A Clustering Scheme for Hierarchical Control in Wireless Networks

Suman Banerjee, Samir Khuller
Department of Computer Science, University of Maryland at College Park
{suman, samir}@cs.umd.edu

Abstract—**In this paper we present a clustering scheme to create hierarchies for wireless networks. A cluster is defined as a subset of vertices, whose induced graph is connected. In addition, a cluster is required to obey certain constraints that are useful for hierarchical routing. While all these constraints cannot be met simultaneously for general graphs, we show how for wireless network topologies, such a clustering can be obtained. Finally, we present an efficient distributed implementation of our scheme for a set of wireless nodes to create the desired clustering.**

I. Introduction

Rapid advances in hardware design have greatly reduced cost, size and the power requirements of network elements. As a consequence, it is now possible to envision networks comprising of a large number of such small devices. In the Smart Dust project at UC Berkeley [28] and the Wireless Integrated Network Sensors (WINS) project at UCLA [2] researchers are attempting to create a wireless technology, where a large number of mobile devices, with wireless communication capability, can be rapidly deployed and organized into a functional network.

Hierarchical structures have been used to provide scalable solutions in many large networking systems that have been designed. For networks composed of a large number of small, possibly mobile, wireless devices, a static manual configuration would not be a practical solution for creating such hierarchies. In this paper, we focus on the mechanisms required for rapid self-assembly of a potentially large number of such devices. More specifically, we present the design and implementation of an algorithm that can be used to organize these wireless nodes into clusters with a set of desirable properties.

Typically, each cluster in the network, would select a "cluster-representative" that is responsible for cluster management – this responsibility is rotated among the capable members of the cluster for load balancing and fault tolerance.

Target Environment : While our clustering scheme can be applied to many networking scenarios, our target environment is primarily wireless sensor networks [23], and we exploit certain properties of these networks to make our clustering mechanism efficient in this environment. These networks comprise of a set of sensor nodes scattered arbitrarily over some region. The sensor nodes gather data from the environment and can perform various kinds of activities depending on the applications – which include but is not limited to, collaborative processing of the sensor data to produce an aggregate view of the environment, re-distributing sensor information within the sensor network, or to other remote sites, and performing synchronized actions based on the sensor data gathered. Such wireless networks can be used to create "smart spaces", which can be remotely controlled, monitored as well as adapted for emerging needs.

Applicability : The clustering scheme can be used as a service for different applications in wireless sensor networks (or other environments). For example, it can be used to provide a scalable address allocation mechanism, or a scalable service location and discovery service in the network, analogous to the Service Location Protocol [25] by distributing the responsibility of necessary management in each cluster, to the clusterheads, and to provide location management of devices for QoS support as in [22].

Our target application of the clustering mechanism is providing a hierarchical control infrastructure in the wireless sensor networks. Hierarchical control infrastructures have been used to create routing solutions in the literature [3], [20]. Clustering-based hierarchies have been proposed for wireless networks in [9], [17],

[14] and [15].

