Max-Min D-Cluster Formation in Wireless Ad Hoc Networks

Alan D. Amis **Ravi Prakash** **Thai H.P. Vuong** **Dung T. Huynh**
Department of Computer Science
University of Texas at Dallas
Richardson, Texas 75083-0688
E-mail: aamis@telogy.com, ravip@utdallas.edu, tvuong@nortelnetworks.com, huynh@utdallas.edu

Abstract— **An ad hoc network may be logically represented as a set of clusters. The clusterheads form a d-hop dominating set. Each node is at most d hops from a clusterhead. Clusterheads form a virtual backbone and may be used to route packets for nodes in their cluster. Previous heuristics restricted themselves to 1-hop clusters. We show that the minimum d-hop dominating set problem is NP-complete. Then we present a heuristic to form d-clusters in a wireless ad hoc network. Nodes are assumed to have non-deterministic mobility pattern. Clusters are formed by diffusing node identities along the wireless links. When the heuristic terminates, a node either becomes a clusterhead, or is at most d wireless hops away from its clusterhead. The value of d is a parameter of the heuristic. The heuristic can be run either at regular intervals, or whenever the network configuration changes. One of the features of the heuristic is that it tends to re-elect existing clusterheads even when the network configuration changes. This helps to reduce the communication overheads during transition from old clusterheads to new clusterheads. Also, there is a tendency to evenly distribute the mobile nodes among the clusterheads, and evently distribute the responsibility of acting as clusterheads among all nodes. Thus, the heuristic is fair and stable. Simulation experiments demonstrate that the proposed heuristic is better than the two earlier heuristics, namely the LCA [1] and Degree based [11] solutions.**

I. INTRODUCTION

Ad hoc networks (also referred to as packet radio networks) consist of nodes that move freely and communicate with other nodes via wireless links. One way to support efficient communication between nodes is to develop a wireless backbone architecture [1], [2], [4], [8]. While all nodes are identical in their capabilities, certain nodes are elected to form the backbone. These nodes are called clusterheads and gateways. Clusterheads are nodes that are vested with the responsibility of routing messages for all the nodes within their cluster. Gateway nodes are nodes at the fringe of a cluster and typically communicate with gateway nodes of other clusters. The wireless backbone can be used either to route packets, or to disseminate routing information, or both.

Due to the mobility of nodes in an ad hoc network, the backbone must be continuously reconstructed in a timely fashion, as the nodes move away from their associated clusterheads. The election of clusterheads has been a topic of many papers as described in [1], [2], [8]. In all of these papers the leader election guarantees that no node will be more than one hop away from a leader. Furthermore, their time complexity is $O(n)$, where n is the number of nodes in the network. Our work started with the aim of generalizing the clustering heuristics so that a node is either a clusterhead or at most d hops away from a clusterhead.

We prove that constructing the minimum d-hop dominating set in an ad hoc network is NP-complete. Then, we propose a new distributed leader election heuristic for an ad hoc network, guaranteeing that no node is more than d hops away from a leader, where d is a value selected for the heuristic. Thus, this heuristic extends the notion of cluster formation. Existing 1-hop clusters are an instance of the generic d-hop clusters. The proposed heuristic provides load balancing among clusterheads to insure a fair distribution of load among clusterheads. Additionally, the heuristic elects clusterheads in such a manner as to favor their re-election in future rounds, thereby reducing transition overheads when old clusterheads give way to new clusterheads. However, it is also fair as a large number of nodes equally share the responsibility for acting as clusterheads. Furthermore, this heuristic has time complexity of $O(d)$ rounds which compares favorably to $O(n)$ for earlier heuristics [1], [4] for large mobile networks. This reduction in time complexity is obtained by increasing the concurrency in communication. Simulation experiments support these claims. Thus, it is an improvement over other known heuristics.

II. SYSTEM MODEL

In an ad hoc network all nodes are alike and all are mobile. There are no base stations to coordinate the activities of subsets of nodes. Therefore, all the nodes have to collectively make decisions. All communication is over wireless links. A wireless link can be established between a pair of nodes only if they are within wireless range of each other. The Max-Min heuristic only considers bidirectional links. It is assumed the MAC layer will mask unidirectional links and pass bidirectional links to Max-Min. Beacons could be used to determine the presence of neighboring nodes. After the absence of some number of successive beacons from a neighboring node, it is concluded that the node is no longer a neighbor. Two nodes that have a wireless link will, henceforth, be said to be 1 wireless hop away from each other. They are also said to be immediate neighbors. Communication between nodes is over a single shared channel. The Multiple Access with Collision Avoidance (MACA) protocol [14] may be used to allow asynchronous communication while avoiding collisions and retransmissions over a single wireless channel. MACA utilizes a *Request To Send/Clear To Send (RTS/CTS)* handshaking to avoid collision between nodes.

A modified MACA protocol, MACA-BI (By Invitation) [6], suppresses all RTS and relies solely on CTS, invitations to transmit data. Simulation experiments show MACA-BI to be superior to MACA and CSMA in multi-hop networks. Other protocols such as spatial TDMA [10] may be used to provide MAC layer communication. Spatial TDMA provides deterministic performance that is good if the number of nodes is kept relatively small. However, spatial TDMA requires that all nodes be known and in a fixed location to operate. In ad hoc networks the nodes within each neighborhood are not known *a priori*. Therefore, spatial TDMA is not a viable solution initially. We suggest that MACA-BI be used initially for this heuristic to establish clusterheads and their associated neighborhoods. Then the individual cluster may transition to spatial TDMA for inter-cluster and intra-cluster communication.

All nodes broadcast their node identity periodically to maintain neighborhood integrity. Due to mobility, a node's neighborhood changes with time. As the mobility of nodes may not be predictable, changes in network topology over time are arbitrary. However, nodes may not be aware of changes in their neighborhood. Therefore, clusters and clusterheads must be updated frequently to maintain accurate network topology.

Definition 1 (d-neighborhood) - The d-neighborhood of a node is the set of all nodes that are within d hops of the node. This includes the node itself. Thus, the 0-neighborhood is only the node itself.

III. Previous Work and Design Choices

There are two heuristic design approaches for management of ad hoc networks. The first choice is to have all nodes maintain knowledge of the network and manage themselves [7], [12], [13]. This circumvents the need to select leaders or develop clusters. However, it imposes a significant communication responsibility on individual nodes. Each node must dynamically maintain routes to the rest of the nodes in the network. With large networks the number of messages needed to maintain routing tables may cause congestion in the network. Ultimately this traffic will generate huge delays in message propagation from one node to another. This approach will not be considered in the remainder of this paper.

