Journal of King Saud University – Computer and Information Sciences (2016) **28**, 276–288

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A hierarchical virtual backbone construction protocol for mobile ad hoc networks

Bharti Sharma [a], **Ravinder Singh Bhatia** [b], **Awadhesh Kumar Singh** [b,*]

[a] *DIMT Kurukshetra, India*
[b] *NIT Kurukshetra, India*

Received 11 January 2014; revised 10 May 2014; accepted 4 June 2014
Available online 31 October 2015

KEYWORDS

MANET;
Leader election;
Diameter;
Clustering;
Backbone

Abstract We propose a hierarchical backbone construction protocol for mobile ad hoc networks. Our protocol is based on the idea of using an efficient extrema finding method to create clusters comprising the nodes that are within certain prespecified wireless hop distance. Afterward, we apply our 'diameter' algorithm among clusters to identify the dominating nodes that are, finally, connected via multi-hop virtual links to construct the backbone. We present the analytic as well as simulation study of our algorithm and also a method for the dynamic maintenance of constructed backbone. In the end, we illustrate the use of the virtual backbone with the help of an interesting application.

1. Introduction

1.1. The background

The mobile ad hoc network (MANET) is a very popular and challenging computing environment to work with. The computational capability, stable storage, power backup, and communication range of the mobile nodes are limited. Furthermore, the nodes being mobile and susceptible to various other types of failures like, mechanical damage and theft, the ad hoc networks need frequent coordination. Hence, the mobile applications that involve multiple nodes usually require a node to act as coordinator (a.k.a. leader). However, if there is single leader for the whole network, then coordination incurs high network overhead. Moreover, the majority of routing algorithms use flooding to perform the discovery and updates related to routes, though some routing algorithms, (e.g. Khamayseh et al., 2011) have attempted to reduce the adverse effects of flooding by confining the rebroadcast messages to the quasi-mobile and lightly loaded nodes. Therefore, by dividing the network into small sub-networks (a.k.a. clusters), both of the above mentioned problems can be mitigated with more convenience. The clustering simplifies coordination and significantly reduces the flooding overhead.

The key idea is to store the topology related information on some *selected subset* of nodes, rather than storing on all nodes,

* Corresponding author.
E-mail addresses: bharti_kanhiya@yahoo.co.in (B. Sharma), rsibhatia@yahoo.co.in (R.S. Bhatia), aksinreck@rediffmail.com (A.K. Singh).
Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2014.06.020
1319-1578

so that the routing and broadcasting responsibility may rest on the *selected subset* of nodes only. The reason is three fold: first, since no node uses entire topological details, it would not like to waste energy in gathering extraneous information; second, the nodes always prefer to send route request queries to nearby part of the network in order to have quick replies; third, in mobile ad hoc network, the topological details soon get stale due to frequent link loss and link formation among nodes. Such *selected subset* of nodes is called backbone nodes and the clusters are connected via the (virtual) backbone. The virtual backbone has been used extensively in various popular applications that include routing, broadcast, scheduling, route maintenance, point and area coverage, and topology management. The non-backbone nodes communicate by passing messages through their backbone counterparts. It constrains the redundancy, which is involved in broadcast or flooding and thus, reduces the overall power consumption.

A *selected subset* of nodes from the graph representation of a network is called dominating if all the nodes in the network are either included in the set or neighbors of nodes in the set. The nodes in dominating set are called dominators, while nodes not in dominating set are called dominatees. The dominating set problem is an NP-complete problem (Garey and Johnson, 1979). The nodes in the dominating set can be used to work as a virtual backbone over the network topology if they are connected. In this way, the backbone structure will be much sparser than the original communication graph. A dominating set is called connected 1-hop dominating set or simply, connected dominating set (CDS) if neighboring dominating nodes are 1-hop away (Dai and Wu, 2004). A CDS is a dominating set, which induces a connected subgraph. For clustering ad hoc networks, Chen-Liestman (Chen and Liestman, 2002) introduced another alternative, the use of weakly connected dominating set (WCDS). The WCDS is a set that is dominating and all the edges with at least one end point in the set form a connected subgraph (weakly induced subgraph). Due to relaxed connectivity requirement of the dominating set, the size of backbone constructed using WCDS is smaller than the size of its CDS counterpart. The small sized backbone can be used effectively in MANETs for the purpose of efficient routing and zone surveillance. However, the message overhead is toward the higher side in the backbone construction methods using CDS as well as WCDS. Furthermore, the CDS and WCDS backbones are fault prone. The fault tolerance and controlled message overhead are highly desired features in the algorithms proposed for ad hoc networks. Therefore, our motivation is to construct a robust backbone with low message overhead.

1.2. The contribution

In this paper, we propose a backbone construction algorithm. Initially, it uses a leader election algorithm for cluster formation like Alzoubi et al. (2003); however, it does not construct MST like Alzoubi et al. (2003) and Zonal algorithm (Chen and Liestman, 2003). Hence, the closest to our approach is Area algorithm (Han and Jia, 2007). However, the Area algorithm is designed for unit disk graph (UDG) and it uses WCDS as dominator set to construct small size backbone. Unlike Area algorithm, our protocol is designed for general graph and it produces robust backbone with low message overhead, though our backbone size is larger than its Area counterpart. An UDG is the simplest model of wireless ad hoc network in which each node is assumed to have the same transmission range that is represented by value 1. Even for the restricted class of UDGs, the dominating set problem is NP-hard (Cohen and Opatrny, 2009). However, we do not restrict the network nodes to have the same and unit transmission range; rather, various nodes may have a different transmission range that is non-zero integer, for simplicity. Consequently, we model the network as multi-hop communication graph, which is a more general and realistic structure. If all nodes are located in the plane and have the same transmission radius d, then graph G is called a UDG. Thus, the UDG is an instance of the generic multi-hop communication graph (Amis et al., 2000). Therefore, the set of backbone nodes produced by our algorithm does not form MIS (maximal independent set), CDS or WCDS. Moreover, minimum d-hop dominating set problem is NP-complete even for UDGs (Amis et al., 2000). Our backbone construction method consists of two phases: the first phase finds the dominators and the second phase finds a set of nodes to connect these dominators in order to construct the virtual backbone of the network. Both the phases are detailed in Section 2. Usually, both phases interleave in any normal construction, they are presented separately for easy understanding.

1.3. The basic idea

We use our message efficient leader election protocol MELFA (Singh and Sharma, 2011) to find the best node in a geographic region to become the cluster head. The MELFA is message efficient in average case; however, its message overhead is toward the higher side in the worst case, especially when a number of nodes concurrently initiate the leader election protocol and subsequently flood the network with excessive election messages. In order to get rid of this difficulty we allow no election messages to propagate beyond a selected distance (called radius R, henceforth) from the originator of the election message, where $R \geqslant 1$. The value of R is represented as the number of wireless hops and R is used as a parameter of the heuristic to have control over the size of clusters and the number of elected cluster heads in the network. The large size cluster incurs high storage and processing overhead at the cluster head as well as the cluster members because each node has to maintain complete state information about all its peer cluster members and handle the traffic generated by them. On the other hand, the small size clusters do not use the storage and processing resources available at the nodes efficiently and hence they are not able to reap the benefits of clustering (Banerjee and Khuller, 2001). Thus, a clustering scheme should construct the appropriate number of clusters of moderate size. By adjusting the value of R we can easily make our scheme scalable by optimizing the size of clusters. Since the leader election is another area of research and not the focus of this paper, we adopt our existing leader election algorithm MELFA (Singh and Sharma, 2011).