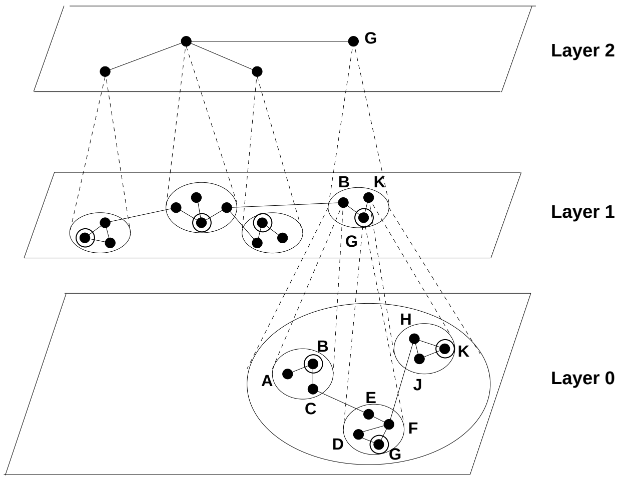

Fig. 1. An example of a three layer hierarchy

We derive the motivation of our work from the need to generate an applicable hierarchy for multi-hop wireless environment as defined in the Multi-hop Mobile Wireless Network (MMWN) architecture [22]. Such an architecture may be used to provide different distributed scalable services, including routing and state aggregation. In this architecture, wireless nodes are either *switches* or *endpoints*. Only switches can route packets, but both switches and endpoints can be source or destination of data. In wireless sensor networks, all the sensor devices deployed will be identical, and hence we treat all the nodes as switches, by MMWN terminolgy. Switches are expected to autonomously group themselves into clusters, each of which functions as a multi-hop packet radio network. A hiearchical control structure as illustrated in Figure 1, with nested clusters permits the network to scale to a large number of nodes. An instance of the clustering scheme operates on each of the *layers*, comprising of peer entities (all switches in Layer 0 or cluster-representative switches in the higher layers). Connectivity between adjacent clusters are provided by *virtual gateways*, that essentially are pairs of peer switches, in the two different clusters. Each virtual gateway comprises of multiple such peer pairs. Other details on routing and addressing schemes within and outside the cluster may be found in [22].

The MMWN architecture is "not currently designed to operate effectively in a network comprised of highly mobile nodes". Hence, a routing scheme created using clustering-based control infrastructure, like MMWN, is most effective in a slower mobility domain than a flat routing scheme using on-demand routing solutions for highly mobile environments like Dynamic Source Routing (DSR) [13], Ad-hoc On-Demand Distance Vector routing protocol (AODV) [19] and Temporally Ordered Routing Algorithm (TORA) [30], as shown in Figure 2, but would have the benefits of scalability of a hierarchical scheme.

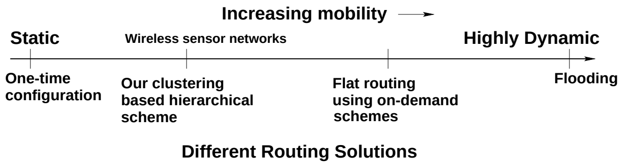

Fig. 2. A routing scheme using a clustering-based hierarchical structure would be most useful in a slowly changing mobility domain

We hypothesize that topology changes in wireless sensor networks will also be slow and infrequent. Once a cloud of sensor devices is deployed, they would mostly stay stationary. New nodes will occasionally join the network, by drifting into the vicinity of the existing network, or some existing nodes will drift away or disappear (e.g., due to loss of power). The clustering scheme needs to maintain clusters across such topology changes and we address such issues in Section IV.

Desired goals of the clustering scheme : For the hierarchical routing infrastructure described above, we postulate the following desirable properties that should be present in the clustering mechanism. that runs in each layer of the hierarchy.

- **Each cluster is connected.** This is an obvious requirement to localize and restrict cluster traffic to within the clusters.
- **All clusters should have a minimum and maximum size constraint.** A maximum size constraint limits the cluster size to within what can be efficiently maintained by a cluster head. It can also be chosen to bound the amount of state that needs to be maintained with the cluster. Ideally, we want all clusters to be of the same size, so that no cluster is overburdened, or under-burdened with processing and storage requirements of cluster maintenance. Small clusters are inefficient use of resources at the nodes, while large clusters increase the overheads. For ease of the clustering

scheme design, we set the minimum cluster size to be half the maximum size.