The second approach is to identify a subset of nodes within the network and vest them with the extra responsibility of being a leader (clusterhead) of certain nodes in their proximity. The clusterheads are responsible for managing communication between nodes in their own neighborhood as well as routing information to other clusterheads in other neighborhoods. Typically, backbones are constructed to connect neighborhoods in the network. Past solutions of this kind have created a hierarchy where every node in the network was no more than 1 hop away from a clusterhead [1], [4], [10]. In large networks this approach may generate a large number of clusterheads and eventually lead to the same problem as stated in the first design approach. Therefore, it is desirable to have control over the clusterhead density in the network.

Furthermore, some of the previous clustering solutions have relied on synchronous clocks for exchange of data between nodes. In the Linked Cluster Algorithm [1], LCA, nodes communicate using TDMA frames. Each frame has a slot for each node in the network to communicate, avoiding collisions. For every node to have knowledge of all nodes in it neighborhood it requires $2n$ TDMA time slots, where n is the number of nodes in the network. A node x becomes a clusterhead if at least one of the following conditions is satisfied: (i) x has the highest identity among all nodes within 1 wireless hop of it, (ii) x does not have the highest identity in its 1-hop neighborhood, but there exists at least one neighboring node y such that x is the highest identity node in y's 1-hop neighborhood. Later the LCA heuristic was revised [5] to decrease the number of clusterheads produced in the original LCA. In this revised edition of LCA (LCA2) a node is said to be *covered* if it is in the 1-hop neighborhood of a node that has declared itself to be a clusterhead. Starting from the lowest id node to the highest id node, a node declares itself to be a clusterhead if among the non-covered nodes in its 1-hop neighborhood, it has the lowest id.

The LCA heuristic was developed and intended to be used with small networks of less than 100 nodes. In this case the delay between node transmissions is minimal and may be tolerated. However, as the number of nodes in the network grows larger, LCA will impose greater delays between node transmissions in the TDMA communication scheme and may be unacceptable. Additionally, it has been shown [15] that as communications increase the amount of skew in a synchronous timer also increases, thereby degrading the performance of the overall system or introducing additional delay and overhead.

Other solutions base the election of clusterheads on degree of connectivity [11], not node id. Each node broadcasts the nodes that it can hear, including itself. A node is elected as a clusterhead if it is the highest connected node in all of the *uncovered* neighboring nodes. In the case of a tie, the lowest or highest id may be used. As the network topology changes this approach can result in a high turnover of clusterheads [8]. This is undesirable due to the high overhead associated with clusterhead change over. Data structures have to be maintained for each node in the cluster. As new clusterheads are elected these data structures must be passed from the old clusterhead to the newly elected clusterhead. Re-election of clusterheads could minimize this network traffic by circumventing the need to send these data structures.

IV. Contributions

The main objective was to develop a heuristic that would elect multiple leaders in large ad hoc networks of thousands of nodes. Additionally, we wished to generalize the cluster definition to a collection of nodes that are up to d hops away from a clusterhead, where $d \geq 1$, $i.e.$, a d-hop dominating set. First, we show that forming a minimum d-hop dominating set is NP-complete. Then we propose a heuristic to solve the problem. Some of the design goals and contributions of this heuristic are:

1. Nodes asynchronously run the heuristic: no need for synchronized clocks,
2. Limit the number of messages sent between nodes to $O(d)$,
3. Minimize the number and size of the data structures required to implement the heuristic,
4. Minimize the number of clusterheads as a function of d,
5. Formation of backbone using gateways,
6. Re-elect clusterheads when possible: *stability*.

7. Distribute responsibility of managing clusters is equally distributed among all nodes: *fairness*.

Due to the large number of nodes involved, it is desirable to let the nodes operate asynchronously. The clock synchronization overhead is avoided, providing additional processing savings. Furthermore, the number of messages sent from each node is limited to a multiple of d, the maximum number of hops away from the nearest clusterhead, rather than n, the number of nodes in the network. This guarantees a good controlled message complexity for the heuristic. Additionally, because d is an input value to the heuristic, there is control over the number of clusterheads elected or the density of clusterheads in the network. The amount of resources needed at each node is minimal, consisting of four simple rules and two data structures that maintain node information over $2d$ rounds of communication. Nodes are candidates to be clusterheads based on their node id rather than their degree of connectivity. As the network topology changes slightly the node's degree of connectivity is much more likely to change than the node's id relative to its neighboring nodes. As will be described below, if a node A is the largest in the d-neighborhood of another node B, then node A will be elected a clusterhead, even though node A may not be the largest in its d-neighborhood. This provides a smooth and deliberate transition of clusterheads rather than an erratic exchange of leadership. This last design goal is intended to help minimize the amount of data that must be passed from an outgoing clusterhead to a new one when there is a change over.

V. NP Completeness of D-hops Dominating Set

An ad hoc network can be modeled as a graph $G = (V, E)$, where two nodes are connected by an edge if they can communicate with each other. If all nodes are located in the plane and have the same transmission radius d, then G is called a *unit disk graph*. Clearly, unit disk graphs are the simplest model of ad hoc networks.

A set S of nodes in $G = (V, E)$ is called a *d-hops dominating set* if every node in V is at most d ($d > 1$) hops away from a vertex in S. *Minimum d-hops dominating set* is the problem of determining for a graph G and an integer $k \geq 0$ whether G has a dominating set of size $\leq k$. In this section we show that the minimum d-hops dominating set problem is NP-complete. In fact, we will prove that minimum d-hops dominating set is NP-complete even for unit disk graphs.

Theorem: Minimum d-hops dominating set is NP-complete for unit disk graphs.

Proof: Since it is obvious that the minimum d-hops dominating set problem is in NP, it remains to show that it is NP-hard. We will construct a reduction from the (1-hop) dominating set problem for planar graphs with maximum degree 3 which was shown to be NP-complete in [9]. To this end, we make use of the following result which shows how planar graphs can be efficiently embedded into the Euclidian plane [16]:

A planar graph with maximum degree 4 can be embedded in the plane using $O(|V|)$ area in such a way that its vertices are at integer coordinates and its edges are drawn so that they are made up of line segments of form x=i or y=j, for integers i and j.

Moreover, according to [3] such embeddings can be constructed in linear time.

Thus, in constructing our reduction we may assume that we are given a graph $G = (V, E)$ that is embedded in the plane according to [16]. We construct in polynomial time a unit disk graph $G' = (V', E')$ with radius δ such that G has a dominating set S of size $\leq k$ if and only if G' has a d-hops dominating set S' of size $\leq k'$, where k' is determined from G and k.

Construction of the unit disk graph G':

Define $\delta = 1/(2d+1)$ unit as the radius of the unit disk graph G'. For each unit length in G we add $(2d+1)$ new intermediate vertices in equal distance. Thus, for each *original edge* (u, v) in G of length $l_{u,v}$, we add $(2d + 1) \times l_{u,v}$ *intermediate vertices*. Moreover, we add $(d - 1)$ *auxiliary vertices* $u_1, \ldots, u_{d-1}$ sequencially from *original vertex* u at each distance d as shown in Figure 1. Obviously the resulting graph $G' = (V', E')$ is a unit disk graph with radius δ, and G' can be constructed from G in polynomial time.

