run for considerably period of time (1000 s) and the results are averaged over 100 different topologies to ensure the suppression of startup transients in the values obtained. All nodes have been assigned random weights between 0 and 80. The GDMAC values for K and H are set to 1, 3, and 4, and to 16, 32, 64, and 80, respectively. We have observed that for values of $H \geq 32$ and fixed constant K, there is no significant improvement on the performance of GDMAC. For this reason, we also have considered smaller values of H. In this paper, we show results obtained for the two values $K = 3$ and $H = 32$. Simulations are performed where the speed of the nodes (maxspeed, as termed in the description above) is 2, 5, and 10 m/s.

4.2. Cluster Density

Our first metric concerns the average number of clusters formed during the simulation time. Cluster forming and updating due to election and re-affiliation is the major source of clustering overhead, and thus this metric gives us an idea on how much the two protocols creates clusters while the nodes move. The average cluster density for DMAC and GDMAC ($H = 32$, $K = 3$) when the nodes move at 2 m/s are shown in Figure 1(a)–(c). With increasing n, when the nodes move according to the RWP model the average number of CHs 'converges' to 17 in case of GDMAC. For the RW model, the average number of cluster is a little less than 10, and for the MAN model the average value is 22. For DMAC ($K = H = 0$), the average number of clusters levels around 10 (RWP), 5 (RW), and 14 (MAN) for increasing values of n. This is clearly because GDMAC allows clusterheads to be neighbors.

Figure 2(a)–(c) shows the effect of mobility on the GDMAC average cluster density. For greater values of n, mobility has little effect. With increasing n, the density increases until the average density of 17 CHs (RWP), 9 CHs (RW), and 22 CHs (MAN) is achieved.

4.3. Cluster Stability

Figure 3(a)–(c) shows cluster lifetime (in seconds) over the total simulation time. For the RWP and MAN models, lifetime for both DMAC and GDMAC

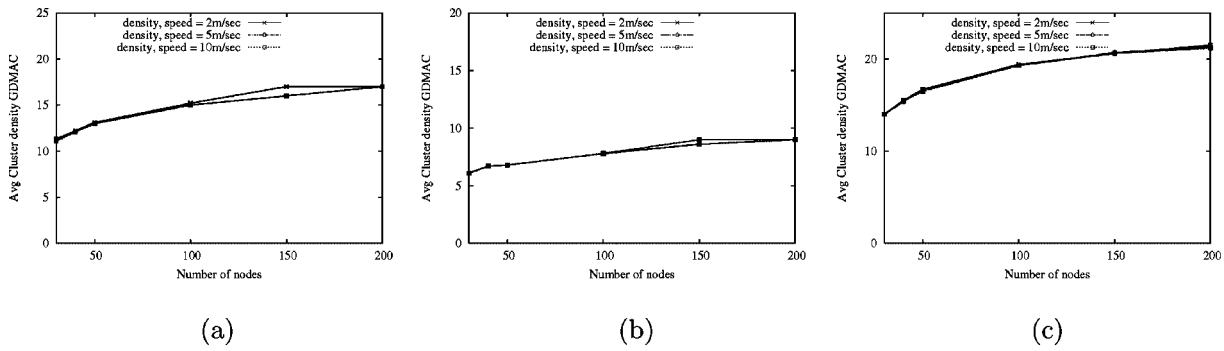

Fig. 2. Impact of mobility in GDMAC. (a) RWP, (b) RW, (c) MAN.

DOI: 10.1002/wcm

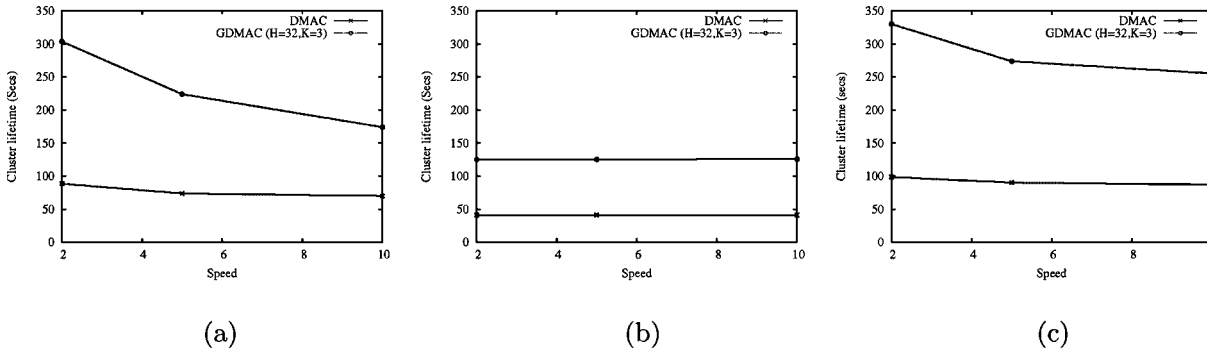

Fig. 3. Cluster lifetime for varying speeds. (a) RWP, (b) RW, (c) MAN.