The network is assumed to have finite number of nodes distributed in a geographic area. A node has unique ID represented by a binary sequence. A node's ID may be, for example, its MAC/IP address or CPU ID (Zeng et al., 2010). The unique ID assignment is a non-trivial problem and beyond

the scope of this paper. Each node is assumed to have some weight, which is a function of its various capabilities, like computation power, residual energy, stability etc. Initially, each node locally broadcasts election message to all of its 1-hop neighbors. Each recipient of the election message acknowledges the initiator with its weight value and forwards the election message further to 1-hop neighbors that are yet to receive it. A node forwards the election message initiated by the minimal ID node when it receives multiple election instances and the remaining election instances are consumed by the node. The depth of propagation of election message from any initiator is limited to R. Thus, after a finite number of rounds, which would be upper bounded by R, all lower ID election instances would be consumed at some node. Eventually, within a region of radius R, the only election instance that was initiated by the minimal ID node survives and elects the maximal weight node as leader. Thus, the protocol ensures the election of unique leader per each region of radius R. Subsequently, the initiator propagates the leader ID to all R hop neighbors through leader message. Consequently, a node may receive multiple leader messages. A node receiving multiple leader messages determines its leader using some local heuristic based on parameters like hop distance, node weight etc. Afterward, it communicates that it is a member of the cluster to the leader. When the leader has heard from every immediate neighbor it has the information about each node in its region, called cluster. In this way each node is associated with a leader that is its cluster head and thus the clusters are non-overlapping. Furthermore, after cluster head selection each node broadcasts its elected cluster head to all of its neighbors. This step helps a node to know if it is a border node, explained later in Section 2.1. The outcome of our clustering heuristic is that a node is either a cluster head or at most $2R$ hops away from a cluster head. This marks the end of clustering phase. For more details on MELFA, the reader may refer Singh and Sharma (2011).

Any two nodes are called neighbors if they are 1-hop apart. Alternatively, they can also be called 1-hop neighbors (or 1-nieghbors, for short). Similarly, in a general graph, if two nodes are not 1-neighbors, they may belong to MIS or 1-MIS. In the present context, the cluster heads, being $(2R + 1)$ hops apart, are $(2R + 1)$-neighbors and hence there exists no other cluster head within $2R$ hop distance from a cluster head. Thus, the set of cluster heads form $2R$-MIS. In a special distribution of nodes, if each cluster head is assumed at the center of its cluster, then such $2R$-MIS can also be obtained by generalizing Luby's MIS algorithm (Luby, 1986) to R rounds. However, in general, the cluster head, being best node in its region of radius R, may not necessarily be located at the center of its cluster. The heads of two adjoining clusters may be located in the fringes of their respective cluster and hence they may be either 1-hop apart (refer Fig. 1 next) or $(4R + 1)$ hops apart (refer Fig. 2 next). Thus, the set of cluster heads may not form $2R$-MIS; therefore, the Luby's MIS algorithm (generalized to R rounds) may not be applicable here, in general (or real) setting. Once the cluster heads are elected and the process of clustering is over; we identify some nodes, in addition to cluster heads, in order to construct virtual backbone for the network. The algorithm for dominator selection is explained in Section 2.

2. Virtual backbone construction

2.1. The diameter algorithm

The above mentioned clustering mechanism partitions the network into non-overlapping clusters, the leader of each cluster becomes its cluster head, and the clusters are assigned unique ID (may be same as cluster head ID). Once the clustering process is over, each cluster head knows the ID of its neighbor cluster heads in the network and each cluster head has accumulated the complete information about all its affiliated cluster members; also, each affiliated node knows its cluster ID as well as the cluster ID of all its 1-hop neighbors.

Consider any two adjacent clusters of a clustered MANET as shown by dotted circles in the following Fig. 1. The nodes a and d are the election originators, the nodes b and f are the elected cluster heads, and the nodes c and e are ordinary

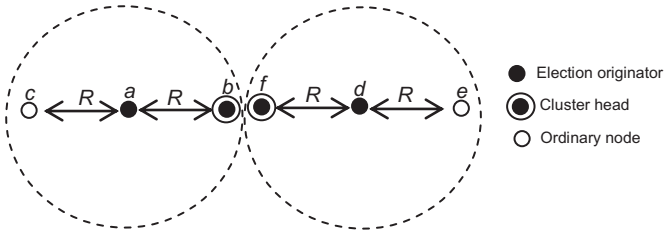

Figure 1 The cluster heads 1-hop apart.

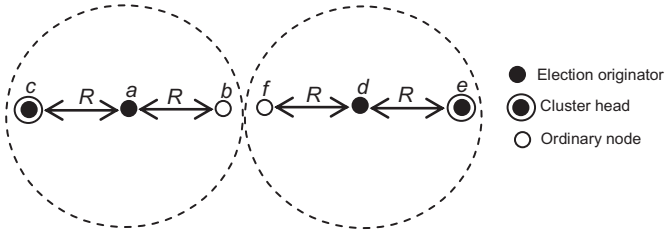

Figure 2 The cluster heads $(4R + 1)$ hops apart.

cluster member nodes in respective clusters. Since, the highest weight node is elected as cluster head, the cluster head can be located anywhere within the cluster, not necessarily at the center. In the best case, the heads of two adjacent clusters may be located close to the common cluster boundary and thus 1-hop neighbors. Hence, the distance between two cluster heads b and f would be minimum, i.e. 1-hop (refer Fig. 1). However, in the worst case, the heads of two adjacent clusters may be located at two opposite extremities of their respective clusters (refer Fig. 2). Thus, the distance between two cluster heads c and e would be maximum, i.e. $(4R + 1)$ hop because in our leader election method no node can be more than R hops away from the election originator, where $R \geqslant 1$. Both the situations are depicted in the following Figs. 1 and 2, respectively. Similarly, we can see in both Figs. 1 and 2 that in a cluster, if the cluster head and an ordinary node are located on two opposite extremities of the cluster, then the maximum distance between them can be $2R$, which we call 'diameter' in our illustration and hence, the name diameter algorithm.

In order to construct the virtual backbone, some additional nodes at the cluster borders should be added into the final backbone to connect the cluster heads. To adjust cluster borders, we need to know which nodes are in the cluster borders. A node is a border node if it has a neighbor with a different cluster ID. Once a node has identified itself as a border node it then begins a convergecast to its cluster head sending its node ID, all neighboring border nodes and their associated cluster heads. Each border node will execute the following local algorithm:

Case 1: If a cluster head is a border cluster head and its neighbor is also a cluster head, it colors the in between link black.

Case 2: If a cluster member is a border cluster member and its neighbor is a cluster head, it will send border-CM message to the neighbor cluster head. On receiving border-CM message, the cluster head sends joint-CH message and colors the in between link black. On receiving joint-CH message, the border cluster member becomes the joint-cluster head.

Case 3: All the neighbors of a border cluster member are cluster heads: The border cluster member will send border-CM message to the neighbor cluster head with lowest ID. On receiving border-CM message, the cluster head sends joint-CH message and colors the in between link black. Now, on receiving joint-CH message, the border cluster member becomes joint-cluster head; subsequently, it forwards joint-CH message to the other neighboring cluster heads and colors the in between link black. In this way, the path traversed by joint-CH message is colored black and the cluster heads at the ends of the black path become aware of the ID of their joint-cluster head.