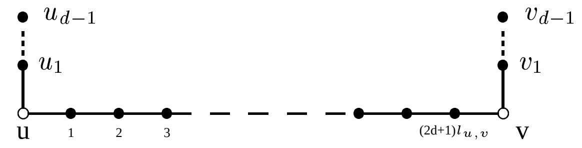


Fig. 1. **Construction of intermediate and auxiliary vertices**

Claim. G has a dominating set S of size

$$|S| \leq k$$

if and only if G' has d-hops dominating set S' of size

$$|S'| \leq k' := k + \sum_{\{u,v\} \in E} l_{u,v}$$

where $l_{u,v}$ is the length of the edge (u, v) in G.

Proof of Claim: For the only if direction suppose that G has a dominating set S of size m. We construct the d-hops dominating set S' in G' as follows. S' contains all vertices in S. Moreover, for each original edge (u, v) we add certain intermediate vertices to S' according to the following rules:

Rule 1: if u (or v) is in S, we add a total of $l_{u,v}$ intermediate vertices such that consecutive vertices are $(2d + 1)$ hops apart starting from $u(v)$.

Rule 2: if both u and v are in S, we add a total of $l_{u,v}$ intermediate vertices such that consecutive vertices are $(2d + 1)$ hops apart starting from u.

Rule 3: if both u and v are not in S, we add a total of $l_{u,v}$ intermediate vertices such that consecutive vertices are $(2d + 1)$ hops apart starting from position d.

An example of these rules is shown in Figure 2. Clearly, we have

$$|S'| \leq k + \sum_{(u,v) \in E} l_{u,v}$$

We now prove that the set S' is a d-hops dominating set. First observe that there is one intermediate vertex in S' for every $(2d + 1)$ consecutive intermediate vertices on any original edge,

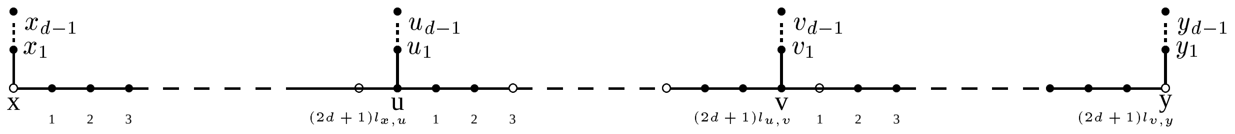


Fig. 2. **Vertices in d-hops dominating set**

so the intermediate vertices are either in S' or at most d hops from a vertex in S'. If an original vertex v is in S, then it is also in S' by Rule 1. Therefore its auxiliary vertices $v_1, \ldots, v_{d-1}$ is d hops away from v, a vertex in S'. Otherwise, if v is not in S, then there is a neighboring vertex u which is in S due to the fact that S is a dominating set. Consider the original edge (u, v). According to Rule 1, the intermediate vertex at position $(2d+1) \times l_{u,v}$ is in S'. Thus, vertex v and its auxiliary vertices $v_1, \ldots, v_{d-1}$ are at most d hops away from the vertex at position $(2d+1) \times lu, v$, which is in S'.

We have shown that any (original, auxiliary, or intermediate) vertex in G' is either in S' or at most d hops away from a vertex in S'. Hence S' is a d-hops dominating set in G'.

We now show the if direction. To this end, suppose that S' is a d-hops dominating set of size k' in G'. An *extended edge*(u', v') is an extension of an original edge (u, v) that includes all intermediate vertices as well as the auxiliary vertices $u_1, \ldots, u_{d-1}$ and $v_1, \ldots, v_{d-1}$. For the sake of convenience, the set of vertices in S' that are on an extended edge is denoted by $S'_{u,v}$. We construct the dominating set S for the graph G as as follows. For each extended edge (u', v') we remove vertices from S' according to the following rules:

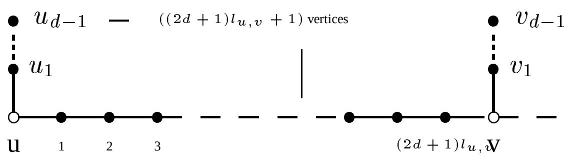
Rule 1: If only u (v) is in S' (See Figure 1):
Remove all vertices in $S'_{u,v}$ except u (v).
Rule 2: If both u and v are in S' (See Figure 2):
Remove all vertices in $S'_{u,v}$ except u and v.
Rule 3: None of u and v is in S':
If $|S'_{u,v}| \geq l_{u,v} + 1$, then add vertex u to S' and remove all vertices in $S'_{u,v}$. Otherwise remove all $S'_{u,v}$.

Observe that the number of intermediate vertices in S' from each original edge (u, v) is at least $l_{u,v}$ because we have a total of $(2d+1) \times l_{u,v}$ intermediate vertices on the edge. Moreover, when applying the above rules, the total number of vertices removed is at least $l_{u,v}$ for each extended edge (u', v'). Therefore the size of the resulting set S is

$$|S| \leq |S'| - \sum_{\{u,v\} \in E} l_{u,v} = k$$

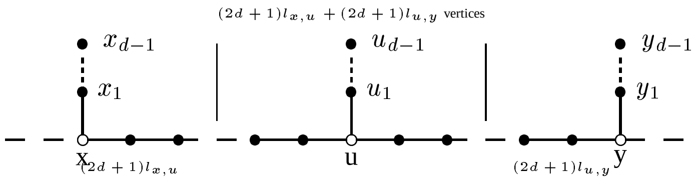
To verify that the set S is a dominating set in the original graph G, we just need to prove that every original vertex is either in S or adjacent with a vertex in S. To this end, consider any original vertex u which is not in S we have following cases:
Case 1: u is a degree 1 vertex with a neighbor v (See Figure 3). If v is in S', then v is also in S by Rule 1. Otherwise, on the extended edge (u', v'), there are $(2d+1) \times l_{u,v} + 1$ vertices which are at most d hops from a vertex in $S'_{u,v}$. Therefore $|S'_{u,v}| \geq l_{u,v} + 1$ and v is in S by Rule 3.
Case 2: u is a neighbor of degree 1 vertex v
Same reasoning as in Case 1.
Case 3: u is a neighbor of at least two degree 2 vertices x and y (See Figure 4).

If either x (or y) is in S', then x (or y)is also in S by Rule 1. Otherwise, on the extended edges (x', u') and (u', y'), there are $(2d+1) \times l_{x,u} + (2d+1) \times l_{y,u}$ vertices which are at most d hops away from a vertex in $S'_{x,u} \cup S'_{y,u}$. Due to the auxiliary vertices $u_1, \ldots, u_{d-1}$ we have $|S'_{x,u} \cup S'_{y,u}| \geq l_{x,u} + l_{y,u} + 1$. That leads to either $|S'_{x,u}| \geq l_{x,u} + 1$ or $|S'_{y,u}| \geq l_{y,u} + 1$. From Rule 3, we can see that either original vertex x or y is in S.
Thus S is a dominating set in G. This completes the proof of the theorem.