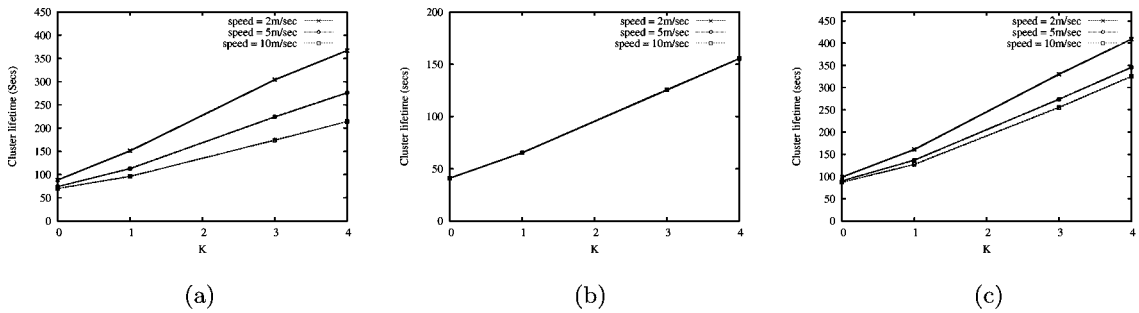

Fig. 4. Cluster lifetime for varying K. (a) RWP, (b) Random walk, (c) MAN.

decreases with increasing speed as nodes tend to drift away faster with higher velocity. Greater lifetime is achieved for GDMAC with $K = 3$ as less CHs are forced to resign when they become neighbors thus increasing the cluster lifetime over DMAC's. However, for the RW model speed does not have too much of an impact on the cluster lifetime since according to this model the nodes tend to stay in the center of the simulation area most of the time, which results in a static/quasi-static scenario. As expected, GDMAC tends to be more stable than its particular instance DMAC.

In Figure 4(a)–(c), it is seen that for $n = 150$, and $H = 32$, cluster lifetime increases with increasing K since for larger K less CHs are forced to resign thereby making the clusters more stable, in all the three mobility models. Again stability is greater for slower speeds for RWP and MAN while RW is independent of nodal speed.

Figure 5(a)–(c) shows that cluster lifetime is improved by increasing H for all the mobility models. As expected, the improvement is more noticeable for smaller values of H. As in previous cases, cluster lifetime is affected by the nodal speed only when the nodes move according to RWP and MAN.

Figure 6 shows the residence time (in seconds) of the ordinary nodes for DMAC and GDMAC. For the RWP and MAN models, we see that increasing

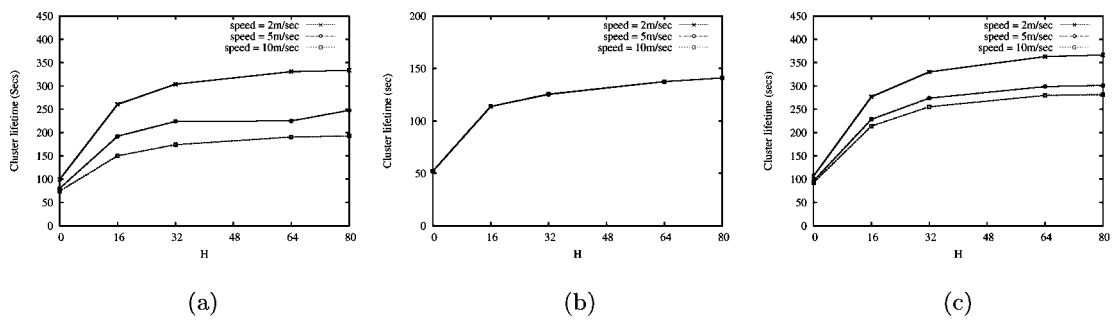

Fig. 5. Cluster lifetime for varying H. (a) RWP, (b) RW, (c) MAN.