Case 4: All the neighbors of a border cluster member are cluster members: The border cluster member will send border-CM message to lowest cluster ID neighbor that will forward the same toward its cluster heads. On receiving border-CM message, the cluster head reacts as in above case 3. However, if it receives multiple border-CM messages from an adjacent cluster ID, it is clear that there are multiple border cluster members to reach that cluster ID. Hence, it sends joint-CH message to the nearest (in terms of wireless hops) border cluster member and colors the in between link and enroute nodes black. However, for a particular cluster ID, if it has multiple border cluster members at the same lowest hop distance, then it sends joint-CH message to the cluster member with lowest ID among them and colors the in between link and enroute nodes black. On receiving joint-CH message, the border cluster member does the same as in above case 3.

If some cluster head finds that no joint-cluster head is a member of its cluster, it appoints its lowest ID cluster member as co-cluster head by sending message, co-CH and colors the in between link as well as enroute nodes black. It helps in dynamic maintenance of the backbone and will be explained later. The union of the set of cluster heads, joint-cluster heads, and co-cluster heads is called the set of backbone (a.k.a. dominator) nodes. All the black nodes become connectors for the backbone. All the connector nodes color their respective lowest ID 1-hop neighbor cluster member gray, excluding backbone nodes. Now, the construction of virtual backbone is complete. It is not the intent of the diameter heuristic to minimize the number of backbone nodes and thus our algorithm produces a relatively big size backbone. Nevertheless, the higher number of backbone nodes results in a backbone with multiple paths between neighboring cluster heads that provides fault tolerance and reduces congestion in the backbone network.

2.2. An illustration

Consider Fig. 3. The dotted curved lines signify the cluster boundaries. There are six clusters having cluster ID, say C1 through C6. The cluster heads are shown as encircled black nodes and numbers beside them represent their ID. Node A in cluster C1 is a border cluster member and thus sends border-CM message to lowest ID neighbor cluster head 12. The cluster head 12 sends joint-CH message to node A and colors in between link black. Node A becomes joint-cluster head and thus shown as shaded node. Now, it forwards joint-CH message to cluster head 75 and 32 and colors in between link black.

Nodes B and E are both border cluster members. Both will send border-CM message to cluster head 12. Now, cluster head 12 will send joint-CH message to node B only, because it is at a lower hop distance compared to node E and colors in between

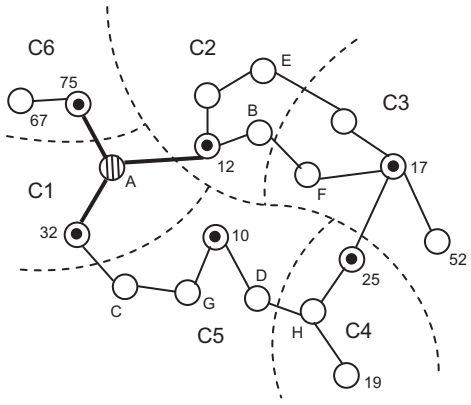

Figure 3 Node A selected joint-cluster head.

link black. Node B becomes the joint-cluster head and forwards joint-CH message to cluster head 17, colors in between link as well as enroute node F black (refer Fig. 4). Similarly, nodes C and D become joint-cluster head and nodes G and H are colored black. Nodes F, G, and H become connector nodes (refer Fig. 5). The cluster C3, C4, and C6 have joint-cluster head, respectively node B, D, and A, which is not their cluster member; hence, the heads of cluster C3, C4, and C6 appoint the node with ID 52, 19, and 67, represented as double concentric circles, as co-cluster head for their respective cluster and color in between link black, (refer Fig. 6). The black path is the virtual backbone route.

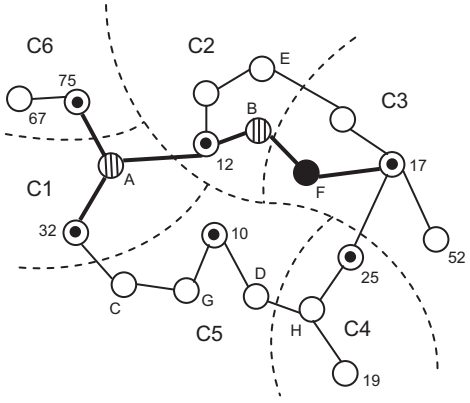

Figure 4 Node B selected joint-cluster head.

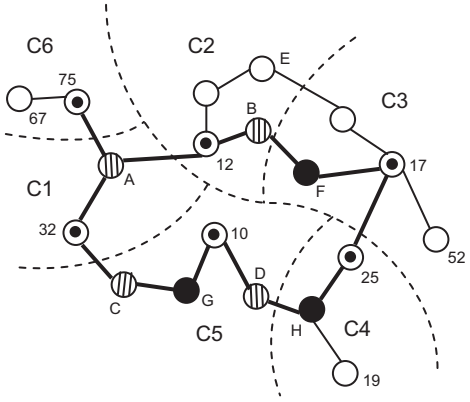

Figure 5 Nodes C and D selected joint-cluster head.

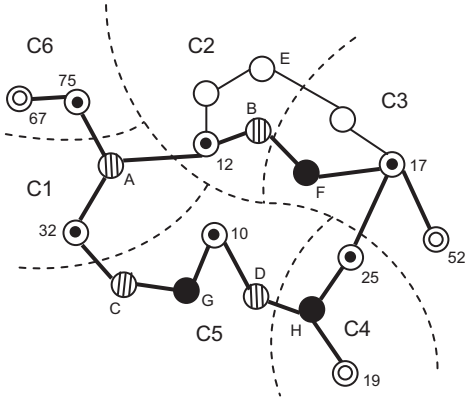

Figure 6 The selection of co-cluster heads.

3. The dynamic maintenance of backbone

Any cluster member can be at most $2R$ hops away from its cluster head. Similarly, any two adjacent (border sharing) cluster heads are at most $(4R + 1)$ hops apart. The cluster head broadcasts HELLO message periodically to its 1-hop neighbors, which acknowledge as well as forward the same to their neighbors and so on. In this way, the HELLO message reaches to all its cluster members. The HELLO message also contains the ID of all joint-cluster heads, within its cluster, which connect the cluster head with adjacent clusters. A joint-cluster head can be in either of the following two states: normal and surrogate-CH.

3.1. Link failure of ordinary node with cluster head

If a node loses link with its cluster head, then it sends adapt-REQ message to lowest ID joint-cluster head in its cluster, which will act as surrogate cluster head. It puts the node ID in its adapt-LIST, switches to surrogate-CH state, and starts forwarding HELLO message to the node. When the node starts receiving HELLO message again, from its cluster head, the node sends desert-REQ message to surrogate cluster head, which removes the node from its adapt-LIST after receiving desert-REQ message, stops forwarding HELLO message to it, and sends updated adapt-LIST to the cluster head. If its adapt-LIST becomes empty, it switches back to normal joint-cluster head state.

3.2. Inter-cluster movement of a cluster member

If an affiliated cluster member has moved to some other cluster, it discovers neighbors first and if it is within $2R$ wireless hops from any existing cluster head, then it sends join-REQ to new cluster head, which allows the requester to become its cluster member by sending HELLO message. However, if the node is beyond $2R$ wireless hops from any existing cluster head, then the node runs leader election to create a new cluster.

3.3. Movement (or crash) of joint-cluster head

If adjacent cluster heads do not hear a common joint-cluster head within a time threshold, the lowest ID cluster head among them infers that the joint-cluster head has either crashed or moved out of range and it appoints some other border cluster member connecting the adjacent clusters as new joint-cluster head.

