Energy Efficient Connected Clusters for Mobile Ad Hoc Networks

Sayan Mitra* and Jesse Rabek**
{mitras,jesrab}@csail.mit.edu
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar Street,
Cambridge, MA 02139, USA

***Abstract*— A Mobile Ad Hoc Network (MANET) is a wireless infrastuctureless network with mobile nodes. Clustering is a common basis for building higher level applications for such networks. The merit of a clustered decomposition depends on the application that is meant to use it. A power control based distributed clustering service is proposed that maintains cluster connectivity under reasonable assumptions. The size and sparsity of the clustering can be controlled by two parameters, namely, the minimal separation between the clusterheads, and the maximum angular gap between neighboring clusterheads. The optimal value of the latter is derived; this minimizes the transmission power of the clusterheads while guaranteeing connectivity of the cluster graph. Experimental studies presented show that the algorithm rapidly stabilizes to a new clustered organization after the network topology changes due to node joins and failures.**

***Index Terms*— Ad hoc networks; Power control; Clustering algorithms; Cluster graph connectivity.**

I. Introduction

In mobile ad hoc networks (MANETs) clustering is a common basis for building higher level applications like routing, tracking, and location management. Underlying every MANET application there are inherent trade-offs between accuracy, energy consumption, robustness, and memory requirements [22]. Accordingly, the merit of a clustered decomposition depends on the application which uses it. In a routing protocol, for example, where each clusterhead maintains complete route to all cluster members, smaller clusters imply less state maintained by the clusterhead and therefore are preferable over larger clusters for scalability. Although small clusters result in a high latency between nodes that are far apart in the network, this delay in message delivery is tolerated. Accordingly, the clustering schemes typically used for routing decompose the network into 2-clusters. In contrast, for reference broadcast based clock synchronization (RBS) [6] large, densely overlapping clusters are preferable. In RBS, each cluster behaves as a synchronized unit and timing information is shared between nodes belonging to different clusters through common "gateway" nodes constituting a time-routing path. Shorter the path, more accurate the synchronization between clusters. So, it is desirable to have few large clusters spanning the entire graph. In general, larger, heavily overlapping clusters improve the robustness, accuracy, and latency of the application using the clusters, while adversely affecting the power consumption, memory requirements, and the longevity of the mobile nodes.

Irrespective of the size of the clusters, most applications require the clusters to be connected. This can be achieved in an ad hoc fashion, by arbitrarily increasing the broadcast radius of the clusterheads. This is undesirable for two reasons: first, it shortens the battery life of the clusterheads because the power required to transmit a message over distance d increases as an n^{th} degree polynomial of d, where $n \geq 2$ [20], and secondly, it gives rise to extra interference.

In this paper, we propose a robust distributed clustering service which can produce the desired type of clustering of a network for a wide range of MANET applications. The size and the sparsity of the clustered decomposition are controlled by two parameters, namely, s - the minimal separation between clusterheads, and a - the maximum allowed *angular gap* between neighboring clusterheads. Further, the algorithm minimizes the broadcast power of the clusterheads while guaranteeing the connectivity of the cluster graph. We prove that for any value of s, the optimal value of a is $2\sin^{-1}\frac{\sqrt{15}}{8}$ (1.0107 radians approx.), in following sense: this value of a

*The first author's research is supported by AFRL contract number F33615-010C-1850.

**The second author was an M.Engg student at the Laboratory for Computer Science, MIT, at he time of this work.

minimizes the transmission power of the clusterheads while guaranteeing connectivity of the cluster graph, provided the given distribution of the mobile nodes can be connected when each node transmits with maximum power.

The clustering service is implemented in two layers. The bottom layer selects the clusterheads based on a maximal independent set algorithm, such that each clusterhead is at least s distance away from all other clusterheads. The top layer decides the transmission power used by the clusterheads, based on certain locally checkable condition. Specifically, each clusterhead increases its transmission power until it *learns* about another clusterhead (via some common node) in every a-cone around itself. We discuss the different clustered decompositions that can be obtained by running the algorithm with different (s, a) settings. We also present experimental evidence supporting the robustness of the algorithm when subjected to changes in the underlying network topology owing to node failures and joins.

The rest of the paper is organized as follows: The next section cites and differentiates related work. In Section III the system model is described. In Section IV the details of the algorithms constituting the clustering service are presented. In Section V the optimal value of the parameter a, the angular gap, is derived and certain other optimizations to the basic clustering service are suggested. In Section VI the simulation results evaluating the behavior of the algorithm with different parameter settings are presented, with respect to stability, robustness, and the generated cluster topologies. Finally, Section VII concludes the paper with a synopsis of the contributions and directions for future research.

II. Related Work

The notion of cluster organization has been used for ad hoc networks ever since their appearance. In [2], [7] a distributed clustered architecture is introduced for hierarchical routing. Gerla et. al. [9], [16] have presented clustering algorithms for efficient resource allocation, namely bandwidth and channel, in order to support multimedia traffic. Most clustering algorithms produce a 2-clustering of the network graph. The generalization of this problem to k-clustering was introduced in [1], and used for routing in [13]. It is known that k-clustering is NP-complete [5] for simple undirected graphs. Fernandess and Malkhi [8] have given a polynomial time approximation algorithm for k-clustering with $O(k)$ worst case ratio over the optimal solution.

Most clustering algorithms including ours, work in two phases. First, a subset of nodes in the network are selected to act as coordinators or the clusterheads. The criterion for selecting the clusterheads differ between algorithms, the most common ones being based on the node identifiers [2], [7], [9] and node degrees [17]. In [3] a node mobility based criterion has been used for selecting the clusterheads to cope with dynamic changes in the network topology. The second phase, which is typically initiated by the clusterheads, is concerned with maintaining the extent of the clusters. A clusterhead may inform its member nodes about their membership explicitly, by sending a message, or depending on the application, may just store the membership information in its own state.

Instead of sending messages expressly for imparting the membership information to each member node, the clusterhead can periodically broadcast a beacon with a particular transmission power to mark its extent [14]. It has been shown in [10] that for a uniform distribution of a large number of wireless nodes, a common transmit power level is optimal with respect to the capacity of the network. In general, however this is not true, and hence the need for clustering by power control arises. In [12] two routing algorithms are proposed which decide the transmit power for each packet, thereby implicitly creating transmission power based clusters. Our clustering service is closer to the algorithm presented in [14], in that, we use power control to explicitly maintain the extent of the clusters.