$|S'_{u,v}| \geq (l_{u,v} + 1)$ vertices

Fig. 3. u **is degree 1 vertex with a neighbor vertex** v



$(|S'_{x,u}| + |S'_{u,y}|) \geq (l_{x,u} + l_{u,y} + 1)$ vertices

Fig. 4. u **is degree 2 vertex with 2 neighbor degree 2 vertices** x **and** y

VI. Heuristic

A. Data Structures

The heuristic runs for $2d$ rounds of information exchange. Each node maintains two arrays, WINNER and SENDER, each of size $2d$ node ids: one id per round of information exchange.

The WINNER is the winning node id of a particular round and used to determine the clusterhead for a node, as described below in the $Basic\ Idea$.

The SENDER is the node that sent the winning node id for a particular round and is used to determine the shortest path back to the clusterhead, once the clusterhead is selected.

B. Basic Idea

Initially, each node sets its WINNER to be equal to its own node id. This is followed by the Floodmax phase.

Definition 2 (Floodmax) - Each node locally broadcasts its WINNER value to all of its 1-hop neighbors. After all neighboring nodes have been heard from, for a single round, the node chooses the largest value among its own WINNER value and the values received in the round as its new WINNER. This process continues for d rounds.

Definition 3 (Floodmin) - This follows Floodmax and also lasts d rounds. It is the same as Floodmax except a node chooses the smallest rather than the largest value as its new WINNER.

Definition 4 (Overtake) - As flooding occurs in the network, WINNER values are propagated to neighboring nodes. At the end of each flooding round a node decides to maintain its current WINNER value or change to a value that was received in the previous flood round. Overtaking is the act of a new value, different from the node's own id, being selected based on the outcome of the information exchange.

Definition 5 (Node Pairs) - A node pair is any node id that occurs at least once as a WINNER in both the 1^{st} (Floodmax) and 2^{nd} (Floodmin) d rounds of flooding for an individual node.

We simulate rounds of the flooding algorithm by having every node send and receive the equivalent of a synchronous round of messages. This is accomplished by requiring each node to send a round r message tagged with r as the round number. After a node has received round r messages from all its neighbors it may proceed with round r transition and ultimately to round $r+1$.

The heuristic has four logical stages: first the propagation of larger node ids via floodmax, second the propagation of smaller node ids via floodmin, third the determination of clusterheads, and fourth the linking of clusters.

The first stage uses d rounds of floodmax to propagate the largest node id in each node's d-neighborhood. At the conclusion of the floodmax the surviving node ids are the elected clusterheads in the network. Nodes record their winning node for each round. Floodmax is a greedy algorithm and may result in an unbalanced loading for the clusterheads. In fact, there may be cases where clusterhead B is disjoint from its cluster as a result of being overtaken by clusterhead A. Therefore, a node must realize not only if it is the largest in its d-neighborhood but also if it is the largest in any other node's d-neighborhood. This is similar to the strategy employed in [1]. The second stage uses d rounds of floodmin to propagate the smaller node ids that have not been overtaken. This allows the relatively smaller clusterheads the opportunity to (i) allow them to regain nodes within their d-neighborhood and, (ii) realize that they are the largest node in another node's d-neighborhood. Again each node records the winning node for each round.

At the conclusion of the floodmin, each node evaluates the round's WINNERs to best determine their clusterhead. In order to accommodate cases where a node's id is overtaken by another node id, the smallest node id appearing in both of the flooding stages is chosen as the clusterhead. The smaller clusterhead is chosen to provide load balancing. However, in the worst case where clusterhead A and clusterhead B are one hop away from one another; clusterhead B will record its own node id as a WINNER only in the final round of flooding. Therefore, if a node receives its own node id in the floodmin stage it knows that other nodes have elected it their clusterhead so it declares itself a clusterhead. Additionally, there may be scenarios where a node is overtaken in the floodmax stage by a set of nodes and then overtaken by a completely different set of nodes in the floodmin stage, none of which is its own node id. In this case the node has no other option but to select a clusterhead that is within d hops. The only known clusterhead that is within d hops is the WINNER of the final round of floodmax.

Finally, the *gateway* nodes (nodes at the periphery of a cluster) begin a convergecast message to link all nodes of the cluster to the clusterhead and, link the clusterhead to other clusters. Each gateway node will include its id and all other gateway nodes of other neighboring clusters in the message. This will establish the backbone of the network. During the convergecast it may be determined that a clusterhead resides on the path between a node and its selected clusterhead, as shown in Figure 5 with nodes 3, 16, 28, and 48 electing clusterhead 100. In this case the clusterhead closest to the node adopts it as a child. Figure 5 shows the clusters formed when the heuristic terminates.

The proposed heuristic provides an optimal solution when the largest node ids are spaced d distance apart. However, even when the largest node ids are located in close proximity the heuristic provides a good solution at low cost in time and messages.

C. Clusterhead Selection Criteria

The mechanics of the heuristic are quite simple. At some common epoch each node initiates $2d$ rounds of flooding. Each node maintains a logged entry of the results of each flooding round. The rounds are segmented into the 1^{st} d rounds and the 2^{nd} d rounds. The 1^{st} d rounds are a floodmax to propagate the largest node ids. After completion of the 1^{st} d rounds of flooding the 2^{nd} d rounds of flooding begin, using the values that exist at each node after the 1^{st} d rounds. The 2^{nd} d rounds of flooding are a floodmin to allow the smaller node ids to reclaim some of their territory. After completion of the 2^{nd} d rounds each node looks at its logged entries for the $2d$ rounds of flooding. The following rules explain the logical steps of the heuristic that each node runs on the logged entries.

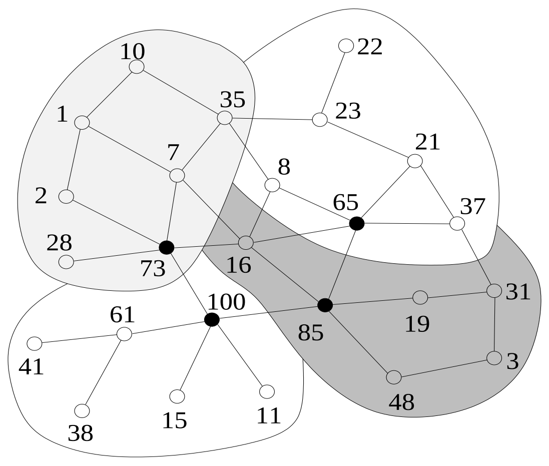
Rule 1: First, each node checks to see if it has received its own original node id in the 2^{nd} d rounds of flooding. If it has then it can declare itself a clusterhead and skip the rest of this phase of the heuristic. Otherwise proceed to Rule 2.