3.4. Movement (or crash) of cluster head

On detecting the movement or crash of cluster head, based on time threshold, the lowest ID joint-cluster head of the cluster takes temporarily the charge of cluster head and requests the lowest ID cluster member to start leader election, provided the leader election is not already initiated by some other node on detecting the loss of leader *i.e.* cluster head. However, if no joint-cluster head is member of the cluster and an ordinary

node detects the loss of cluster head, it will send join request to co-cluster head that will adapt it (as does the joint-cluster head in Section 3.1). However, in the mean time based on time threshold, if co-cluster head too detects the loss of cluster head, it will initiate the leader election and act as stop gap cluster head till the election is over. Thus, the protocol guarantees the availability of a leader in every cluster at any time during the execution. If there is high churn, multiple nodes may initiate new instances of leader election protocol for the missing cluster head. However, as explained in Section 1.3 above, within a cluster the election message, initiated by the minimal ID node only, would be able to carry on and elect best node as leader. Therefore, each cluster has single unique leader.

3.5. Movement/crash of co-cluster head or connector node

When co-cluster head becomes unreachable, the cluster head appoints the next lowest ID cluster member as co-cluster head. If a gray node detects the connector node mobility or switch off, based on time threshold, it turns black and becomes connector node. However, if any connector, which has no non-backbone neighbor, becomes unreachable, this situation is equivalent to the situation arising from the switch off or mobility of backbone nodes and can be handled in the same manner as explained above.

4. The correctness proof

The correctness of our clustering approach is directly dependent on the correctness of MELFA (Singh and Sharma, 2011) that is our leader (cluster head) election algorithm. The detailed proof of MELFA is presented in Singh and Sharma (2011); however, it is being reproduced for the sake of completeness. The leader election protocols are required to elect a correct leader within finite time. This kind of execution is known as *stable leader election*. MELFA also falls in this category and satisfies safety as well as liveness properties.

Some key assumptions about the system settings are being reiterated here. We assume there are n nodes in a single connected component and the network does not partition/merge during election. The high churn may result in arbitrary topology changes including network partitioning and merging. Hence, we assume the nodes being quasi-mobile during election. The nodes may be in one of the three states: NORMAL – node performing its normal computation in the presence of the leader; CANDIDATE – node is a candidate to become leader because either it has lost link with the leader or the leader departed; LEADER – a node is in leader state if all nodes accept it as leader. Initially, each node state is assumed in NORMAL state.

4.1. Safety

Theorem: After the termination of MELA, all nodes in the network agree on the unique elected leader.

Proof: Assume the contrary. Say, there are two nodes i and j that are in NORMAL state having highest node weight and their leader ID is different. The statement holds in either of the following two cases:

Case 1: Node i and j are isolated from the network and they possess no neighbors. In this case, each node i and j elects itself as best node and become leader of own single node component, not for the entire network. It contradicts our initial assumption that there exists a single connected component.

Case 2: Node i and j lie in two disjoint components. Hence, node i and j become the leader of their respective component. However, it is in contrast with our initial assumption that there exists a single connected component.

Thus, node i and j must belong to a single connected component and the protocol elects the highest weight node as leader. Therefore, node i becomes leader of the component because in the case of same node weight the tie is broken in favor of lower ID and i is less than j in lexicographic order. This is a contradiction. Thus, the theorem holds.

4.2. Liveness

Theorem: The nodes start election in CANDIDATE state and eventually they reach NORMAL state in which they agree on the unique leader.

Proof: In order that this statement to hold, either of the following conditions must be true: (*i*) Node i losses its leader and needs a new leader, (*ii*) The leader crashed. In the first case, node i initiates MELFA and in the second case, some arbitrary node, which detected the loss of leader, initiates MELFA. Since, a time out mechanism is in place; also, there exists a single connected component and the number of nodes being finite, the election initiator node will receive the weight value from all correct nodes in finite time. Afterward, it will elect the highest weight node as leader and propagate this result in the network. Therefore, eventually, all nodes in the component become aware of the leader and switch to NORMAL state.

Since the above correctness argument holds for any ad hoc network, provided the assumptions hold, it also holds for our clustering process in which the depth of propagation of messages is limited to R, the radius of the cluster.

5. The performance study

5.1. Note on complexity

Each node propagates $3R$ rounds (R rounds of election message, R rounds of weight acknowledgment message, and R rounds of leader message) of messages to elect cluster heads, where R is the radius. Afterward, a convergecast is initiated to inform the cluster head of its children. Because each node is at most $2R$ hops from its cluster head the convergecast will be $2R$ rounds of messages. Furthermore, after cluster head selection each node broadcasts its elected cluster head to all of its neighbors and thus the broadcast will be a single round of messages. Hence, the message and time complexity of our clustering heuristic is $O(3R + 2R + 1)$ rounds, i.e. $O(R)$ rounds.

The diameter algorithm has both time and message complexity $O(R^2)$. In the worst case, all the n nodes may form a straight line (refer the following Fig. 7), though this configuration is highly unlikely in a real world application. In this case, any cluster head like b or c would have two neighbor cluster heads, one toward its left and the other toward its right, except the cluster heads a and z that are on the opposite ends of the straight line. Also, assume that each cluster head is located

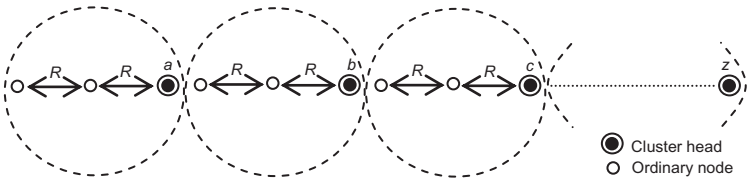

Figure 7 The nodes arranged in a straight line.

at the maximum possible hop distance from its both neighbor cluster heads. Thus, each cluster head is located on the fringe of its cluster. Therefore, any two neighbor cluster heads would be $(2R + 1)$ hops apart and each cluster would be constituted of $(2R + 1)$ nodes. If we consider $n = k \times R$, where k is some positive integer, then the total number of clusters in the network would be a function of R. Because the radius R is an input value to the heuristic, each node sends $O(R)$ rounds of messages. As the total number of clusters is a function of R; in the worst case, the diameter algorithm would execute $O(R)$ sequential steps each with $O(R)$ rounds of messages. Hence, the total number of messages exchanged is $O(R^2)$ rounds. Thus, both, the time and message complexity of the diameter algorithm is $O(R^2)$ rounds. Therefore, the message and time complexity of the backbone formation heuristic is $O(R) + O(R^2) = O(R^2)$ rounds that compares favorably to $O(n)$ for the Area algorithm (Han and Jia, 2007). The simulation results shown in the next Fig. 8 are also in conformity with this analysis.

5.2. Discussion on the convergence time

The pure localized algorithms converge only in constant number of steps (Dai and Wu, 2004). However, the convergence time of our algorithm is closely related to the cluster diameter, *i.e.* the maximum possible distance between cluster head and joint-cluster head of a cluster that is $2R$. Hence, when a new node joins (leaves) a cluster or a new communication link appears (disappears), the event notification takes $O(R^2)$ time before reaching a backbone node. Refer Section 5.3 for the detailed proof of convergence time. Informally, the event notification time corresponds to the time it takes for the update about a node or link to propagate through the cluster, so that all nodes can adjust their behavior accordingly. Thus, the diameter algorithm has larger convergence time than pure local algorithms. Nevertheless, since the cluster radius R is a value selected for the heuristic, the convergence time is upper bounded by $O(R^2)$ even in the high churn when the various updates progress in overlapped manner. Moreover, the most popular algorithms, e.g. (Alzoubi et al., 2003; Chen and Liestman, 2003) are quasi-localized algorithms. The pure localized algorithms are rare in the literature (Han and Jia, 2007).