Varying the transmission power of nodes for efficient topology control in wireless networks was studied in [19]. Li et al. [15], [21] describe a cone based topology control (CBTC) algorithm using directional sensors which guarantees global connectedness. The CBTC(a) algorithm increases the transmission power of a node v until there is a node within its transmission radius in every cone of angle a around it. It has been shown in [21], [15] that for $a \leq 2\pi/3$ the power settings obtained from CBTC(a) are sufficient for maintaining connectivity. Hajiaghayi et. al. [11] presented a modified version of the CBTC protocol with $a \leq \frac{2\pi}{3k}$, to ensure k-connectivity of the network. Our algorithm for determining the transmission power of each clusterhead is similar to CBTC but it maintains connectivity between clusters rather than individual vertices.

III. Preliminaries

In this section we discuss our model and introduce the notations and definitions used throughout the rest of this paper.

The mobile nodes are assumed to be distributed in a 2-dimensional plane and each node has an unique identifier. Each node can broadcast messages at different transmission power levels; the maximum power P being the same for all nodes. The nodes do not possess knowledge about their location in the plane, however they do have directional antennas, and a common sense of direction. Having directional antennas is considered to be a reasonable assumption and have been used elsewhere in the ad hoc networking literature (see, e.g., [15], [21], [4]). The common sense of direction can be achieved by means of a compass.

The mobile nodes can fail or migrate, and new nodes can join the network. When a node fails it loses its state. If a failed node recovers, it behaves as if it were a new node and tries to join the network.

At a given instant of time the mobile ad hoc network is represented as a directed graph $G = (V, E)$, where V is the set of mobile nodes and E the set of edges determined by the transmission power of the nodes. The transmission power p_v of node v determines the transmission radius, $d(p_v)$, and if v is a cluster head then $d(p_v)$ also defines the cluster around v, which is the set of nodes $C_v^k = \{x \mid dist(v, x) \leq d(p_v)\}$.

A clustering algorithm organizes a network of nodes into a set of clusters $\mathcal{C} = \{C_1, \ldots, C_m\}$. The cluster cover $\mathcal{C}$ is characterized by its *radius* or *size*, and its *sparsity*. The radius of $\mathcal{C}$ is the radius of its largest cluster in terms of physical distance, or the number of network hops. The size is the cardinality of the largest cluster in $\mathcal{C}$. The sparsity of $\mathcal{C}$ is a measure of connectivity between the clusters. It is the maximum number of clusters to which any particular node belongs. If $\mathcal{C}$ is a partition, that is, if all the clusters are disjoint, then sparsity is the maximum number of neighboring clusters of any cluster.

Definition 1. *The* **cluster graph** *induced by a set of clusters $\mathcal{C} = \{C_1, \ldots, C_m\}$ is the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \mathcal{C}$ and $\mathcal{E} = \{(C_u, C_v) \mid C_u \cap C_v \neq \emptyset\}$.*

Definition 2. *A* **path** *in a cluster graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of clusters $(C_1, \ldots, C_l)$, such that each $C_i \in \mathcal{V}$ and $(C_i, C_{i+1}) \in \mathcal{E}$ for every $i < m$. The cluster graph $\mathcal{G}$ is* **connected** *if there is a path between every pair of cluster in $\mathcal{V}$.*

IV. The Algorithms

The clustering service is implemented in two layers. The first layer, namely the *Start Cluster* algorithm (SC), maintains a set of cluster heads W, no two of which are closer than a distance s, where s is a parameter of the algorithm. The second layer, namely the *Cluster Control* (CC) algorithm, determines the broadcast power of the clusterheads thus defining the extent of each cluster. The broadcast power is determined by monitoring local membership and the overlap with neighboring clusters. This ensures that the set of clusters cover the entire network and that the resulting cluster graph is connected.

A. Start Cluster Algorithm

The Start Cluster algorithm (SC) is a dynamic version of the classical maximal independent set (MIS) algorithm [18]. The nodes selected to be in the set are the clusterheads and all other nodes are the *ordinary* nodes.

Definition 3. *The* **s-neighborhood** $\Gamma^s(v)$ *of node v is the set of nodes within distance s from v.*

Definition 4. *An* **s-independent set** *of a graph G with vertices V is a set of vertices $W \subseteq V$, no two of which are within a distance of S of each other. An s-independent set is* **maximal** *if no vertex can be added to it without violating s-independence property.*

In the classical MIS algorithm a node decides to be a clusterhead once and for all when it learns that all the neighboring nodes with higher identifier have decided not to be clusterheads. And, a node decides to be an ordinary node when it learns about a neighbor who has decided to be a clusterhead. Our reactive version of the MIS algorithm makes decisions for a finite interval of time, and so it has to be executed repeatedly by every participating node. This makes it possible for a node to take the necessary actions when there are significant changes in its neighborhood. Once a node decides to be a clusterhead, it remains a clusterhead for an interval T_b, after which it continues to be a clusterhead only if all the nodes with higher identifiers in its s-neighborhood are ordinary. A node decides to remain ordinary for as long as it is aware of a clusterhead in its S-neighborhood, and renews its bid to become a clusterhead only when it stops hearing from the latter.

Every participating node executes the SC algorithm shown in Figure 1. The `main` subroutine decides if the particular node is going to be a clusterhead or not, and the `messages_thread` handles the input message queue. Every decision message received by a node is stamped with an expiration time (TTL) measured by the local clock now of the node. When node v receives a `decided(0)` message from node u in its s-neighborhood, it adds id_u in an array K_v with a TTL T_b time after the time of reception of the message. Node v decides to become/remain a clusterhead at time now_v if it has received a `decided(0)` message from every node in its s-neighborhood within the last T_b interval of time. The algorithm is said to have *stabilized* once all the nodes in the network G have decided. Once the network stabilizes, and there are no further joins, failures, or migrations, the set of nodes W with $h_v = 1$ constitute a maximal s-independent set of the network. In the worst case, stabilization may take $O(diam(G))$ time.

```
message_thread:
On receiving (decided(1), id_u)
    if id_v > id_u then
        h_v ← 0 ; bcast(decided(0), id_v) with r_v = S.
On receiving (decided(0), id_u)
        K_v ← K_v ∪ {(id_u, now_v + T_b)}

main:
initially h_b ← ⊥; every T_b time
If for every w ∈ Γ^S(v) ∧ id_w > id_v
    ∃ (w, t) ∈ K_v, t > now_v then h_v ← 1 ;
        bcast(decided(1), id_u) with r_v = S.
```

Fig. 1. The Start Cluster algorithm. The `decided(1)` message is sent when a node decides to become a clusterhead, while `decided(0)` is sent when a node decides otherwise.

B. Stability of SC

If the network topology changes owing to failure, migration, or joining of nodes, the algorithm regains stability, in $O(diam(G))$ time, in the worst case. We briefly discuss the performance of the algorithm in each of these cases.