• **A node in any layer of the hierarchy belongs to a constant number of clusters in that layer.** Ideally, we would want a node to belong to only a single cluster in a layer. However, it is apparent, that for connected clusters with sizes bounded as described in the previous property, such a requirement cannot always be met. Some nodes in the network that have very high degree, might need to be included in each cluster, that would be otherwise disconnected, since this can be the only node that connects the remaining members; the connected sub-parts independently may be too small to be acceptable as clusters. Hence, we require that each node in a layer, belong to only a few clusters in the layer. Even this requirement may not be satisfiable for generic network topologies. There are certain network topologies, (e.g. the star network) as discussed in Section II, for which some nodes would need to be part of a large number of clusters, to keep the clusters connected and within the desired size bounds. However, we are able to leverage special properties due to the broadcast nature of wireless networks to guarantee a small constant upperbound for the number of clusters to which a node belongs to. [1]

• **Two clusters (in any layer) should have low overlap.** All nodes common to two clusters, will have to maintain cluster state and carry intra-cluster traffic for both the clusters, Ideally, clusters should have no overlap. But as discussed above, since in some topologies, some nodes might have to belong to more than a single cluster, zero overlap is not possible. Our clustering scheme guarantees that no two clusters in a layer will have an overlap of more than one node.

Main contributions : In this paper we propose a clustering scheme to create a layered hierarchy, similar to desired in MMWN, for wireless networks. We define our clustering problem in a graph theoretic framework, and present an efficient distributed solution that meets all desirable properties mentioned earlier. For arbitrary graph topologies, sometimes no solution may exist that can satisfy all the requirements of a desirable solution. But in wireless network topologies, properties of the underlying communication graphs may be exploited to achieve desired solutions, as we demonstrate in this paper.

The rest of the paper is structured as follows. We pose our problem in a graph theoretic framework in Section II. We discuss the clustering algorithm in Section III. In Section IV, we demonstrate how our clustering algorithm can be implemented in a distributed environment as the sensor network. We evaluate our clustering scheme through simulations in Section V. Finally, we discuss related work in Section VI and conclude in Section VII.

II. Problem Statement

We first define a generic network clustering problem as follows: Given an undirected connected graph $G = (V, E)$, and a positive integer k, such that, $1 \leq k \leq |V|$, find a collection of subsets (clusters), $V_1, \ldots, V_l$ of V, so that the following conditions are met.

1. $\cup_{i=1}^{\ell} V_i = V$. All vertices are part of some cluster.
2. $G[V_i]$, the subgraph of G induced by V_i is connected.
3. $k \leq |V_i| < 2k$. This is the size bound for the clusters.
4. $|V_i \cap V_j| \sim O(1)$. Two clusters should have up to a small constant number of common vertices. We show, later in the section, why all clusters cannot be guaranteed to be non-overlapping and yet meet the other requirements. [2]
5. $|S(v)| \sim O(1)$, where $S(v) = \{V_i | v \in V_i\}$, i.e.. a vertex belongs to a constant number of clusters.

We note that there may not be a feasible solution to the above problem for any general graph. Requirement (5) would be violated in a star graph (see Figure 3). For $k \geq 2$, any cluster in the graph would include the center vertex, for the cluster to be connected. Hence, for the center vertex, c, we would have $|S(c)| \sim O(\frac{n}{k})$, violating requirement (5) of the problem statement.

However, the underlying graph structure for a network of wireless nodes has certain useful properties that can be exploited. A wireless node A, can communicate with another node B, if and only if, B lies within the transmission radius, R_A, of node A. The underlying graph, in this case, would have a directed edge $A \to B$. For our algorithm, we only consider

[1] It should be noted, that in our clustering scheme, most of the nodes belongs to only a single cluster in each layer and the few that does not are within the small constant bound as described.

[2] Our clustering algorithm actually guarantees that $|V_i \cap V_j| \leq 1$, i.e. two clusters overlap in not more than one vertex.

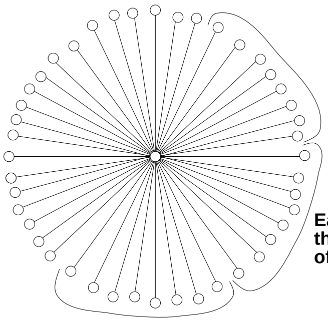

Fig. 3. Clustering in a star graph with one central vertex and $n-1$ radial vertices

bi-directional edges. So, a valid edge in the graph reflects the fact that both the nodes are within each other's transmission range, i.e., $d(A, B)$, the distance between the nodes A and B is at most $\min(R_A, R_B)$, for them to have an edge in the graph. This is in conformance with the assumptions made for most MAC protocols for wireless environments, including MACA [7], MACAW [8], IEEE standard 802.11 [1], FAMA [16] and RIMA [24].