DOI: 10.1002/wcm

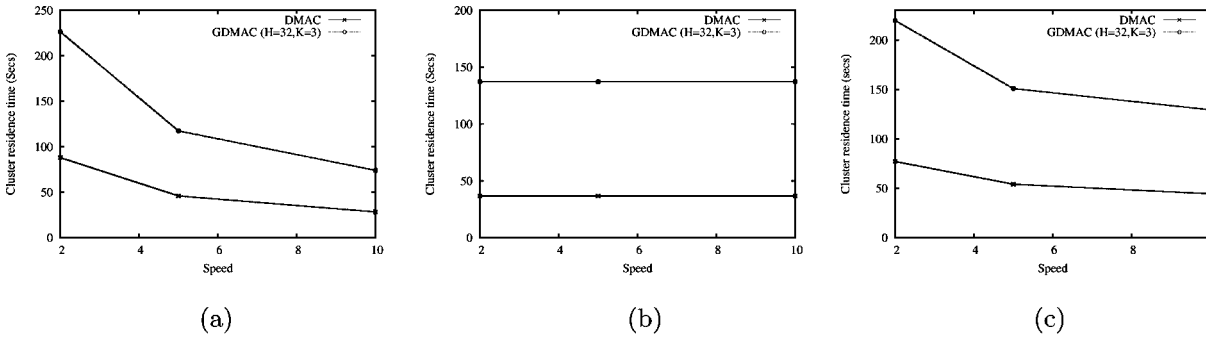

Fig. 6. Residence time for varying speeds. (a) RWP, (b) RW, (c) MAN.

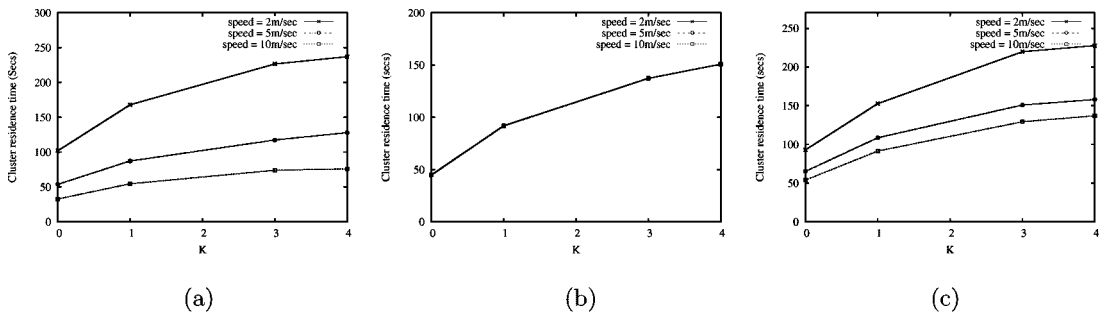

Fig. 7. Cluster residence time for varying K. (a) RWP, (b) RW, (c) MAN.

velocity forces both ordinary nodes and CHs to move away more rapidly, thereby decreasing cluster stability. This pattern is particularly noticeable in DMAC. In GDMAC, the effect of increasing nodal speed is mitigated by the introduction of H and K. Although increased speed has little impact on scenarios where the nodes move according to the RW mobility model GDMAC is more stable than DMAC.

Figure 7(a)–(c) shows results for clustering residence time for different values of K. We observed that for increasing K, CHs resign less frequently and hence the clusters formed are more stable and existing members are not forced to look out for new CHs in case of cluster failure. As a consequence, we obtain increased average cluster residence time. Although the trend is the same for all the three different node nobilities, speed is no issue for nodes that move according to the RW mobility model.

It is seen that by increasing H the residence time does increase gradually. However, we noticed that the change is not significant for $H \geq 20$. Figure 8(a)–(c) depicts variations in residence time when H varies between 0 and 20. While in the RW case, differences due to different speeds are not observed, in the other two cases we notice better performance at lower speeds. Residence time increases with increasing H because