5.3. The overhead analysis

In this section, we investigate the signaling overhead involved per topology change. The investigation is motivated by Er and Seah (2006) and Xing et al. (2008). It is a known fact that the maintenance of a logical structure consumes extra time and incurs additional messages that are over and above the price already paid to construct the logical structure. Conventionally, the message overhead is an important metric for the performance evaluation of network algorithms. Also, we compute the time complexity per topology change. We assume the random waypoint mobility model. The analysis is divided into three discrete steps with zero pause time. The following is the list of notations that are used in our analysis (Er and Seah, 2006; Xing et al., 2008):

n: the number of nodes in the network.
m: the average number of cluster members in a cluster. In the worst case, it is $(2R + 1)$ because each node can be at most $2R$ hops away from its cluster head (refer Fig. 7).
μ = average node speed.
r = transmission radius.
h_i = hop distance of node i from its cluster head.
MAX_{hop} = maximum hop distance of a node from its cluster head, which is $2R$ in our protocol.
f_{HELLO}: the number of HELLO messages broadcast by a node per time unit. According to Sucec and Marsic (2004), $f_{HELLO} = \Theta(1)$ because f_{HELLO} is proportional to the average node speed μ and inversely proportional to the transmission radius r, and both μ and r are less than or equal to some constants.

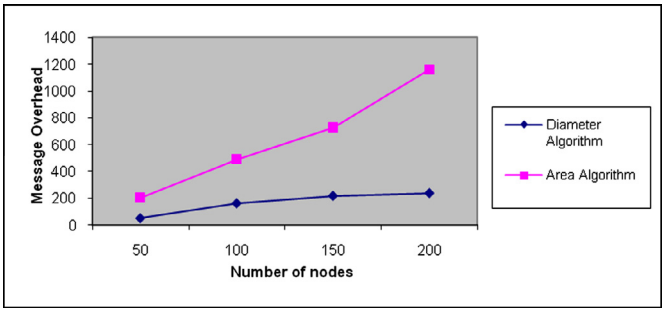

Figure 8 The number of nodes vs. message overhead.

f_{LINK}: the average frequency of topology changes occurred per time unit. According to Sucec and Marsic (2004), $f_{LINK} = \Theta(n)$. Due to Er and Seah (2006), $f_{LINK} = \Theta(\mu|E|/r)$, where E is the total number of links available in the network. Because, in the worst case (refer Fig. 7), $|E| = n\text{-}1$. Thus, we have $\Theta(\mu|n-1|/r) = \Theta(n)$.
T: the number of time units consumed by the algorithm after a topology change to re-establish a valid cluster structure (a.k.a. re-clustering).
M: the number of messages involved after a topology change to re-establish a valid cluster structure.
L: the total evaluation time.

Further, if node i and node j are members of the same cluster, then we define the notions of upstream node, downstream node, and peer node as follows: node i is upstream node of node j if $h_i < h_j$; node i is downstream node of node j if $h_i > h_j$; node i is peer node of node j if $h_i = h_j$.

(i) *The HELLO message overhead*: Each node broadcasts HELLO messages periodically for neighbor discovery and other useful state information aggregation. Thus, it introduces an overhead of $f_{HELLO} \times n$ messages per time unit for all nodes in the network.
(ii) *The backbone formation overhead*: In Section 5.1, the backbone formation overhead is computed as $O(R^2)$, where R is the radius.
(iii) *The cluster maintenance overhead*: The message and time complexity of cluster maintenance are dependent on the configuration of neighborhood at the time of occurrence of topology change. However, in the worst case (refer the above Fig. 7), it is the set of cluster heads only that form backbone nodes. Thus, in the worst case, the backbone maintenance overhead is, in fact, the cluster maintenance overhead. Therefore, in this section, we use the term cluster maintenance instead of backbone maintenance. The topological changes are detected by the periodical HELLO messages. Once a topology change or a cluster head loss occurs, the relating nodes take appropriate actions to re-establish a valid cluster structure. Now, we investigate the cluster maintenance overheads that are triggered by these events.

5.3.1. Link loss between a member and its cluster head

A link loss between nodes from different clusters or between nodes from the same cluster will not cause any re-clustering. Only the link loss between a node and its cluster head or its upstream node will trigger the re-clustering. Consequently, only the downstream node will respond to the network topology change. The cluster head or upstream node simply removes downstream node from their member lists. We denote the responding downstream node as node i. Now, there are two possibilities:

Case (1): First, we consider the base case when node i is a fringe node, i.e. it has no downstream nodes: – In the event of link loss between a member and its cluster head, the cluster head removes this node from its member list and hence no message is exchanged. The node joins another reachable cluster head, if any. We call a cluster head reachable if it is less than MAX_{hop} (i.e. $2R$, in our case) hops away from the node. This is done by choosing a neighbor that is connected to its cluster head by an unsaturated link (i.e., link that may consist of multiple hops but the hop distance is less than MAX_{hop} hops). It will then broadcast a join-REQ message in one time unit. The time needed for this decision to arrive is at most MAX_{hop} time units because the cluster head is at most MAX_{hop} hops away. Therefore, $T \leqslant MAX_{hop}$ and $M \leqslant MAX_{hop}$. Note that similar process is performed when a new node joins the network. However, if this condition fails and no cluster head is reachable, the node considers all non-clustered neighbors to form a new cluster. Therefore, the time and message complexity is the same as those in the cluster formation. The cluster formation overhead, as shown in Section 5.1, is $(5R + 1)$ rounds, i.e. time units, where R is the radius. Therefore, $T \leqslant (5R + 1)$ and $M \leqslant (5R + 1)$. However, if the node has no neighbors, a trivial case occurs. It declares itself leader (cluster head) forming a new single member cluster and broadcast its decision in the next HELLO message ($M = 1$). This process is done in one time unit. Hence, in the worst case, we have: $T \leqslant (5R + 1)$ and $M \leqslant (5R + 1)$ for one of this kind of link breaks.

Case (2): If the node i has some downstream nodes: – Each downstream node has to respond when they receive messages from their upstream node about the topology changes. This ripple action will end at the fringe node of the cluster where the above mentioned base case is executed. Then, re-clustering will be completed and a valid cluster structure is re-established. The total number of nodes that are affected by this ripple action can be at most $(MAX_{hop} - h_i + 1)$ nodes, where h_i represents the hop distance of node i from its cluster head. Thus, the upper bound on time and message complexity is, as follows:

$$T \leqslant (MAX_{hop}) \times (MAX_{hop} - h_i + 1) \text{ i.e. } T \leqslant 4R^2$$
$$M \leqslant (MAX_{hop}) \times (MAX_{hop} - h_i + 1) \text{ i.e. } M \leqslant 4R^2$$

5.3.2. Link creation between a node and its cluster head

A link creation between two member nodes will not lead to any re-clustering since both nodes are still connected to their cluster heads. Similarly, no re-clustering occurs even in case of new link creation between a member node and a cluster head as the cluster structure is still valid. However, a process similar to the link loss case will be performed. The base case occurs when the responding node i is a fringe node. Thus, the time and message complexity for the base case are same as given in above case. If the responding node has downstream nodes, each downstream node has to respond on receiving HELLO messages that indicate cluster head or topological changes. Therefore, a similar ripple action would be triggered that will also end at the fringe node. Because each node can be at most MAX_{hop} hops away from its cluster head, the ripple action may take place at all cluster members and its propagation depth will be limited to MAX_{hop} hops. To summarize, after a link creation event, we have message and time complexity upper bounds that are almost similar to the link loss case:

$T \leqslant MAX_{hop} \times MAX_{hop}$ i.e. $T \leqslant 4R^2$
$M \leqslant m \times MAX_{hop}$ i.e. $M \leqslant 4R^2$, as the value of m is $(2R + 1)$ in the worst case (refer Fig. 7).