Joins: When a new node v joins the network G, there are two possible scenarios that may arise. First, if v is within the s-neighborhood of some other vertex $u \in W$, then it receives the `decided(1)` message from u and immediately decides to be an ordinary node. Otherwise, v is outside the s-neighborhood of every node in W and it would decide to be a clusterhead within T_b time. Therefore, in either case the algorithm regains stability within $O(1)$ time.

Failures: When node $v \in V$ fails, again there are two possibilities. Let the network after the failure be denoted G'. If $v \notin W$, that is if v is not a clusterhead then its failure does not affect the stability of the algorithm. Otherwise, $v \in W$ then one or more of the nodes in $\Gamma^S(v)$ would become eligible to be a clusterhead. This change may propagate across the network, and in the worst cast a new stable configuration would be achieved in $O(diam(G'))$ time.

Migrations: When an ordinary node $v \notin W$ changes its physical location it can be viewed as a combination of the above two cases. That is, it fails in its existing cluster and joins the cluster in its new location (or becomes a new clusterhead). On the other hand, if a clusterhead $v \in W$ moves into $\Gamma^S(u)$ for some other clusterhead $u \in W$, then the situation is indistinguishable from the case where u migrates into $v's$ cluster. We break the symmetry using some heuristics, e.g., the clusterhead with the larger cluster and the lower identifier remains a clusterhead and the other node becomes ordinary. If $v \in W$ then its movement would be indistinguishable from its failure to the nodes in $\Gamma^S(v)$ and this may trigger a complete reorganization as discussed above.

Thus, under all possible scenarios the SC algorithm regains stability within $O(diam(G))$ time. If the rate of failures, migrations, etc., in the network is not too large then the set W remains stable most of the time. The Cluster control algorithm presented in the next section relies on the stability of W to produce a robust clustered decomposition of the network.

C. Control Cluster Algorithm

The Control Cluster algorithm (CC) determines the minimum transmission power for each clusterheads such that the cluster graph is connected. The idea behind CC is similar to the cone based topology control algorithm (CBTC) of [15]. The algorithm is parameterized by a *cone angle* a which can be tuned to change the sparsity of the cluster graph. The cone angle a defines the gap_a predicate which is the key property that enables the clusterheads to guarantee global connectivity.

Definition 5. *For a given angle a, a clusterhead v is said to satisfy the gap_a condition if there exists an a-cone at v in which there is no cluster graph edge from v to any other clusterhead (see Figure 2).*

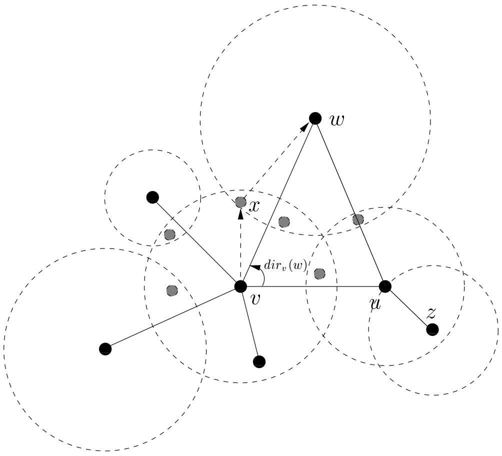

Fig. 2. The dark vertices are clusterheads and the light ones are ordinary nodes. There is a cluster graph edge between the clusters C_v and C_w through node x, and the direction of the edge $dir_v(w)$ is known to v. There is no edge between C_v and C_z because there is no common node. Clusterhead v satisfies $gap_{\frac{\pi}{3}}$ but clusterhead u does not.

The CC algorithm proceeds in rounds; starting from some initial value p_0, in each round the transmission power is increased in steps, according to some function Inc, until the maximum power P is reached or the gap_a predicate is violated. An exception to this stepwise power increment rule is made when an ordinary node u requests to join the cluster. In which case, the clusterhead increases its transmission power in a single step to include u in its cluster.

Before presenting the algorithm more formally, we discuss how the gap_a predicate can be efficiently checked by a clusterhead using only local information. As Figure 2 shows, there exists an edge between two clusterheads v and w, if and only if some node x is located in the intersection of their clusters $C_v \cap C_w$. Since the nodes possess a common sense of direction, assuming that the nodes can also infer their distances from signal attenuation, a clusterhead v can derive the direction of the common edge $dir_v(w)$ to w, using $dir_v(x)$ and $dir_x(w)$, if the latter direction is informed by x. Once v learns about all its common edges, the gap_a condition can be checked by maintaining a sorted list D_v of all edges; if there exists two consecutive elements in D_v whose difference is greater than a, then v satisfies gap_a.

The CC algorithm (see Figure 3) has a `main` subroutine that controls the periodic broadcasts and manages the internal data structures, and a `message_thread` that handles the incoming messages. Each node u in the network maintains a set M_u containing an element (v, d) for every cluster it belongs to. The first component v is the unique identifier of the clusterhead, and the second component $d = dir_u(v)$, that is, the direction of v with respect to u. In addition to the set M_v and the list D_v mentioned above, a clusterhead v also stores the highest value of req_u received through a `JoinReq` message (described below), in a variable called J_v.

`Message_thread:`
On receiving $(\texttt{Hello}, id_u)$
  $M_v \leftarrow M_v \cup \{(id_u, dir_v(u))\}$
  send$(\texttt{Ack}, M_v, id_v)$ with power ack_v
On receiving $(\texttt{Ack}, id_u, M_u)$
if $h_b = 1$ **then**
  $D_v \leftarrow D_v + \{dir_v(x) \mid (id_x, dir_u(x)) \in M_u\}$
    `% sorted list`
  $gap_a(D_v) \leftarrow \exists d_i \in D_v, |d_i - d_{i+1}| \geq a$
On receiving $(\texttt{joinReq}, req_u)$
if $h_b = 1$ **then**
  $J_v \leftarrow max(J_v, req_u)$

`main:`
if $h_b = 0$ **then** `% ordinary node`
  Wait for T_b
  if $M_v = \emptyset$ **then**
    bcast $(\texttt{JoinReq}, req_v)$ with power req_v
  $req_v \leftarrow min(P, Inc(req_v))$
if $h_b = 1$ **then** `% clusterhead`
  if $gap_a(D_v) \wedge p_v < P$ **then**
    $p_v \leftarrow max(Inc(p_v), J_v)$
    $J_v \leftarrow 0$

Fig. 3. Cluster control procedure. For ordinary nodes $h_b = 0$ and for clusterheads $h_b = 1$.