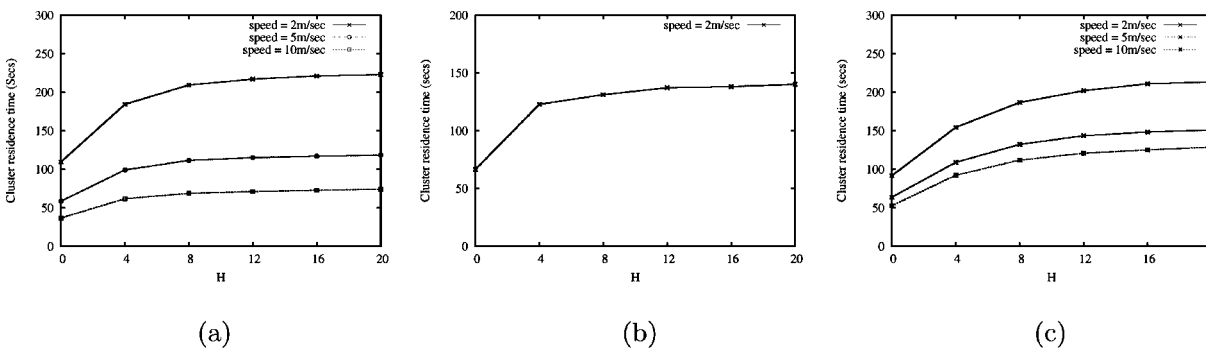

Fig. 8. Cluster residence time for varying H. (a) RWP, (b) RW, (c) MAN.

DOI: 10.1002/wcm

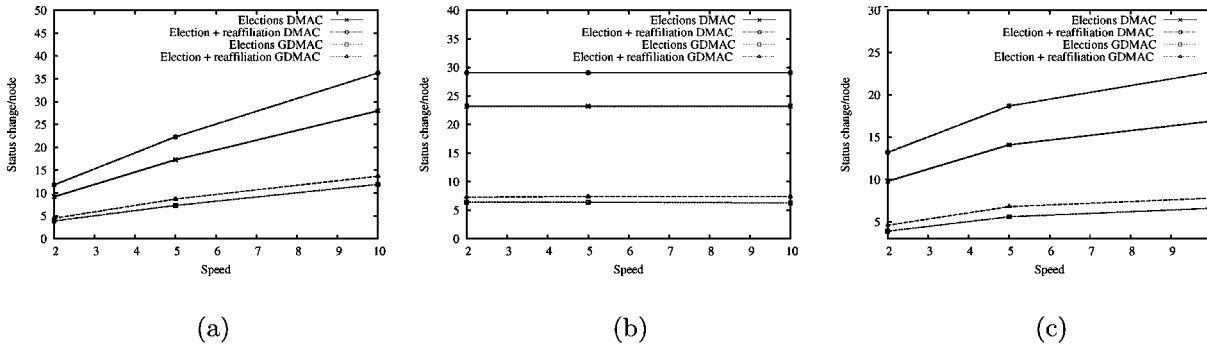

Fig. 9. Elections and re-affiliations. (a) RWP, (b) RW, (c) MAN.

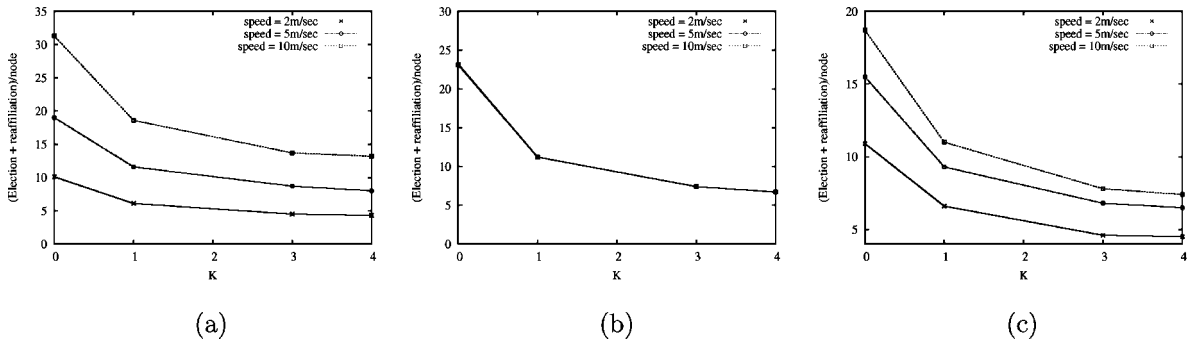

Fig. 10. Elections and re-affiliations for varying K. (a) RWP, (b) RW, (c) MAN.

the ordinary nodes tend to stick to their current CHs even when a new CH drifts in the neighborhood. Smaller values of H suggest that the new CH is fitter than the existing CH forcing ordinary nodes to change cluster affiliation.