After receiving the join-REQ message from a node the new cluster head allows the requester to become its cluster member by sending HELLO message. The node joins the new cluster successfully. To summarize, $T = 2R$ and $M = 2R$ for one of this kind of link creations.

5.3.3. Link loss due to the loss of a cluster head

This event triggers a cluster head re-election. Thus, the overhead would be same as the cluster formation overhead. In Section 5.1, it is shown as $O(5R + 1)$, where R is the radius. Thus, $T = (5R + 1)$ and $M = (5R + 1)$.

Total cluster maintenance overhead: The message overhead M is upper bounded by $4R^2$ in both the cases, i.e. in the case of link loss between a member and its cluster head as well as in the case of link creation between a node and its cluster head. Thus, the total number of messages transmitted per topology change due to link state changes would be upper bounded by $8R^2$. Also, the average topology changes occurred per time unit is f_{LINK}. Therefore, there are totally $f_{LINK} \times 8R^2$ messages per time unit due to link state changes.

There are $n/(2R + 1)$ cluster heads (refer Fig. 7) and hence there could be at most $n/(2R + 1)$ cluster head losses in an evaluation. An evaluation period consists of L time units, thus, the average number of cluster head losses per time unit would be $n/\{(2R + 1)L\}$. Therefore, the total number of messages is $\{n(5R + 1)\}/\{(2R + 1)L\}$ per time unit due to the cluster head losses. In summary, total cluster maintenance overhead is $[(f_{LINK} \times 8R^2) + \{n(5R + 1)\}/\{(2R + 1)L\}]$ messages per time unit.

Total message overhead: The total message overhead (O_T) is the sum of the overhead due to HELLO messages, the overhead due to cluster formation and the overhead due to cluster maintenance, that is,

$$O_T = f_{HELLO} \times n + O(R^2) + [(f_{LINK} \times 8R^2) + \{n(5R + 1)\}/\{(2R + 1)L\}]$$

Since $f_{HELLO} = \Theta(1)$, $f_{LINK} = \Theta(n)$, $O(R^2) = O(1)$ as radius R is a predefined value for the heuristic, L is an integer and $L > 1$, given some constants k_1, k_2, k_3, and k_4 we have: $f_{HELLO} \leqslant k_1$, $f_{LINK} \leqslant k_3 \times n$, and $(1/L) < 1$. Therefore, the total message overhead (O_T) can be expressed as follows:

$$\begin{aligned}
&O_T = f_{HELLO} \times n + O(R^2) + [(f_{LINK} \times 8R^2) + \{n(5R + 1)\}/\{(2R + 1)L\}]\\
&\Rightarrow \quad O_T \leqslant f_{HELLO} \times n + O(R^2) + [(f_{LINK} \times 8R^2) + \{n(5R + 1)\}/\{(2R + 1)L\}]\\
&\Rightarrow O_T \leqslant k_1 \times n + k_2 + [k_3 \times n + k_4 \times n]\\
&\Rightarrow O_T \leqslant [(k_1 + k_3 + k_4) \times n] + k_2\\
&\Rightarrow O_T = O(n)
\end{aligned}$$

After dividing $O(n)$ by the number of nodes n, the message overhead O_T is $O(1)$ per time unit per node. Therefore, the diameter-hop clustering has the advantages of multi-hop clusters; nevertheless, the overhead incurred has the asymptotic bound similar to 1-hop clustering.

The time complexity $T \leqslant 4R^2$ in the case of link break between a member and its cluster head as well as in the case of link creation between a member and its cluster head. Although, $T = (5R + 1)$ in the case of link loss due to the loss of a cluster head, the convergence time is at most $4R^2$ time units per topology change.

5.4. The connectivity analysis

The algorithm forms a diameter-hop dominating set for the network. As no cluster member node can be more than R hops away from the election originator, the minimum distance between the heads of adjacent clusters is 1-hop and in the worst case it is $(4R + 1)$ hops (refer Figs. 1 and 2). Thus, any cluster member node can be reached within $2R$ wireless hops from at least one backbone node. Thus, all the backbone nodes form a diameter-hop (called D-hop, henceforth) dominating set. Moreover, any two heads of adjacent clusters are either 1-hop neighbors or connected by multi-hop virtual link passing via joint-cluster head and connectors. Thus, the diameter algorithm forms a connected D-hop dominating set (CDDS, for short) for the network. Also, the clusters being non-overlapping, *i.e.* no cluster member belongs to more than one cluster simultaneously, the cluster heads form independent set.

5.5. The simulation results

The simulation experiments have been performed with the diameter algorithm for clustered mobile ad hoc networks with varying size and density using NS2. The number of nodes in the simulated system varies as 50, 100, 150 and 200. The nodes are randomly distributed in an area of 350×350 units. The density μ of the graph can be calculated as $\mu(r) = n\Pi r^2/A$, where n is the number of nodes in the graph, A is the area of the graph and r is the transmission range. Thus, $\mu(r)$ represents the average number of nodes within the transmission radius of each node. This allows us to test our algorithm with varying network density by considering the nodes with different transmission range, r. Our values range from $r = 75$, and $\mu(r) = 14$ (sparse network) to $r = 200$, and $\mu(r) = 102$ (dense network).

The parameters considered for the evaluation of the protocol are (a) the overhead due to control messages (b) the size of the backbone and (c) the backbone robustness, i.e. the number of backbone nodes whose removal causes disconnection of the backbone. Firstly, we compare the number of control messages required by our diameter algorithm and those required by the area algorithm. The diameter algorithm requires significantly less number of messages than the Area algorithm as demonstrated by the following Fig. 8.

The next Fig. 9 compares the size of backbone formed for networks with varying number of nodes due to the area algorithm and the diameter algorithm. The size of the backbone is higher when constructed by the diameter algorithm since a connected backbone is being formed as compared to the Area algorithm which is constructing a weakly connected dominating set (WCDS). In the diameter algorithm, if any two cluster heads (CHs) are not in direct transmission range, the intermediate nodes are chosen as joint CHs in order to form a connected backbone. In the Area algorithm, CHs at a distance of up to 2 hops do not require any border adjustment. However, in case, two CHs are at a distance of 3 hops, one other CH is chosen as additional CH.

The network density increases as the transmission range of mobile nodes increases. As the density of nodes increases, the size of the network backbone decreases as expected and

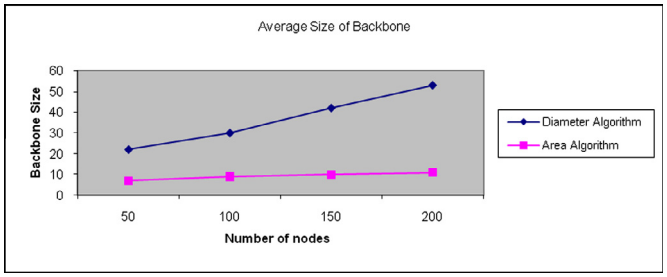

Figure 9 The number of nodes vs. backbone size.