Rule 2: Each node looks for node pairs. Once a node has identified all node pairs, it selects the minimum node pair to be the clusterhead. If a node pair does not exist for a node then proceed to Rule 3.

Rule 3: Elect the maximum node id in the 1^{st} d rounds of flooding as the clusterhead for this node.

D. Gateway Selection and Convergecast

After a node has determined its clusterhead based on Rules 1, 2, or 3, it communicates that it is a member of the cluster to the clusterhead. In order to minimize messages this information is communicated from the fringes of the cluster, gateway nodes, inward to the clusterhead. Furthermore, a node has no way to know if it is a gateway node. Therefore, after clusterhead selection each node broadcasts its elected clusterhead to all of its neighbors. Only after hearing from all neighbors can a node determine if it is a gateway node. If all neighbors of a node have the same clusterhead selection then this node is not a gateway



Node	10	1	2	7	35	8	23	22	21	65	37	31	19	85	16	100	73	28	41	61	11	48	3	15	38
Max 1	35	10	73	73	35	65	35	23	65	85	65	37	85	100	85	100	100	73	61	100	100	85	48	100	61
Max 2	35	73	100	100	73	85	65	35	85	100	85	85	100	100	100	100	100	100	100	100	100	100	85	100	100
Max 3	73	100	100	100	100	100	85	65	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Min 1	73	73	100	100	73	100	65	65	85	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Min 2	73	73	73	73	65	73	65	65	65	85	85	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Min 3	65	73	73	65	65	65	65	65	65	65	65	85	100	85	73	100	73	100	100	100	100	100	100	100	100
Result	73	73	73	73	73	65	65	65	65	65	65	85	100	85	100	100	73	100	100	100	100	100	100	100	100

WINNER

Fig. 5. **3-cluster formation in a network of 25 nodes.**

node. However, if there are neighboring nodes with clusterhead selections that are different, then these nodes are gateway nodes.

Once a node has identified itself as a gateway node it then begins a convergecast to the clusterhead node sending its node id, all neighboring gateway nodes and their associated clusterheads. The SENDER data structure is used to determine who next to send the convergecast message. The process continues with each node adding its own node id such that when the clusterhead has heard from each of its immediate neighbors it has a database of every node in its cluster. It is not the intent of this heuristic to minimize the number of gateways.[1] Rather this heuristic maximizes the number of gateways resulting in a backbone with multiple paths between neighboring clusterheads. This provides fault tolerance, and eases congestion in the backbone network.

Rule 4: There are certain scenarios, as shown in Figure 5, where this heuristic will generate a clusterhead that is on the path between a node and its elected clusterhead. In this case, during the convergecast the first clusterhead to receive the convergecast will adopt the node as one of its children. The clusterhead will immediately send a message to the node identifying itself as the new clusterhead.

E. Correctness of Heuristic

The correctness of the heuristic is solely dependant on nodes electing clusterheads that actually become clusterheads. The following assumptions are used to first show that every node that survives the floodmax stage of the heuristic becomes a clusterhead. Then we will show that every node that is elected as a clusterhead does in fact become a clusterhead.

[1] Restricting the number of gateways minimizes the number of paths between clusterheads [12], [8].

Assumption 1: During the floodmin and floodmax algorithms no node's id will propagate farther than d-hops from the originating node itself (definition of flooding).

Assumption 2: All nodes that survive the floodmax elect themselves clusterheads.

Proof of Assumption 2: The floodmax will propagate the individual node ids outward creating a dominating set of node ids. This dominating set of node ids will consist of two classes. Class1 nodes will be those node ids that are the largest in their d-neighborhood. Class2 nodes will be those that are the largest in at least one of their d-hop neighbors' d-neighborhood. A Class2 node can not be a Class1 node.

Consider a Class1 node id, say node A. Node A will overtake each node that is d-hops away from it during the floodmax. Therefore, all nodes that are within the d-hop coverage area of node A will possess node A's id value in the WINNER data structure.

At the conclusion of the floodmin a Class1 node will elect itself a clusterhead, based on Assumption 1 and Rule 1. Consider a Class2 node id, say node B. Although node B is overtaken by larger node ids, its node id continues to propagate out and consume all smaller node ids within d-hops of node B. Therefore, at the completion of the floodmax node B's id and larger node ids (Class1 or Class2) will cover the d-hop coverage area of node B. Therefore, the Class2 id is the smallest surviving id in the d-hop neighborhood of the originating Class2 node.

Based on Assumption 1 we can conclude that the floodmin process will successfully propagate the Class2 node id back to the originating Class2 node. A Class2 node will elect itself a clusterhead, based on Rule 1.

Therefore, any node that survives the floodmax stage will elect itself a clusterhead.

Lemma 1: If node A elects node B as its clusterhead, then node B becomes a clusterhead.

Proof: The proof of the Lemma will consider all possible ways that a node may elect its clusterhead, and then prove that this node does in fact become a clusterhead.

Case 1: Node A elects itself as a clusterhead based on Rule 1. If node A receives its own id in the floodmin stage, it knows that other nodes have elected it a clusterhead based on Assumption 2. Therefore, it elects itself a clusterhead.

Case 2: Node A elects node B its clusterhead based on Rule 2. Node A receives an entry for B in the floodmin portion of the heuristic. Therefore, based on Assumption 2 we conclude that B does become a clusterhead. We choose the smaller node id pair to promote fairness and distribute the load among elected clusterheads.

Case 3: Node A elects node B it clusterhead based on Rule 3. Node A receives no node pairs and must select the only node known to be a clusterhead. The only node that is guaranteed to be a clusterhead is the last WINNER of the floodmax. This node survives the floodmax and again based on Assumption 2 it will become a clusterhead.

VII. Illustrative Examples

Figure 5 shows an example of the network topology generated by the heuristic with 25 nodes. Here we see four clusterheads elected in close proximity with one another, namely nodes 65, 73, 85, and 100. This figure shows how cluster division has the effect of drawing a line between clusterheads and splitting the nodes among themselves. Additionally, Figure 5 demonstrates the need for Rule 4, as nodes 3, 16, 28, and 48 have elected node 100 as their clusterhead but must pass through other clusters on their convergecast to node 100. On application of Rule 4, clusterhead 85 instructs nodes 3, 16, and 48 to join its cluster. While clusterhead 73 instructs node 28 to joins its cluster.

Figure 6 shows the resulting network topology after slightly perturbing the network in Figure 5. Here we see that three of the previous four clusterheads are re-elected. The fourth clusterhead, node 65 from Figure 5, is overtaken by clusterhead 85.