Figure 9(a)–(c) shows the effect of speed on elections and re-affiliations between DMAC and GDMAC ($K = 3$ and $H = 32$). Since an increasing K forces less clusterheads to resign and increasing values of H induce less ordinary nodes switching among clusterheads, GDMAC has a smaller number of elections and re-affiliations than DMAC. For increasing speed, the clusters become less stable and hence the number of role changes increases with increasing speed in the case node move according to the RWP and MAN models.

For fixed H and increasing K, an existing CH is less likely to resign thereby decreasing the number of resignation. An ordinary node is more likely to find another CH in its periphery, thereby decreasing the number of election events. This is shown in Figure 10(a)–(c). As noticed multiple times, the effects of speed are observed only when nodes move according to the RWP and MAN models.

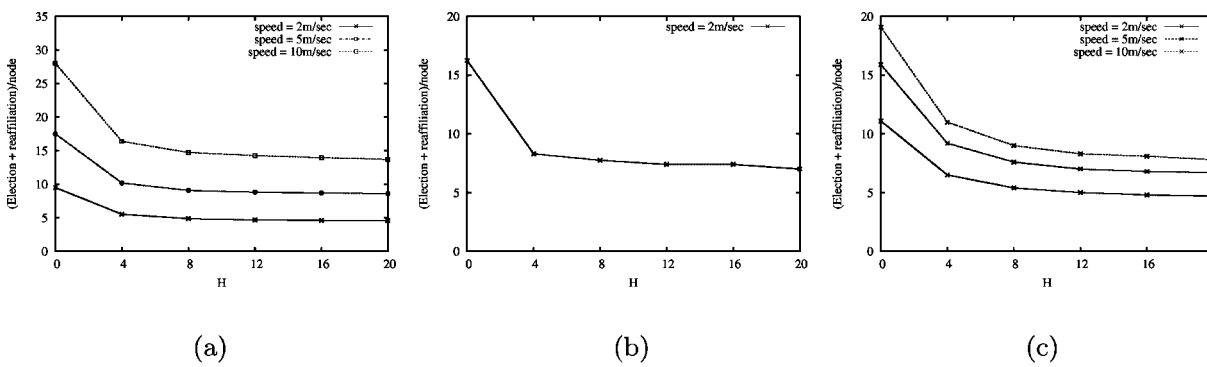

Fig. 11. Elections and re-affiliations for varying H. (a) RWP, (b) RW, (c) MAN.

DOI: 10.1002/wcm

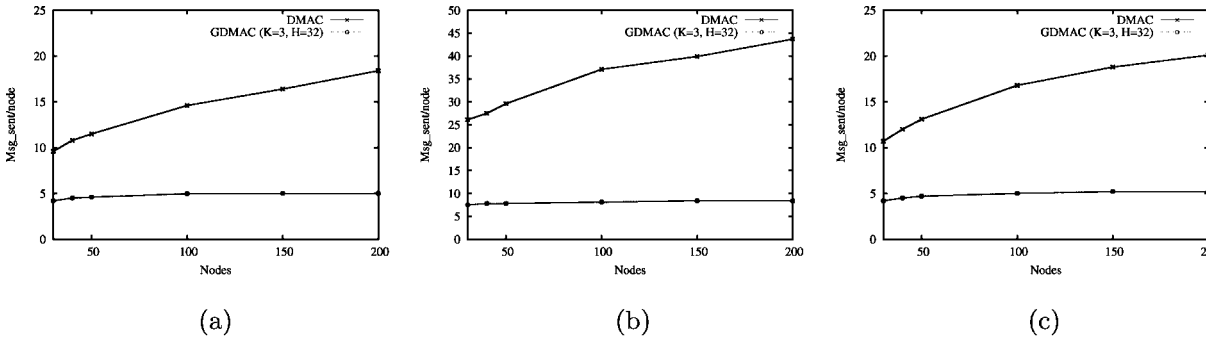

Fig. 12. Number of messages sent. (a) RWP, (b) RW, (c) MAN.