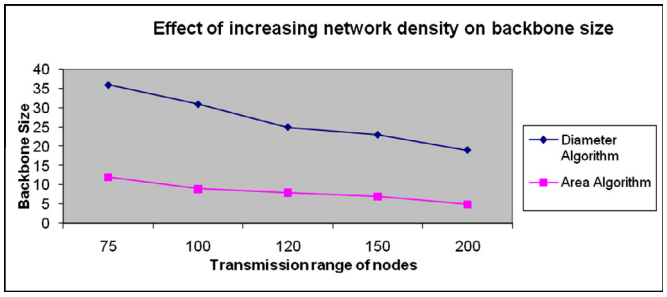

Figure 10 The network density vs. backbone size.

demonstrated by the Fig. 10. However, the decrease in backbone size is more significant in the case of diameter algorithm as the possibility of CHs being connected via shorter paths (less number of hops) increases in a dense network.

The next Fig. 11 shows the effect of increasing network density on the control message overhead due to both the algorithms. On increasing the network density, the number of control messages required for the formation of backbone by the Area algorithm increases while the number of messages required for diameter algorithm decreases. The number of control messages required for CH selection in the Area algorithm is proportional to the number of neighbors of each node and therefore increases with increasing node density. On the other hand, as the node density increases, need for border adjustment decreases in the diameter algorithm; thereby lowering the control messages required.

Finally, the diameter algorithm delivers more robust backbone. In order to substantiate this claim, we consider the robustness of the backbone created by the diameter algorithm in a network comprising of 100 nodes. The following Fig. 12 shows the number of backbone nodes removed due to reasons like failure, energy depletion etc. and the corresponding number of backbone nodes disconnected from the backbone. The number of affected nodes increases much slower than the number of backbone node removals.

However, the Area algorithm provides connectivity using WCDS, and thus there are no alternate (or multiple) path between any two CHs. Therefore, the removal of any single node from the WCDS would cause the set of dominator nodes to lose the WCDS property that leads to backbone disconnection. The following Table 1 summarizes the findings of our analytic and simulation study and presents a comparative overview of both the algorithms.

6. A mutual exclusion protocol for mobile networks using virtual backbone

In this section, we present an interesting application of our virtual backbone protocol. The period of time when a process accesses a shared resource is called critical section (CS) and the portion of code executed while using shared resource is called critical code. Thus, the act of using a shared resource is also called executing critical code or accessing CS. The control algorithm to avoid contention for CS is called mutual

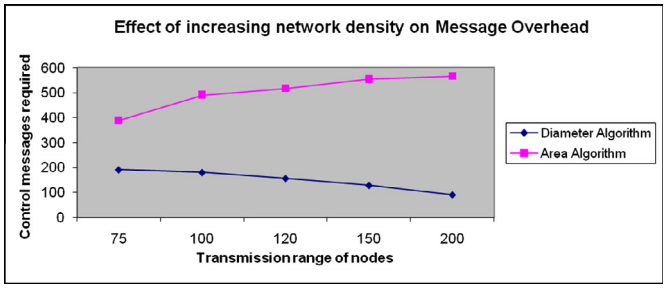

Figure 11 The network density vs. message overhead.

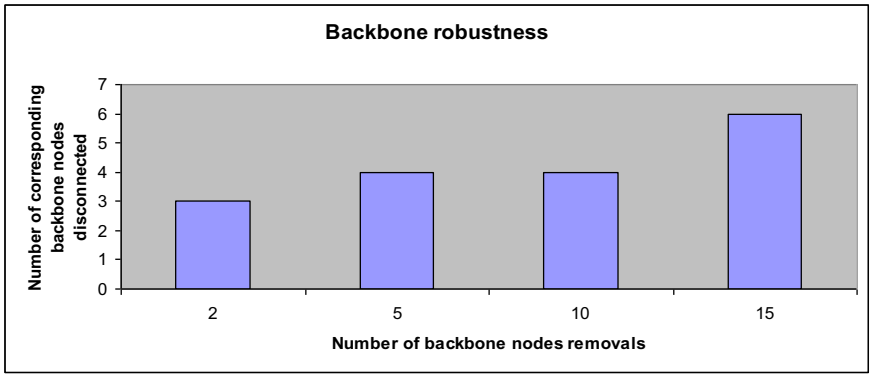

Figure 12 The node removals vs. nodes disconnected.

Table 1 The comparison of area and diameter algorithm.

Sr. no.	Properties	Area algorithm	Diameter algorithm
1	Logical structure	UDG	General graph
2	Dominating set	WCDS	CDDS
3	Approximation ratio	Constant	Not constant
4	Backbone size	Small	Large
5	Robustness	Poor	High
6	Message overhead	$O(n)$	$O(R^2)$
7	Time complexity	$O(n)$	$O(R^2)$

exclusion protocol (MUTEX, for short). There are two major approaches to design MUTEX protocols: permission based and token based. In permission based approach, the interested site requests other sites in the system for permission to access CS. If allowed, it enters CS. However, in token based approach, the system contains a unique token message that circulates among sites. The propagation of token is to transfer privilege to enter CS from one site to another. Since there is unique token in the system, the CS is accessed by no more than one site at a time and hence mutual exclusion is guaranteed. More details on mutual exclusion can be found in Kshemkalyani and Singhal (2008). We can run following simple token based MUTEX protocol over the constructed virtual backbone. The token has following format:

```
Token: <type, sender, receiver, UML, UHL>
type: a variable that may assume value privilege,
pass, or c-c. Intuitively, c-c stands for cluster-
change
sender: id of the sender of token
receiver: id of the destination of token
UML: list of unserved cluster members in the
current cluster
UHL: list of unserved cluster heads in the system
```

Data structures maintained at each cluster head:

```
CH-list: the list of all cluster heads in the system
CM-list: the list of all members of its own cluster
```

Data structure maintained at each node other than cluster head:

```
parent: id of the sender of token containing token.
type = privilege
```

The following three procedures are used in the main protocol that is executed at each node. A node that needs shared resource is called hungry. The previously constructed (virtual) backbone has been called 'black route' throughout the pseudo code.

```
Procedure Serve ();
if hungry then enter CS; on exit CS remove own id
from token.UML
else remove own id from token.UML
Procedure Explore ();
if token.UML = ∅ then token.type ← pass; return
token to parent;
else
    begin
    if no member of token.UML is reachable then
token.type ← pass;
    return token to parent
  else token.type ← privilege; forward token to a
reachable node
    in token.UML
    end
Procedure Cluster-Change ();
remove own id from token.UHL;
if token.UHL = ∅ then token.UHL ← CH-list; token.
type ← c-c;
  forward token on black route toward a joint-
cluster head that
  connects a member of token.UHL //start new round
of token
  circulation//
else token.type ← c-c; forward token on black route
toward a
  joint-cluster head that connects a member of
token.UHL
Initialize: token.type ← c-c; token.sender ← ∅;
token.UML ←
      ∅;
        token.UHL ← CH-list; hand over the
initialized
      token to an arbitrary cluster head.
```

The node receiving token reacts as follows depending on the type of node and the type of token received.