In each round a clusterhead v broadcasts a `(Hello,` id_v`)` message with its current power p_v and waits for Acks. Upon receiving an `(Ack,` id_u`,` M_u`)` from node u, a clusterhead v transforms the directions in M_u to its own coordinate system and adds them to the list D_v in sorted order. The clusterhead checks the gap_a condition by computing differences between consecutive members of D_v. If the gap_a condition is satisfied and the transmission power is less than the maximum power P, then the tentative transmission power for the next round is increased according to some function Inc. The actual power used for broadcast is the maximum of this tentative power and J_v.

An ordinary node v upon receiving a `(Hello,` id_u`)` message adds $(id_u, dir_v(u))$ to its membership set M_v and transmits an `(Ack,` id_v, M_v`)` message with power ack_v. The power ack_v is locally determined by v such that it is sufficient to either reach some intermediate node that can forward the message to u, or it reaches u directly. If it times out before receiving a `Hello` message then it sends a `(JoinReq,` req_v`)` message with power req_v to the nearest known clusterhead. If there is a clusterhead v' within s distance from v, then v sends the `JoinReq` to v', otherwise v performs a discovery process by broadcasting `JoinReq` with increasing power until it receives a `Hello` message.

For small values of a, the gap_a predicate is easier to satisfy, and the broadcast power of a clusterhead has to be high in order to discover nodes which are farther away and learn about other clusterheads. On the other hand, with large values of a the gap_a condition might be violated with a small broadcast radius, but this may result in disconnected clusters. Therefore, we have to find the maximum value of a which ensures connectivity of the cluster graph. In the next section we derive this value of a for a static network.

V. Optimal Cone Angle for Power Efficiency and Connectivity

Given a network $G = (V, E)$ and a set $W \subseteq V$ of clusterheads, let $\mathcal{G}_W^R$ and $\mathcal{G}_W^a$ be the induced cluster graphs corresponding to every $w \in W$ transmitting with maximum power P and the power assigned by CC(a) algorithm, respectively. We shall prove that for $a \leq a_{max} = 2\sin^{-1}\frac{\sqrt{15}}{8}$, the cluster graph $\mathcal{G}_W^a$ is guaranteed to be connected if $\mathcal{G}_W^R$ is connected. The value a_{max} corresponds to the angle at the center of a circle with radius R subtended by the intercepts of an arc of radius $2R$ drawn with the same center with another circle of radius R touching the first circle (see Figure 4). For proving the above statement we make use of the following geometric Lemma:

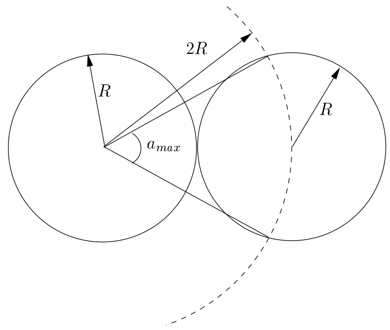

Fig. 4. Geometric interpretation of the maximal cone angle $a = 2\sin^{-1}\frac{\sqrt{15}}{8}$: the angle subtended at the center of a circle from the intercepts of an adjacent circle with the same radius and a concentric circle with double the radius.

Lemma 1. *Consider a point x on the line segment $\overline{uv}$ such that $|\overline{ux}| \leq R$ and $|\overline{vx}| \leq R$, and any point w on the line $\overline{uz}$ making $\angle zuv \leq \sin^{-1}\frac{\sqrt{15}}{8}$. If $0 < |\overline{uw}| < |\overline{uv}|$, then $min(|\overline{wx}|, |\overline{wv}|) \leq R$.*

Proof. See the appendix.

Theorem 1. *If $a \leq 2\sin^{-1}\frac{\sqrt{15}}{8}$ then $\mathcal{G}_W^a$ preserves connectivity of $\mathcal{G}_W^R$; for $u, v \in W$, C_u^a and C_v^a are connected iff C_u^R and C_v^R are connected.*

Proof. Given the same set of clusterheads W, the induced cluster graph $\mathcal{G}_W^a$ is a subgraph of the cluster graph $\mathcal{G}_W^R$, therefore it is clear that if C_u^a, C_v^a are connected then C_u^R, C_v^R must also be connected. We prove the converse as follows.

Suppose cluster graph G_W^R is connected while G_W^a is not. So, there exists at least one pair of clusters, such that there is no path between them in $\mathcal{G}_W^a$. We select one such pair C_u^a, C_v^a, for which the distance between their clusterheads, $dist(u, v)$ is the smallest. Since $\mathcal{G}_W^R$ is connected we know that $dist(u, v) \leq 2R$ and that there exists a vertex $x \in C_u^R \cap C_v^R$. Since $x \notin C_u^a \cap C_v^a$, it follows that the radius of the clusters C_u^a and C_v^a cannot both be equal to R. At least one of the radii is less than R, let us assume without loss of generality that $r_u < R$, therefore $r_v = R$. Then, u must have terminated the $CC(a)$ with $gap_a(D_u) = \mathsf{false}$, that is, there exists an edge between u and some other clusterhead within the a cone bisected by $\overline{uv}$ (as shown in Figure 5). Let w be such a cluster head that makes the angle γ minimum. Therefore, $\gamma \leq \sin^{-1}\frac{\sqrt{15}}{8}$, and $\overline{wu} \leq r_u + r_w \leq r_u + R < \overline{uv}$. From Lemma 1 we know that either $x \in C_w^R$ or $v \in C_w^R$. In other words, C_w^R and C_v^R are connected in $\mathcal{G}_W^R$.

Since $0 < \overline{uw} < \overline{uv}$, and $\gamma < \frac{\pi}{3}$, it follows that $\overline{wv} < \overline{uv}$. By our assumption C_w^a and C_v^a are not connected. So we have a pair of clusters C_u^R, C_w^R which are connected in $\mathcal{G}_W^R$ but not in $\mathcal{G}_W^a$, with $dist(u,w) < dist(u,v)$. This contradicts our assumption that u, v were the closest such pair of disconnected clusterheads. □

To show that the above value $a = 2\sin^{-1}\frac{\sqrt{15}}{8}$ is in fact the maximum possible value ensuring connectivity, we construct a simple counterexample. Consider the scenario where $a+2\epsilon$ is used as the cone angle for the *gap* condition (Figure 5). The clusters C_u^R and C_v^R are connected through vertex x. Owing to the presence of the node p within $R-\delta$ distance of u, there is an edge between $C_u^{a+2\epsilon}$ and $C_w^{a+2\epsilon}$ in the upper half $\gamma+\epsilon$-cone. Similarly, there is an edge between $C_u^{a+2\epsilon}$ and $C_y^{a+2\epsilon}$. Clusterhead u, therefore, violates the $gap_{a+2\epsilon}$ condition and terminates the cluster control algorithm with a $d(p_v) = R - \delta$. As a result $x \notin C_u^{a+2\epsilon}$, and $C_v^{a+2\epsilon}$ is not connected to the rest of $\mathcal{G}^{a+2\epsilon}$.