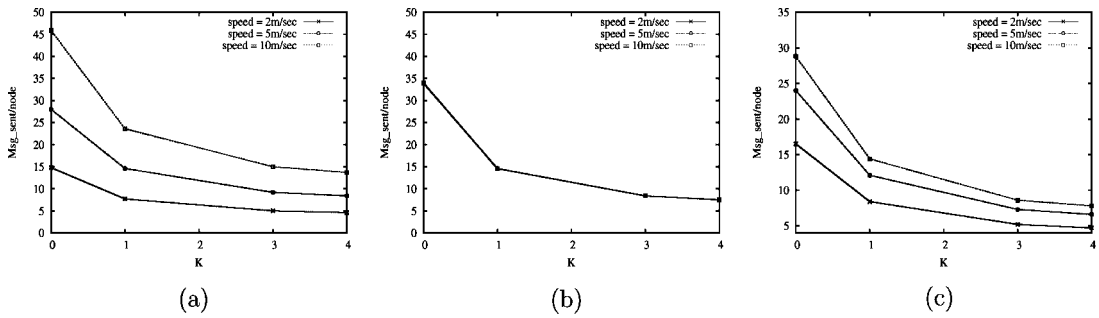

Fig. 13. Messages sent for varying K. (a) RWP, (b) RW, (c) MAN.

For fixed K and increasing H, it is less likely that an ordinary node will become a CH. This justifies the observed decrease in the number of elections. For higher values of H, it is less likely that an ordinary node moves from its existing cluster to a new one (see Figure 11(a)–(c)).

4.4. Message Complexity

Figure 12(a)–(c) shows the average number of messages sent by a node. This gives us a measure of the overhead created for maintaining the clusters, which is also an indication on how much clustering set up and maintenance costs in terms of energy per node. The number of messages sent increases with larger values of n until it levels off. This is due to the higher network density; when n grows, there is increased connectivity among the nodes. This helps in keeping a bound on the number of the messages sent. Again, being GDMAC more stable, the number of messages sent is less than those send by DMAC.

The number of control messages used by GDMAC is also affected by changing values of K. Figure 13(a)–(c) shows that independently of the mobility model, given that by increasing K we decrease the number of CH resignations, the number of control messages

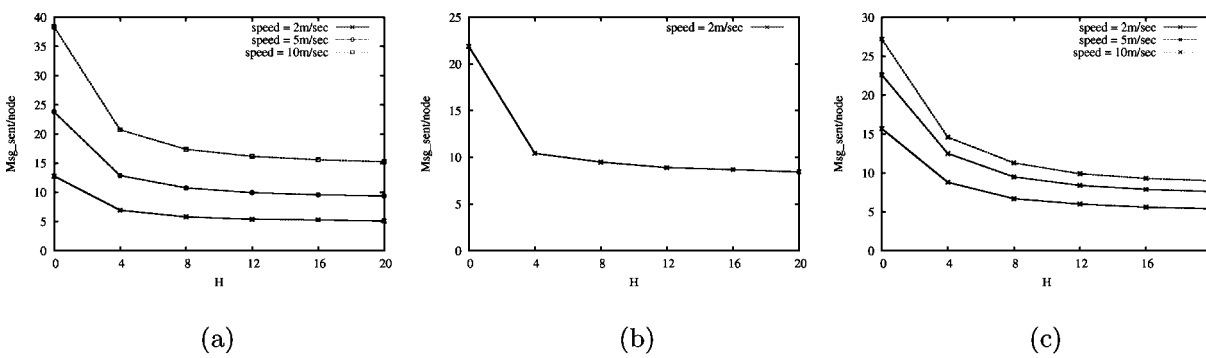

Fig. 14. Messages sent for varying H. (a) RWP, (b) RW, (c) MAN.

DOI: 10.1002/wcm

decreases accordingly. (Here, H is set to $= 32$.) The case of RWP and MAN-dictated mobility are also sensitive to different speeds.

Figure 14(a)–(c) shows what happens to GDMAC when varying the value of H. We observe that increasing H imposes decreasing JOIN messages from ordinary nodes, which tend to stick to their current CHs.

As observed before, speed is also a factor in scenarios where the nodes move according to the RWP and MAN mobility models.

5. Conclusions and Future Research

In this paper, we investigated the impact of different kinds of mobility on a typical clustering organization for a mobile ad hoc network. The DMAC protocol has been demonstrated *via* simulations with respect to metrics relevant for assessing clustering performance, especially in the face of nodes mobility. We have also proposed and studied a variation to the basic DMAC. The resulting protocol, termed GDMAC, has been proven outperforming DMAC with respect to all the considered metrics. In particular, we observed that GDMAC results in good performance independently of the mobility model considered.

It is our aim to keep investigating clustering and mobility. We plan to explore two main directions. First of all, we intend to inspect deeply some other metrics, such as energy cost (in case the nodes are energy constrained, as in wireless sensor networks), and those metrics related to constructing and maintaining a backbone of the clusterheads (and hence several routing-related metrics). Finally, we will be considering some other clustering solutions for mobile ad hoc networks, and we will be comparing it to the protocol GDMAC studied in this paper. Although many solutions have been proposed for clustering and backbone formation, maintenance in the presence of mobility has not been tackled with extensively so far, which we plan to do in the future.