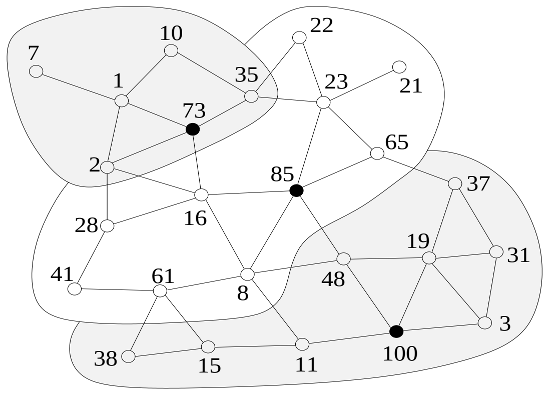


Fig. 6. **3-cluster formation after topology change.**

A. Pathological Case

There is a known configuration where the proposed heuristic fails to provide a good solution. This configuration is when node ids are monotonically increasing or decreasing in a straight line. In this case, the $d + 1$ smallest node ids belong to the same cluster as shown in Figure 7. All other nodes become clusterheads of themselves only. Again, while this is not optimal it still guarantees that no node is more than d hops away from a clusterhead. Furthermore, this configuration is highly unlikely in a real world application. However, this is a topic of future work to be performed with this heuristic.

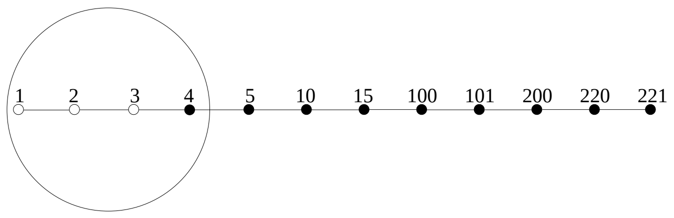


Fig. 7. **Worst case performance scenario for the proposed heuristic, d=3.**

B. Time, Message and Storage Complexity

Each node propagates node ids for $2d$ rounds to elect clusterheads. A convergecast is then initiated to inform the clusterhead of its children. Since no node is more than d hops from its clusterhead the convergecast will be $O(d)$ rounds of messages. Therefore, the time complexity of the heuristic is $O(2d + d)$ rounds = $O(d)$ rounds.

The time complexity and the number of transmissions required to achieve a local broadcast (to all neighbors) for a single round is dependent on the success of the data link layer protocol. While the MACA-BI has been shown to be superior to MACA and CSMA [6] it still suffers from the hidden terminal problem and may require re-transmissions to complete a round. A data link protocol similar to MACA-BI that resolves completely the hidden terminal problem is an area of additional research and not the intent of this paper.

Each node has to maintain $2d$ node ids in its WINNER data structure, and the same number of node ids in its SENDER data structure. Thus, the storage complexity is $O(d)$. This compares favorably with heuristics like [1], [2] where identities of all neighboring nodes is maintained and the storage complexity is $O(n)$.

VIII. Simulation Experiments and Results

We conducted simulation experiments to evaluate the performance of the proposed heuristic and compare these finding against three heuristics, the original Linked Cluster Algorithm (LCA) [1], the revised Linked Cluster Algorithm (LCA2) [5], and the Highest-Connectivity (Degree) [11], [8] heuristic. We assumed a variety of systems running with 100, 200, 400, and 600 nodes to simulate ad hoc networks with varying levels of node density. Two nodes are said to have a wireless link between them if they are within communication range of each other. The performance was simulated with the communication range of the nodes set to 20, 25 and 30 length units. Additionally, the span of a cluster, $i.e.$, the maximum number of wireless hops between a node and its clusterhead (d) was set to 2 and then 3 for each of the simulation combinations above. The entire simulation was conducted in a 200×200 unit region. Initially, each node was assigned a unique node id and x, y coordinates within the region. The nodes were then allowed to move at random in any direction at a speed of not greater than 1/2 the wireless range of a node per second. The simulation ran for 2000 seconds, and the network was *sampled* every 2 seconds. At each sample time the proposed Max-Min heuristic was run to determine clusterheads and their associated clusters. For every simulation run a number of statistics were measured for the entire 2000 seconds of simulation. Some of the more noteworthy simulation statistics measured were: *Number of Clusterheads, Clusterhead Duration, Cluster Sizes, and Cluster Member Duration*. These statistics provided a basis for evaluating the performance of the proposed heuristic.

Definition 6 (Number of Clusterheads) - The mean number of clusterheads in a network for a sample. We do not want too few clusterheads, as they will be overloaded with too many cluster members. Nor is it good to have a large number of clusterheads, each managing a very small cluster.

Definition 7 (Clusterhead Duration) - The mean time for which once a node is elected as a clusterhead, it stays as a clusterhead. This statistic is a measure of stability, the longer the duration the more stable the system.

Definition 8 (Cluster Sizes) - The mean size of a cluster. This value is inversely proportional to the *Number of Clusterheads*. We do not want clusters so large that they will overload their clusterheads, or so small that the clusterheads are idle a good part of the time.

Definition 9 (Cluster Member Duration) - The mean contiguous time a node stays a member of a cluster before moving to another cluster,[2] clusterheads are considered cluster members, also. This statistic is a measure of stability like the Clusterhead Duration, but from the point of view of nodes that are not clusterheads.

LCA, LCA2, and Degree based heuristics generate 1-hop clusters. Therefore, to properly compare these heuristics with the proposed Max-Min heuristic it was necessary to perform a d-closure on the connectivity topology before running each of these heuristics. The d-closure yields a modified graph in which nodes A and B are 1-hop neighbors if they were at most d-hops away in the actual topology graph. Here, d is either 2 or 3. When the LCA, LCA2, and Degree based heuristics are run on this modified graph, they form clusters where each node is at most d wireless hops away from its clusterhead. The LCA heuristic elects clusterheads that may be adjacent to one another while the LCA2 and Degree based heuristics do not allow clusterheads to be adjacent to one another. Therefore, the selection of these three heuristics should provide good coverage for benchmarking the performance of the proposed Max-Min heuristic.

Observing the simulation results of Figure 8 shows that Max-Min, LCA2, and Degree based heuristics never produce more than 33 clusterheads, when 2-hop clusters are formed and the wireless range is equal to 20 length units. Furthermore, as more nodes are added the number of clusterheads produced by these heuristics remains almost unchanged. The LCA heuristic produces a maximum of 130 clusterheads. Observing the LCA plot shows that the slope, approximately 0.17 for high density networks, will generate a clusterhead for every 5.8 newly added nodes. This is an unnecessarily large number of clusterheads. Similar trends are exhibited for other combinations of hop count and wireless range.

[2]A cluster is represented by the identity of its clusterhead.