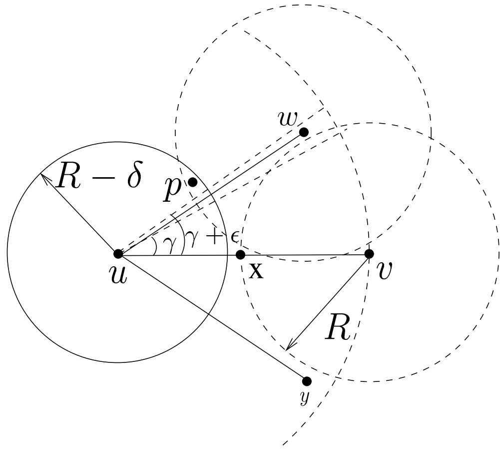

Fig. 5. Clusterhead uses a $gap_{a+2\epsilon}$ criterion which makes its radius $R-\delta$, disconnecting cluster C_v.

A. Optimizations

It is possible that the power assigned by the CC algorithm described above is too high for a clusterhead after new nodes join in its neighborhood. In this section we present some optimizations to the basic CC algorithm to improve its efficiency under such dynamic conditions. Essentially, a clusterhead v should shrink its broadcast radius from a high value p'_v to a lower value p_v when the member nodes at a distance farther than $d(p_v)$ do not contribute to D_v. The `Shrink_back` procedure, shown in Figure 6 is invoked by the cluster control algorithm to reduce the transmission radius.

`Shrink_back(c):`

$B_v \leftarrow \{u \mid u \in C_v \wedge (M_u = \{id_v\})\}$
$B'_v \leftarrow C_v \setminus B_v$
$d_v \leftarrow max_{u \in B_v} dist(v,u)$
$d'_v \leftarrow max_{u \in B'_v} dist(v,u)$
$D'_v \leftarrow \{dir_v(x) \mid (id_x, dir_u(x)) \in M_u \wedge dist(u,v) \leq d'_v - c\}$
if $(gap_a(D_v) \vee \neg gap_a(D'_v) \wedge (d'_v - c > d_v)$
then $d(p_v) \leftarrow d'_v - c$

Fig. 6. Shrink back optimization. c is a parameter of the procedure which determines how aggressively the broadcast radius is cut down.

Clusterhead v keeps record of the distance to the farthest member node which actually contributes towards satisfying the *gap* predicate. The sets B_v and B'_v are complementary subsets of C^v: B_v is the set of nodes which *do not* belong to any other cluster; d_v and d'_v are the corresponding distances to the farthest node. Clearly, v can not shrink the cluster radius below d_v, because that would make some of the nodes in B_v cluster-less. The set D'_v is the set of directions in which v has edges via nodes which are closer than $d'_v - c$. The the broadcast radius is reduced to a $d'_v - c$, if this does not create new gaps, and if the new radius does not exclude any of the nodes in B_v.

VI. Simulation

In this section we study the performance of the clustering service through simulation based experiments. We observe the cluster graph topologies generated by setting different values of the parameters s and a and the robustness of the algorithm under dynamic changes in the network topology due to node joins and failures.

Our discrete event simulator, implemented in C++, emulates individual mobile nodes with asynchronous communication. For the results presented in this section we have used a set of 100 nodes placed in a 2-dimensional plane of 300x300 square units. The nodes are distributed randomly in the plane such that each node lies within the maximum broadcast distance of at least one other node; this guarantees that the underlying network is connected when all the nodes broadcast with maximum transmission power.

A. Cluster Graph Topology

One of the main advantages of our clustering service is that, one can control the type of clustered decomposition of the network by appropriately setting the s and