References

1. Basagni S, Conti M, Giordano S, Stojmenovic I (eds). *Mobile Ad Hoc Networking*. IEEE Press and John Wiley & Sons, Inc.: Piscataway, NJ and New York, NY, April 2004.
2. Sheltami T, Mouftah HT. Cluster-based and power aware routing in wireless mobile ad hoc networks. In *Handbook of Algorithms for Wireless Networking and Mobile Computing*, Computer & Information Science, Chapter 10, Boukerche A (ed.). Chapman & Hall/CRC: Boca Raton, FL, 2005; 217–238.
3. Basagni S, Mastrogiovanni M, Panconesi A, Petrioli C. Localized protocols for ad hoc clustering and backbone formation: a performance comparison. *IEEE Transactions on Parallel and Distributed Systems, Special Issue on Localized Communication and Topology Protocols for Ad Hoc Networks*, Olariu S, Simplot-Ryl D, Stojmenovic I (eds). 2006; **17**(4): 292–306.
4. Basagni S. Distributed clustering for ad hoc networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, Zomaya AY, Hsu DF, Ibarra O, Origuchi S, Nassimi D, Palis M (eds). Perth/Fremantle, Australia, June 23–25 1999. IEEE Computer Society, 310–315.
5. Boukerche A, Bononi L. Simulation and modeling of wireless, mobile and ad hoc networks. In *Mobile Ad Hoc Networking*, Chapter 14, Basagni S, Conti M, Giordano S, Stojmenovic I (eds). IEEE Press and John Wiley & Sons, Inc.: Piscataway, NJ and New York, NY, April 2004; 373–410.
6. Broch J, Maltz DA, Johnson DB, Hu Y-C, Jetcheva J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom 1998*, Dallas, TX, October 25–30 1998; 85–97.
7. Camp T, Boleng J, Davies V. A survey of mobility models for ad hoc network research. *Wiley Interscience Wireless Communications & Mobile Computing, Special Issue on Mobile Ad Hoc Networking: Research Trend and Application*, Basagni S, Lee SJ (eds). 2002; **2**(5): 483–502.
8. European Telecommunications Standards Institute, ETSI. Selection procedures for the choice of radio transmission technologies of the UMTS. Technical Report 101 112 V3.2.0, ETSI SMG-5, Sophia Antipolis, France, April 1998.
9. Basagni S. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In *Proceedings of the IEEE 50th International Vehicular Technology Conference, VTC 1999-Fall*, Vol. 2, Amsterdam, The Netherlands, September 19–22 1999; 889–893.
10. Amis AD, Prakash R. Load-balancing clusters in wireless ad hoc networks. In *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, ASSET 2000*, Richardson, TX, March 24–25 2000; 25–32.
11. Baker DJ, Ephremides A. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communications* 1981; **COM-29**(11): 1694–1701.
12. Parekh AK. Selecting routers in ad hoc wireless networks. In *Proceedings of the SBT/IEEE International Telecommunications Symposium, ITS 1994*, Rio de Janeiro, Brasil, August 22–25 1994; 420–424.
13. Amis AD, Prakash R, Vuong THP, Huynh DT. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of IEEE Infocom 2000*, Vol. 1, Tel Aviv, Israel, March 26–30 2000; 32–41.
14. Lin CR, Gerla M. Adaptive clustering for mobile wireless networks. *Journal on Selected Areas in Communications* 1997; **15**(7): 1265–1275.
15. Gerla M, Tsai JT-C. Multicluster, mobile, multimedia radio network. *Wireless Networks* 1995; **1**(3): 255–265.
16. Chatterjee M, Das SK, Turgut D. WCA: a weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing, Special Issue on Mobile Ad Hoc Networks* 2002; **5**(2): 193–204.
17. Belding-Royer EM. Multi-level hierarchies for scalable ad hoc routing. *ACM/Kluwer Wireless Networks* 2003; **9**(5): 461–478.
18. Denko MK. The use of mobile agents for clustering in mobile ad hoc networks. In *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer*