Figure 9 shows Max-Min with the highest clusterhead duration followed by LCA2, LCA and then finally the Degree based heuristic. Max-Min shows an increase in clusterhead duration as the network becomes more dense, while for LCA, LCA2, and Degree the duration as the system size increases. This is not surprising for Degree as it is based on degree of connectivity, not node id. As the network topology changes this approach can result in high turnover of clusterheads [8]. Similarly, in LCA and LCA2 a single link make or break may move a lower id node within or out of d-hops of a node x, forcing it to transition between clusterhead and normal node states. Such transitions may also have a ripple effect throughout the network. This adversely impacts the stability of clusters.

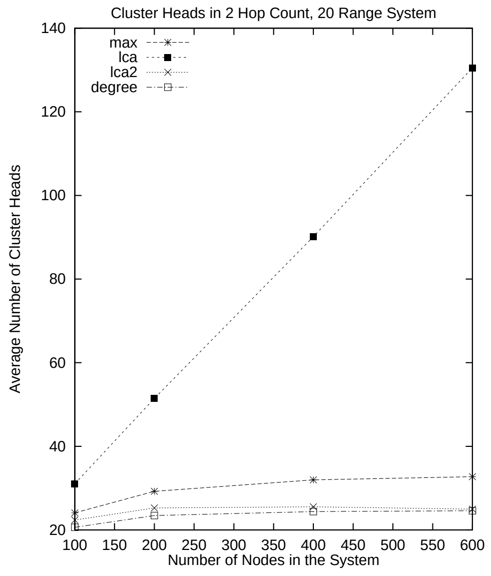


Fig. 8. **Impact of network density on number of clusterheads.**

Figure 10 shows the Degree based and LCA2 heuristics produce the largest cluster sizes followed by the Max-Min, and finally the LCA heuristics. The Degree, LCA2, and Max-Min heuristics produce clusters whose sizes increase by 3.1, 3.1 and 2.3 nodes per 100 nodes respectively. While the LCA heuristic cluster sizes are very flat and only increase slightly as the network density increases. Combining the number of clusterheads and number of cluster sizes results we can see that the LCA heuristic is producing a large number of small clusters as the system size gets larger. This indicates that the LCA heuristic very often suffers from a pathological case where a node becomes a clusterhead under somewhat false pretences. This can happen when a node becomes a clusterhead because it is the largest node in one of its neighbor's neighborhoods.

Figure 11 shows LCA2 and Max-Min with the highest cluster member durations followed by LCA and finally Degree. Here we see that the LCA2 heuristics show a slight increase in cluster member duration as the network becomes more dense, while the LCA heuristic shows a slight decrease. Max-Min has become

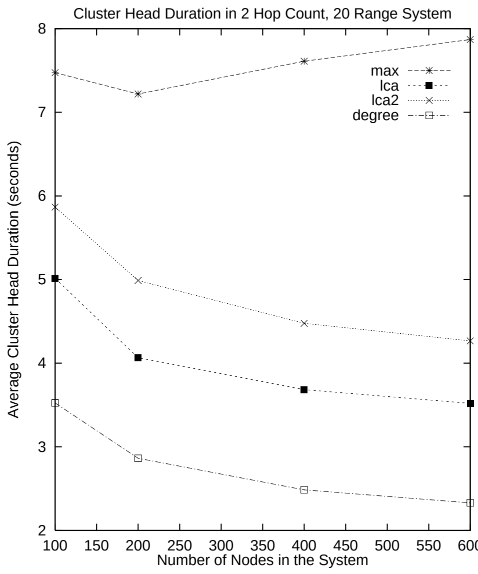


Fig. 9. **Impact of network density on clusterhead duration.**

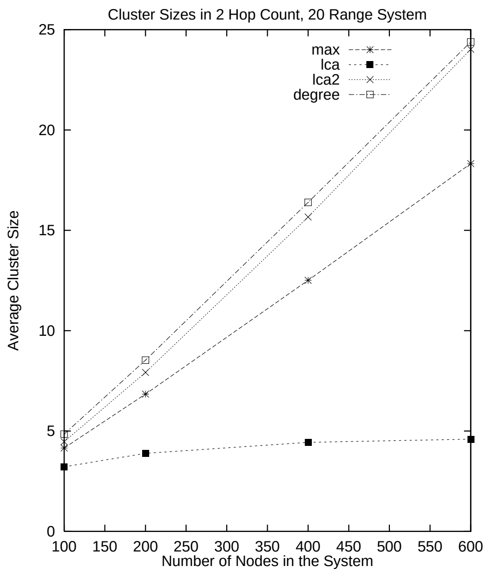


Fig. 10. **Impact of network density on cluster size.**

fairly flat at 3.7 seconds for dense networks, while the Degree heuristic show a steady decline to about 2 seconds, the sampling rate of the simulation.

Finally, Figure 12 shows that Max-Min produces the highest percentage of re-elected clusterheads (consistent with Figure 9). As a result Figure 13 shows that Max-Min elects only a fraction of the total number of nodes as leaders during the entire simulation run of 2000 seconds. This supports the idea that Max-Min will try to re-elect existing leaders. The LCA and Degree based heuristics elected every node or one short of every node as leader at least once during each simulation run of 2000 seconds. So, their plots are superimposed on each other and cannot be distinguished. While LCA2 does not elect every node a clusterhead in each simulation run, it still elects a much higher number of clusterheads than Max-Min. It is not desirable to change leadership too frequently as this causes the exchange of leadership

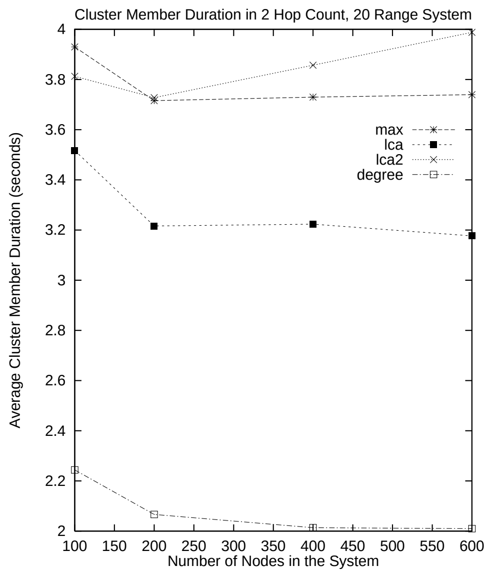


Fig. 11. **Impact of network density on cluster member duration.**

databases to the new clusterheads. This may ulimately cause congestion in the network.