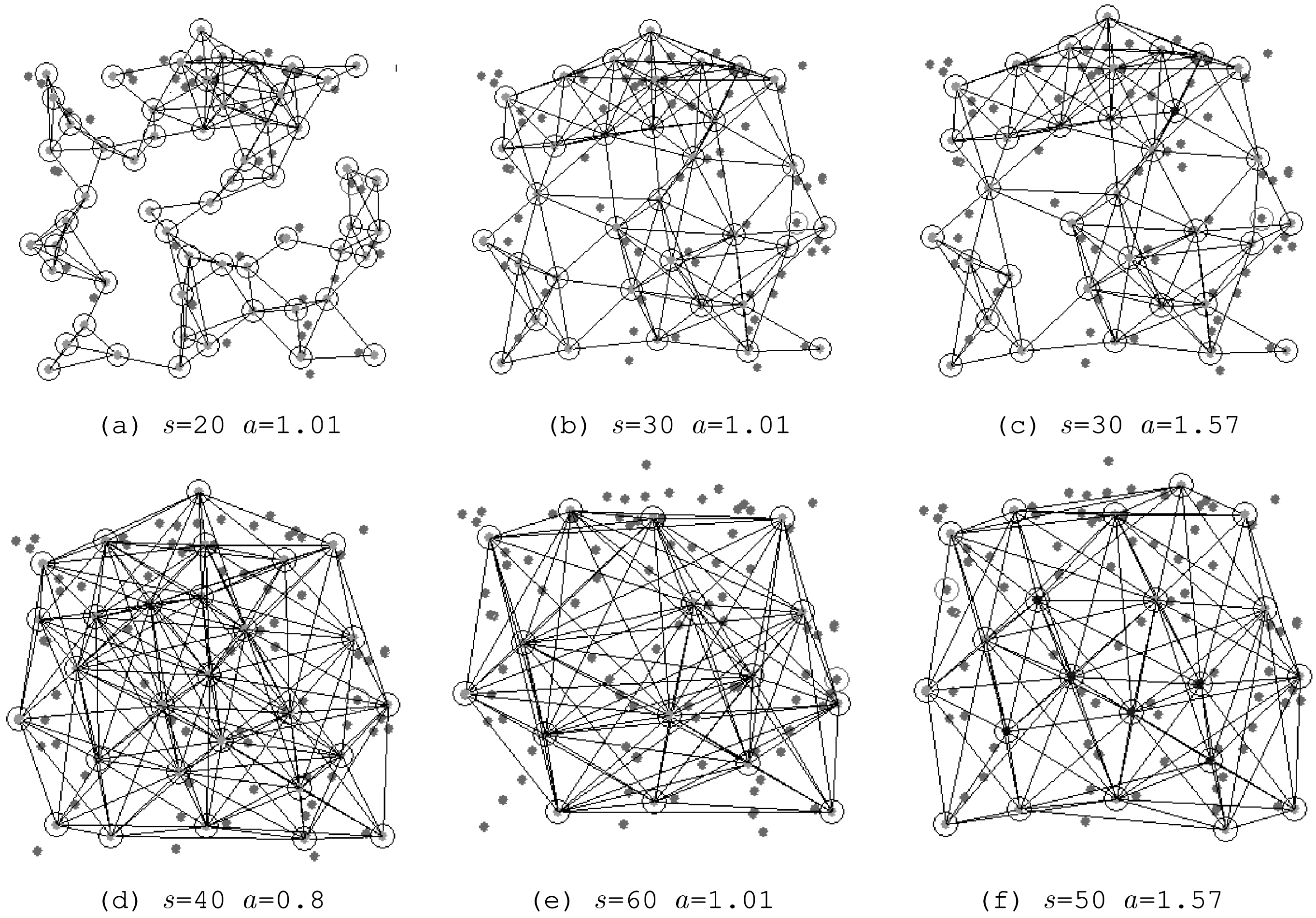

Fig. 7. Generated Network Topologies for Different Values of s and a. Dark dots are the ordinary nodes, and light dots with outer concentric circle are the clusterheads. A line between nodes u and v indicates the presence of an edge between clusters C_u and C_v in the cluster graph.

a. Figure 7 shows the qualitatively different clusterings that were generated by the service on the same distribution of mobile nodes. In Figure 7(a), the minimal distance between clusterheads $s = 20$ is small, as a result a large fraction of the nodes in the network are clusterheads. The cone angle a is set to 1.01 radians, which is close to the maximal cone angle (1.0107 radians) prescribed by Theorem 1, and therefore the cluster graph is sparsely connected. In comparison, the cluster graph of Figure 7(b) with $s = 30$ has fewer clusterheads but they are more densely connected. Increasing the cone angle a keeping s fixed at 30 we observe (Figure 7(c)) that the sparsity of the cluster graph increases. Increasing s to 40 and reducing the cone angle a to 0.8 results in a further decrease in the number of clusterheads and gives the densely connected cluster graph of Figure 7(d). With larger values of a, the gap_a predicate in the CC algorithm is violated with a lower power level for clusterheads which are surrounded by other clusterheads, and therefore the resulting cluster graph is sparser with smaller clusters. Comparing Figures 7(e) and 7(f) to Figures 7(b) and 7(c) respectively, we observe that for the same value of a, increasing s results reduction in the number of clusterheads and an increase in the sparsity of the cluster graphs.

In general, with larger values of s the connectivity of the cluster graph, the size of the clusters, and the power consumed increases, while larger values of a decreases connectivity and size of the clusters. Therefore, based on the requirements of a particular application the values of s and a can be so chosen as to produce desirable clustered decomposition.

B. Average Cluster Degree

Unlike sparsity (the maximum cluster degree), the average cluster degree measures how well the cluster graph is connected, on an average. Average cluster degree is an important robustness metric because it tells us the number of neighboring clusterheads that can fail before an average clusterhead gets disconnected from the rest of the cluster graph. The degree of cluster C_v is obtained simply by counting the number of elements in the set D_v. We take the average over all clusters and observe this value over time as the service stabilizes (Figure 8).

Our first observation is that, with $s = 40$ the stable value of the average cluster degree is higher than that with $s = 20$, irrespective of the value of a. Secondly, for each value of s, the average cluster degree is higher for a smaller value of a. Both these observations are in accordance with our expectations as explained in the previous section. It is to be noted that, for large values of s, a smaller fraction of clusterheads are located at the edge of the grid. Since, these edge clusters have fewer neighboring clusters, and therefore a lower cluster degree, the average degree of the cluster graph is lower than what would be expected otherwise.

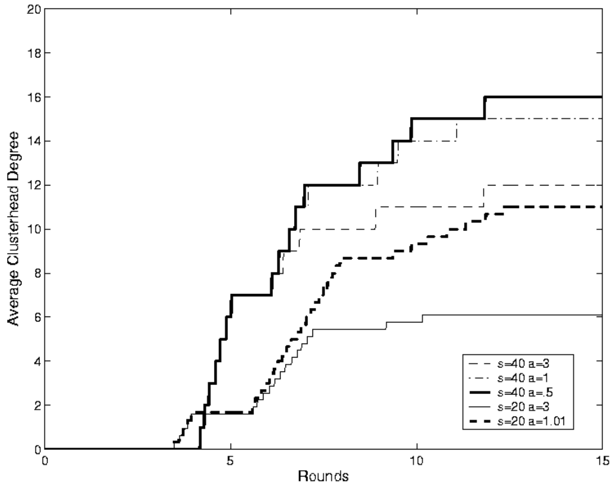

Fig. 8. Average degree of the clusterheads.

C. Stability after Failures and Joins

In this section we study the robustness of the clustering service under dynamic changes in the underlying network topology. The network topology changes when mobile nodes move, fail, or new nodes join the network, and we are interested to examine the *stabilization time*, that is, how quickly the remaining nodes of the network reach a agreeable state where they are organized into a cluster graph. In the simulator we measure stabilization time as follows: First we let the CC algorithm stabilize on the initial network of 100 randomly distributed mobile nodes; then, a certain number randomly chosen existing nodes are failed or new nodes are added in random locations, and we observe the number of rounds required for the network to re-stabilize. The number of execution rounds for re-stabilization give us an indirect measure of time required by the algorithm to reorganize the network.

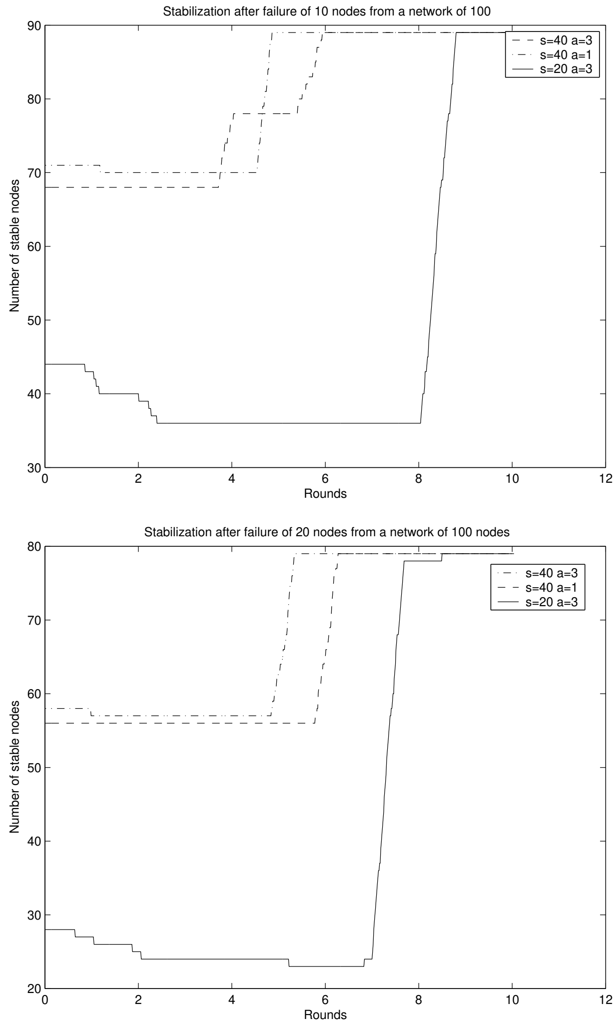

Fig. 9. Stabilization after node failures. Large and heavily overlapping cluster stabilize faster than small and sparse clusters.