DOI: 10.1002/wcm

Scientists and Information Technologists on Enablement through Technology, SAICSIT 03, Gauteng, South Africa, September 17–19 2003; 241–247.
19. Er II, Seah WKG. Mobility-based d-hop clustering algorithm for mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference, WCNC 2004*, Vol. 4, Atlanta, GA, March 21–25 2004; 2359–2364.
20. Seah WKG, Er II. Clustering overhead and convergence time analysis of the mobility-based multi-hop clustering algorithm for mobile ad hoc networks. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems, ICPADS 2005*, Vol. 2, Fukuoka, Japan, July 20–22 2005; 130–134.
21. An B, Papavassilious S. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad hoc wireless networks. *International Journal of Network Management* 2001; **11**(6): 387–395.
22. Basu P, Khan N, Little TDC. A mobility based metric for clustering in mobile ad hoc networks. In *Proceedings of the International Workshop on Wireless Networks and Mobile Computing, WNMC 2001*, Mesa, AZ, April 16 2001; 413–418.
23. Krishna P, Vaidya NH, Chatterjee M, Pradhan DK. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review* 1997; **27**(2): 49–64.
24. McDonald AB, Znati T. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad Hoc Networks* 1999; **17**(8): 1466–1487.
25. Fernandess Y, Malkhi D. *k*-clustering in wireless ad hoc networks. In *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing, POMC 2002*, Toulouse, France, October 30–31 2002; 31–37.
26. Banerjee S, Khuller S. A clustering scheme for hierarchical control in multi-hop wireless networks. In *Proceedings of the 20th IEEE Infocom 2001*, Vol. 2, Anchorage, AK, April 22–26 2001; 1028–1037.
27. Ramanathan R, Steenstrup M. Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. *Mobile Networks & Applications* 1998; **3**(1): 101–119.
28. Wang L, Olariu S. A unifying look at clustering in mobile ad hoc networks. *Wireless Communications and Mobile Computing* 2004; **4**(6): 623–637.
29. Bettstetter C, Friedrich B. Time and message complexity of the generalized distributed mobility adaptive clustering (GDMAC) algorithm in wireless multihop networks. In *Proceeding of the 57th IEEE Semiannual Vehicular Technology Conference, VTC 2003-Spring*, Vol. 1, Jeju, Korea, April 22–25 2003; 176–180.
30. The VINT Project. *The ns Manual*. http://www.isi.edu/nsnam/ns/ [2006].
31. The Rice University Monarch Project. Monarch wireless extension to the ns2 simulator. www.monarch.cs.rice.edu [May 7 2006].
32. Davies V. Evaluating mobility models within an ad hoc network. Master's Thesis, Colorado School of Mines, Golden, CO, 2000.

Authors' Biographies

Rituparna Ghosh holds a Ph.D. in Computer Engineering from the Northeastern University, Boston, MA (May 2006). She received her M.Sc. degree in Computer Science from the University of Texas at Dallas in 2002. Dr Ghosh's current research interests concern research and implementation aspects of mobile networks, with a particular emphasis on wireless ad hoc and sensor networking.

Stefano Basagni holds a Ph.D. in Electrical Engineering from the University of Texas at Dallas (December 2001) and a Ph.D. in Computer Science from the University of Milano, Italy (May 1998). He received his B.Sc. degree in Computer Science from the University of Pisa, Italy, in 1991. Since Winter 2002, he is on the faculty at the Department of Electrical and Computer Engineering at Northeastern University, in Boston, MA. From August 2000 to January 2002, he was a professor of computer science at the Department of Computer Science of the Erik Jonsson School of Engineering and Computer Science, the University of Texas at Dallas. Dr. Basagni's current research interests concern research and implementation aspects of mobile networks and wireless communications systems, Bluetooth and sensor networking, definition and performance evaluation of network protocols, and theoretical and practical aspects of distributed algorithms. Dr. Basagni has published over four dozens of referred technical papers and book chapters. He is also co-editor of two books. Dr. Basagni served as a guest editor of the special issue of the *Journal on Special Topics in Mobile Networking and Applications* (*MONET*) *on Multipoint Communication in Wireless Mobile Networks*, of the special issue on mobile ad hoc networks of the Wiley's Interscience's *Wireless Communications & Mobile Networks* journal, and of the Elsevier's journal Algorithmica on algorithmic aspects of mobile computing and communications. Dr. Basagni serves as a member of the editorial board and of the technical program committee of *ACM* and *IEEE* journals and international conferences. He is a senior member of the *ACM* (including the ACM SIGMOBILE), senior member of the *IEEE* (*Computer and Communication societies*), and member of *ASEE* (*American Society for Engineering Education*).

DOI: 10.1002/wcm