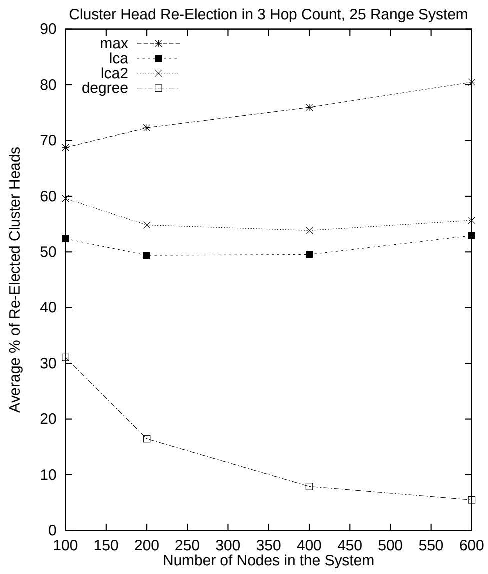


Fig. 12. **Impact of network density on re-elected clusterheads.**

The Max-Min heuristic produces fewer clusterheads, much larger clusters, and longer clusterhead duration on the average, than the LCA heuristic. While the Degree based heuristic does have slightly larger cluster sizes than the Max-Min, it suffers greatly in other categories such as clusterhead duration, and cluster member duration. The LCA2 heuristic produces clusterheads that are comparable in number to that of Max-Min. However, Max-Min has clusterhead durations that are approximately 100% *larger than that of LCA2 for dense networks. Furthermore, the Max-Min clusterhead duration continues to increase with increased network density, while the LCA2 heuristic clusterhead duration decreases with increased network density. Based on these initial simualtion results the Max-Min heuristic*

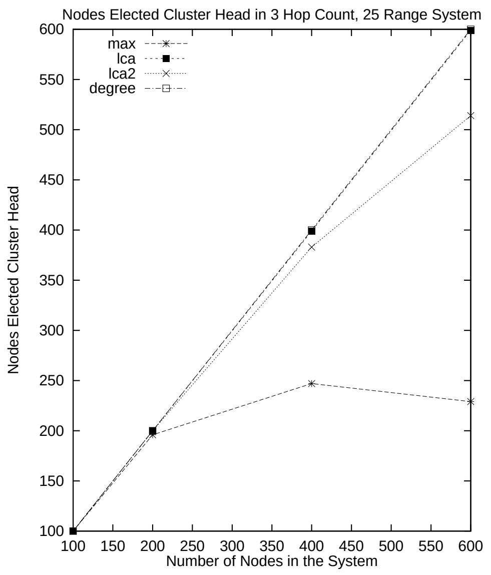


Fig. 13. **Impact of network density on number of nodes elected as clusterheads during the entire simulation.**

provides the best all around clusterhead leader election characteristics.

Future work is needed to determine the appropriate time to trigger the Max-Min heuristic. If periodic triggers are too closely spaced then the system may run the heuristic even when there has been no noticeable topology change to warrant running the heuristic. If the periodic triggers are too far apart then the topology may change without running the heuristic, causing nodes to be stranded without a clusterhead. The triggering condition should be completely asynchronous and localized to a node's cluster and its neighboring clusters to restrict execution of the heuristic to only affected nodes. Furthermore, the triggering scheme should account for topology changes during the progress of the heuristic. The Max-Min heuristic has a tendency to re-elect existing clusterheads. This is desireable for stability, however it must be tempered with a load-balancing scheme. Load-balancing allows re-election of existing clusterheads until they have exceeded their clusterhead duration budget. These clusterheads should then give way to allow other nodes to become clusterheads.

IX. Possible Applications of the Heuristic

Ad hoc networks are suitable for tactical missions, emergency response operations, electronic classroom networks, etc. A possible application for this heuristic is to use it in conjunction with Spatial TDMA. Spatial TDMA provides a very efficient communication protocol for clusters with few nodes. However, the nodes must be known and in a fixed location. Hence, Spatial TDMA is not easily used in ad hoc networks. The proposed heuristic may be used to determine the clusters and the clusterheads in the network. At this point all of the nodes within a cluster are known and assume to be fixed. This information may be used by Spatial TDMA to construct a TDMA frame for the individual clusters. Spatial TDMA will continue as the communication protocol until there is sufficient topology change that the proposed heuristic is run again to form new clusters.

The proposed heuristic can be used for hierarchical routing purposes wherein clusterheads can maintain routing information. It can also be used for location management purposes where the clusterheads receive location updates and queries from other nodes in the system.

X. Conclusion

A new heuristic for electing multiple leaders in an ad hoc networks has been presented, called Max-Min Leader Election in Ad Hoc Networks. Max-Min runs asynchronously eliminating the need and overhead of highly synchronized clocks. The maximum distance a node is from its clusterhead has been generalized to be d hops, allowing control and flexibility in the determination of the clusterhead density. Furthermore, the number of messages is a multiple of d rounds, providing a very good run time at the network level. Simple data structures have been used to minimize the local resources at each node. Re-election of clusterheads is promoted to minimize transferal of databases and to provide stability. The solution is scalable as it generates a small number of clusterheads compared to some other heuristics. Also, a low variance in cluster sizes leads to better load balancing among the clusterheads. Finally, this heuristic utilizes clusterheads and multiple gateway nodes to form a redundant backbone architecture to provide communication between clusters.

References

[1] D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, COM-29(11):1694–1701, November 1981.

[2] D.J. Baker, A. Ephremides, and J. A. Flynn. The Design and Simulation of a Mobile Radio Network with Distributed Control. *IEEE Journal on Selected Areas in Communications*, pages 226–237, 1984.

[3] T. Biedl and G. Kant. A Better Heuristic for Orthogonal Graph Drawing. In 2^{nd} *Annual European Symposium on Algorithms*, 1994.

[4] B. Das and V. Bharghavan. Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. In *Proceedings of ICC*, 1997.

[5] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling. *Proceedings of IEEE*, 75(1):56–73, 1987.

[6] M. Gerla F. Talucci and L. Fratta. MACA-BI (MACA By Invitation). Technical report, University of California at Los Angeles.

[7] E. Gafni and D. Bertsekas. Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, pages 11–18, January 1981.

[8] M. Gerla and J. T.-C. Tsai. Multicluster, mobile, multimedia radio network. *ACM Baltzer Journal of Wireless Networks*, 1(3):255–265, 1995.

[9] D. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer System Science*, 9:256–278, 1974.

[10] L. Kleinrock and J. Silvester. Spatial Reuse in Multihop Packet Radio Networks. *Proceedings of the IEEE*, 75(1):156–167, January 1987.

[11] Abhay K. Parekh. Selecting Routers in Ad-Hoc Wireless Networks. In *ITS*, 1994.

[12] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE INFOCOM*, April 1997.

[13] C.E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of ACM SIGCOMM Conference on Communication Architectures, Protocols and Applications*, pages 234–244, August 1994.

[14] A. Tanenbaum. *Computer Networks(3^{rd} Edition)*. Prentice Hall, Upper Saddle River, N.J., 1996.

[15] Jennifer Lundelius and Nancy Lynch. An Upper and Lower Bound for Clock Synchronization. *Information and Control*, Vol. 62 1984.

[16] L. Valiant. Universality Considerations in VLSI Circuits. *IEEE Transactions on Computers*, 1981.