The plots in Figure 9 show the changing number of stable nodes in the network after the set of nodes have failed. Initially the number of stable nodes decreases as the effect of failure propagates through the network, but this effect stops and eventually the new cluster structure emerges leading to stability of all the remaining nodes in the network. As expected, we observe larger clusters with greater overlaps ($s = 40$) regain stability more quickly than smaller clusters ($s = 20$). A smaller a results in larger clusters and more overlaps, but also takes longer time for the cluster control algorithm to terminate, so the effects of a on the the network stabilization time is complex and dependent on the number of failed nodes.

We performed a similar set of simulations with new nodes joining the stable network of 100 mobile nodes (Figure 10). There are two opposing effects that determine the total time to stabilize the network in this case: first, with larger clusters ($s = 40$) larger fraction of the new nodes turn out to be ordinary nodes and therefore are stable right when they join the network. This makes the initial number of stable nodes large for large values of s. Secondly, small and densely connected clusters ($s = 20$, $a = 1.01$) react more sharply to new nodes than large and sparse ($s = 40$, $a = 3$) clusters.

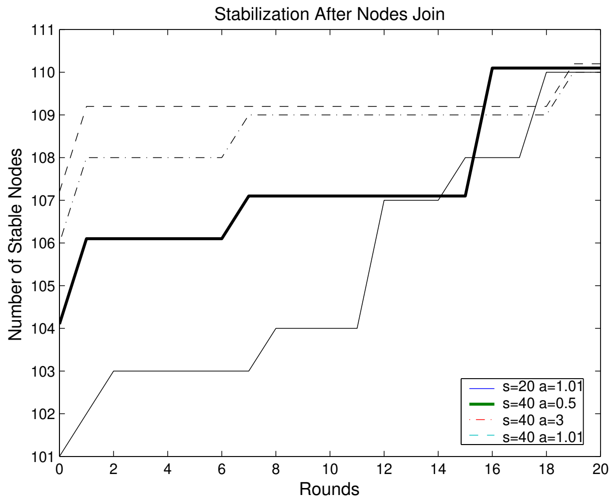

Fig. 10. Stabilization of algorithm after new nodes join the network. With larger values of s, fewer nodes are unstable initially, but the response of the clusterheads in absorbing the new nodes is also slower.

D. Network Longevity

Our last set of simulation results deal with the longevity of the mobile network with the nodes executing the clustering algorithm. We assume that each node is equipped with identical battery packs. The battery power consumed to transmit over a distance d is an n^{th} degree function of d, where $2 \leq n \leq 3$. For this simulation we assume that no new nodes join the network, the only failures are due to battery power outage, and that the nodes are static. Typically, the ordinary nodes broadcast infrequently and over short distances and the clusterheads broadcast frequently over longer distances. Therefore, clusterheads would typically run out of battery and die sooner than the ordinary nodes. Once a clusterhead dies, an ordinary node takes up the job of being the clusterhead and the cycle continues until the very last node runs out of power. The early expiration of the clusterheads can be mitigated by systematically rotating the broadcast responsibility of the clusterhead within the cluster, however we have not implemented this modification in our algorithm in presenting the following results.

Figure 11 shows the number of surviving nodes as the execution of the algorithm progresses over time. With large, dense clusters ($s = 40, a = 1.01$), there is a distinct 'knee' in the curve beyond which the number of surviving nodes diminish sharply. In contrast, sparser clusters ($s = 40, a = 3$) result in a gradually degrading network. Further, when the ($s = 40, a = 1.01$) curve starts to drop at the 'knee' point, the number of surviving nodes on ($s = 40, a = 3$) is already down to 45. Therefore, form the point of view of longevity, the preferred type of clustering would be determined by the type of degradation desired of the network.

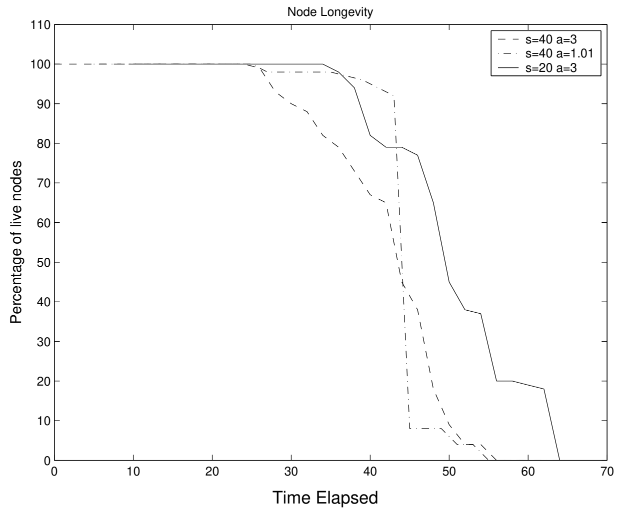

Fig. 11. Lifespan of network executing the clustering algorithm. There is a distinct knee in the curve for large dense clusters($s = 40, a = 1.01$), whereas sparse clusters ($s = 20, 40, a = 3$) result in a graceful degradation of the network.

VII. Conclusions

The merits of a clustered decomposition for a given network graph depends on the MANET application which uses the clustering. For most applications, however, it is desirable to have a connected cluster graph. In this paper we have proposed an adaptive clustering service that can be tuned to suit particular MANET applications. The cluster control algorithm minimizes the transmission power of the clusterheads while guaranteeing that the produced clusters are connected whenever it is physically possible.