Ordinary node or co-cluster head:

```
if token.type = privilege then parent ← token.
sender; to
  ken.sender ← own id; Serve (); Explore ()
if token.type = pass then token.sender ← own id;
Explore ()
if token.type = c-c then token.sender ← own id;
forward token
  to next node on black route
```

Joint-cluster head:

```
if token.type = privilege then parent ← token.
sender; to
  ken.sender ← own id; Serve (); Explore ()
if token.type = pass then token.sender ← own id;
Explore ()
if token.type = c-c then token.sender ← own id;
forward token
  on black route toward a member of token.UHL
```

Cluster head:

```
if token.type = pass then
begin
if token.UML = ∅ then token.sender ← own id;
Cluster-Change ()
else if no member of token.UML is reachable then //
some members
    of the cluster have roamed out and CM-list has
not been updated yet//
  begin
  token.UML ← ∅; token.sender ← own id; Cluster-
Change ()
  else token.sender ← own id; token.type ← privilege;
  forward token to a reachable member of token.UML
  end
end
if token.type = c-c then
begin
token.UML ← CM-list; token.sender ← own id;
  if hungry then enter CS; on exit CS remove own id
from
    token.UML
  else remove own id from token.UML
token.type ← privilege; forward token to a
reachable member of token.UML
end
```

6.1. An illustrating example

The following Fig. 13 contains three non-overlapping clusters C1, C2, and C3 having cluster heads A, B, and C respectively. Node D is joint-cluster head. Node J, K, and L are connectors. The blank circles represent cluster members. In Fig. 10, for the sake of clarity, we have not shown any co-cluster head because

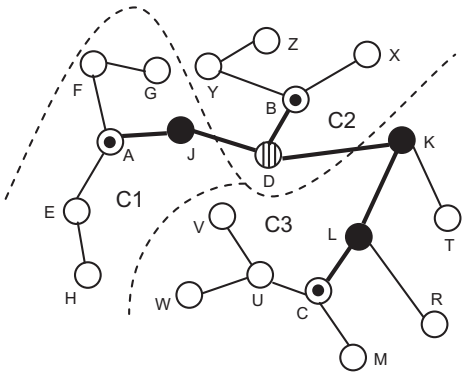

Figure 13 An illustration of mutual exclusion.

they react to token in the same way as ordinary cluster members do. Let, the initialized token is passed to cluster head A that circulates it among its cluster members. After serving its cluster, node A passes the token toward joint-cluster head D. Cluster C2 and C3 are still unserved clusters in the current round of token circulation. Now, node D passes token to one of them, say to C2. Hence, cluster head B gets the token and serves its cluster. Afterward, cluster head B passes token back to joint-cluster head D that passes it to cluster C3, the only cluster yet to be served in the current round of token circulation. Once cluster C3 too has been served, its cluster head C finds that the list of unserved cluster heads is empty. This marks the end of current round of token circulation. Now, cluster head C puts the id of all cluster heads in the empty list of unserved cluster heads that is contained in the token and passes the token toward the joint-cluster head D, to start the next round of token circulation.

6.2. Discussion on the performance of mutual exclusion protocol

Assume a system of n mobile nodes. Our mutual exclusion protocol ensures that by the time a circulation of token is complete, every mobile in the system would have got an opportunity to access critical section. Thus, the total number of token forwarding messages will be proportional to n, the number of nodes in the system. Therefore, our mutual exclusion protocol is $O(n)$ in time as well as in the number of messages.

The message propagation delay between two nodes has been assumed to be bounded. However, if the messages, related to the maintenance of cluster and virtual backbone, use the same channel as the mutual exclusion protocol messages, then collisions may occur. Hence, it is preferable to have a dedicated channel for the mutual exclusion protocol; so that, only one mobile can broadcast during a time slot. In fact, it is also a mutual exclusion problem.

7. Conclusions

The most important feature of our approach is that the construction and maintenance of virtual backbone is simple. Also, the impact of node mobility is less because the heuristic parameter $R \geqslant 1$ (usually 3 to 5) and hence unless a node makes wide area movement, it does not experience cluster change. Any

event, e.g. crash, mobility or switch off, affects a very small fraction of network that is confined within $2R$ hops from the site of the event in single dimension. Further, the value of R can be adjusted to optimize the size of clusters and the diameter-hop clustering exploits the benefits of multi-hop clusters while being able to restrict the overheads like in 1-hop clustering. The backbone is interconnected via multi-hop virtual links between backbone nodes and the route related updates are piggybacked on HELLO message; thus, the cluster members always have up-to-date route information within $2R$ hops. Therefore, any routing algorithm can be used to maintain connectivity among all the backbone nodes that treat multi-hop virtual link between two adjacent backbone nodes as if it were 1-hop link. Secondly, since our definition of radius is multi-hop, there are multiple border cluster members between two adjacent clusters. Hence, there are multiple candidate joint-cluster heads. Thus, it is easy to provide robust connectivity, which renders backbone stability, using multipath or alternate path, between cluster heads or any pair of source and target nodes.

References

Alzoubi, K.M., Wan, P.J., Frieder, O., 2003. Maximal independent set, weakly-connected dominating set, and induced spanners in wireless ad hoc networks. Int. J. Found. Comput. Sci. 14, 287–303.

Amis, A.D., Prakash, R., Voung, T.H.P., Huynh, D.T., 2000. Max-min D-cluster formation in wireless ad hoc networks. IEEE INFOCOM 1, 32–41.

Banerjee, S., Khuller, S., 2001. A clustering scheme for hierarchical control in multi-hop wireless networks. In: 20th IEEE INFOCOM, pp. 1028–1037.

Chen, Y.P., Liestman, A.L., 2002. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In: 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing. MobiHoc'02, pp. 165–172.

Chen, Y.Z., Liestman, A.L., 2003. A zonal algorithm for clustering ad hoc networks. Int. J. Found. Comput. Sci. 14, 305–322.

Cohen, Y., Opatrny J., 2009. A local algorithm for dominating sets of quasi-unit disc graph. In: 2nd Canadian Conference on Computer Science and Software Engineering. C3S2E-09, pp. 223–231.

Dai, F., Wu, J., 2004. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. IEEE Trans. Parallel Distrib. Syst. 15, 908–920.

Er, I., Seah, W., 2006. Clustering overhead and convergence time analysis of the mobility-based multi-hop clustering algorithm for mobile ad hoc networks. J. Comput. System Sci. 72 (7), 1144–1155. Also appeared in the Proc. 11th International Conference on Parallel and Distributed Systems ICPADS'05. 2, 130–134.

Garey, M., Johnson, D., 1979. Computers and Intractability – A Guide to the Theory of NP-Completeness. Freeman.

Han, B., Jia, W., 2007. Clustering wireless ad hoc networks with weakly connected dominating set. J. Parallel Distrib. Comput. 67, 727–737.

Khamayseh, Y., Obiedat, G., Yassin, M.B., 2011. Mobility and load aware routing protocol for ad hoc networks. J. King Saud Univ. – Comput. Inf. Sci. 23 (2), 105–111.

Kshemkalyani, A., Singhal, M., 2008. Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press, NY, 327–336.

Luby, M., 1986. A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput. 15 (4), 1036–1053. Also appeared in Proc. of the 17th ACM Symposium on Theory of Computing STOC'85, 1–10.

Singh, A.K., Sharma, S., 2011. Message efficient leader finding algorithm for mobile ad hoc networks. IEEE-IAMCOM'11, pp. 1–6.

Sucec, J., Marsic, I., 2004. Hierarchical routing overhead in mobile ad hoc networks. IEEE Trans. Mobile Comput. 3 (1), 46–56. Also appeared in Proc. of the 21st IEEE INFCOM'02, 1698–1706.

Xing, Z., Gruenwald, L., Phang, K., 2008. Next Generation Mobile Networks and Ubiquitous Computing. S. Pierre (Ed.), IGI Global Press, Chapter 18, 187–200.

Zeng, Y., Mittal, N., Venkatesan, S., Chandrasekaran, R., 2010. Fast neighbor discovery with lightweight termination detection in heterogeneous cognitive radio networks. In: 9th International Symposium on Parallel and Distributed Computing, pp. 149–156.