The clustering service does not rely on global location information. Two parameters s - the inter-cluster distance and a - the maximum allowed angular gap between neighboring clusterheads, are used to control the size and sparsity of the clusters produced by the service, and thereby achieve the desired trade-offs among latency, energy efficiency, and robustness. We have shown that $2 \sin^{-1} \frac{\sqrt{15}}{8}$ (approx. 1.0107 rads) is the optimal value of a which minimizes the transmission power of the clusterheads while guaranteeing connectivity of the cluster graph, provided that the underlying network is connected when all nodes broadcast with maximum power. We have presented experimental results showing the qualitatively different type of clustered organizations that can be obtained from the algorithm. Our simulation experiments demonstrate that the algorithm rapidly recovers from instability caused by node failures and joins.

The empirical evidence presented here suggests that the algorithm has nice self stabilization properties; in the future we plan to rigorously analyze its behavior under dynamic topology changes and also extend the results to three dimensional distribution of mobile nodes. Our long term goal is to focus on particular MANET applications (e.g., tracking, routing), and use this clustering algorithm in conjunction with the chosen application to examine the performance of the application as a function of the cluster metrics.

VIII. Acknowledgments

The authors would like to thank to Ben Leong and Hari Balakrishnan for their comments and suggestions on this work.

References

[1] R. Prakash A. Amis, T. Vuong, and D. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM*, pages 32–41, March 1999.

[2] 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, Nov 1981.

[3] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 310–315., Perth, Western Australia, June 1999.

[4] R. Choudhury and N. Vaidya. On ad hoc routing using directional antennas. In *Illinois Computer Systems Symposium (iCSS)*, 2002.

[5] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Information Processing Letters*, 61(3):121–127, February 1997.

[6] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[7] A. Ephremides, J. E. Wieselthier, , and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. In *IEEE*, volume 75, pages 56–73, Jan 1987.

[8] Yaacov Fernandess and Dahlia Malkhi. K-clustering in wireless ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, Toulouse, France, 2002. ACM Press.

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

[10] P. Gupta and P. Kumar. Capacity of wireless networks. In *IEEE Transactions on Information Tehory*, volume IT-46, pages 388–404, 2000.

[11] M. T. Hajiaghayi, M. Bahramgiri, and V. S. Mirrokni. Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multi-hop networks. In *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks (IC3N)*, pages 392–398, Miami, Floria., October 2002.

[12] Vikas Kawadia and P. R. Kumar. Clustering by power control in ad hoc networks. In *Proceedings of IEEE MILCOM*, Atlantic City, NJ, October 1999.

[13] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan. A cluster-based approach for routing in dynamic networks. In *ACM SIGCOMM Computer Communication Review*, pages 49–65, April 1997.

[14] Taek Jin Kwon and Mario Gerla. Clustering with power contol. In *Proceedings of IEEE MILCOM*, Atlantic City, NJ, October 1999.

[15] Li Li, Joseph Y. Halpern, Paramvir Bahl, Yi-Min Wang, and Roger Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 264–273. ACM Press, 2001.

[16] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

[17] A. K. Parekh. Selecting routers in ad hoc wireless networks. In *ITS*, 1994.

[18] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, 2000.

[19] Ram Ramanathan and Regina Hain. Topology control of multihop wireless networks using transmit power adjustment. In *INFOCOM (2)*, pages 404–413, 2000.

[20] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[21] Roger Wattenhofer, Li Li, Paramvir Bahl, and Yi-Min Wang. Distributed topology control for wireless multihop ad-hoc networks. In *INFOCOM*, pages 1388–1397, 2001.

[22] Bhaskar Krishnamachari Yasser. The energy-robustness tradeoff for routing in wireless sensor networks. url:citeseer.nj.nec.com/552652.html.

APPENDIX

Proof of Lemma 1.

Proof. Draw circles A and B with radii R and centers u and v, respectively. Let the points of intersection of $\overline{uz}$ with A and B be c, b, and b' respectively. We consider three cases based on the position of w in $\overline{uz}$ (see Figure 12).

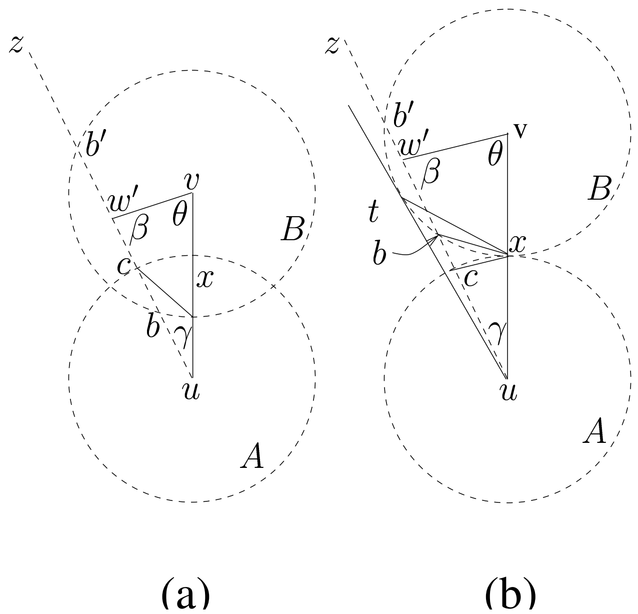

Fig. 12. Nodes are present at u, v, x, and w which is any point on $\overline{uz}$, with $|\overline{uv}| > |\overline{uw}|$.(a) Cases 1 and 2: node w is located either inside circle A or B. (b) Case 3: node w is between c and b but outside both A and B.

Case 1: w is located to between b and z. Let w' be the closest possible location of w from z. Since $\sin\gamma < \frac{\sqrt{15}}{8}$ and $|\overline{uw'}| < |\overline{uv}|$ it follows that $\beta > \theta$ and $\sin\beta > \frac{\sqrt{15}}{4}$. Using $\frac{\sin\beta}{|\overline{uv}|} = \frac{\sin\gamma}{|\overline{vw'}|}$, we get $|\overline{vw'}| \leq |\overline{uv}|/2 \leq R$. Therefore, for any location of w in $\overline{bw'}$, $|\overline{wv}| \leq R$.

Case 2: w is located between c and u. From $|\overline{ux}| \leq R$, $|\overline{ua}| = R$, and $\gamma < \frac{\pi}{3}$, it follows that $|\overline{xc}| \leq R$. Therefore, for any location of w in $\overline{ua}$, $|\overline{wc}| \leq R$.

Case 3: w is located in between c and b and outside of both A and B (Figure 12(b)). First we show that $|\overline{xb}| \leq R$. Let $\overline{ut}$ be a tangent to B on the same side of $\overline{uv}$ as $\overline{uz}$. Since $|\overline{uv}| \leq 2R$, it follows that $|\overline{xt}| \leq R$. Therefore $|\overline{xb}| \leq R$. From case 2 it is known that $|\overline{xc}| \leq R$, it follows that for any position of w between a and b, $|\overline{xw}| \leq R$. $\square$