We first consider the case when all nodes in the network have the same transmission range. In this case, the underlying communication graph, is a *Unit Disk graph* – defined in [6], [21] in terms of "distance" or "proximity" models, which consist of a value $d \geq 0$ and an embedding of the vertices in the plane, such that (u, v) is an edge iff $d(u, v) \leq d$. For such graphs, it can be seen that if a node has many neighbors, i.e., a vertex has very high degree, then all these vertices will be within its transmission radius. These neighboring nodes will be relatively close to each other and consequently will have edges between themselves. This would prevent the graph from having dense "star-like" components embedded in them. This is proved rigorously in Section III-C. We exploit this feature to guarantee that each vertex in the graph is in at most three clusters [3]. This is not possible in general graph topologies, as shown before.

Since the transmission range depends on the power available at the node, in general, for a homogeneous set of sensor nodes, the transmission radii would be close to each other. We also consider the case where different nodes may have different transmission radii. For such scenario, our clustering algorithm would guarantee that no node is a member of more than $O(\log(\frac{R_{\max}}{R_{\min}}))$ clusters, where $R_{\max}$ and $R_{\min}$ are the maximum and minimum transmission radii respectively. We use the term *Bounded Disk graph* to classify these underlying topologies [4]. Hence, the algorithm does not violate Requirement (5) even when orders of magnitude difference exist between the transmission power of the nodes.

If there are nodes with very small transmission radii, then the bound on $|S(v)|$ may be large, but in general, nodes with very small transmission radii would be nearly disconnected from the rest of the network, and can be considered "dead" for all practical purposes.

For the rest of the paper, we will focus only on communication graphs that are either unit disk graphs (for wireless nodes with identical transmission radii R) or bounded disk graphs (where the transmission radii of the nodes are bounded between $R_{\min}$ and $R_{\max}$) [5]. Even for these graphs, to satisfy Requirements (2) and (3), may lead to violation of Requirements (4) and (5), as shown below.

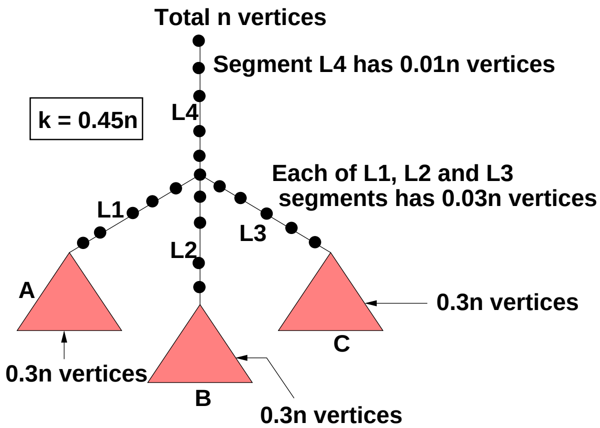

Fig. 4. Violation of constant sharing between clusters (with size parameter k), even for disk graphs

[3]In this paper, we only illustrate the results for two-dimensional topologies. The scheme works for any D-dimensional space, with the upper bound of $S(v)$ being a function of *only* the dimensionality, D.

[4]From our algorithm and its proof, it will be apparent that one can construct some specific Bound Disk Graphs, for which some nodes will necessarily belong to $O(\log(\frac{R_{\max}}{R_{\min}}))$ clusters, to meet the conditions of the problem statement.

[5]Our clustering technique can also be applied to general graph topologies, if we remove the Requirement (5), that each node belong to a constant number of clusters. The upper bound on the number of clusters a node belongs to, *for general graphs*, is the maximum degree of a node, which is usually low for many topologies.

Requirement (4) would be violated even in unit disk graphs as shown in Fig. 4 for any clustering algorithm. The total number of vertices in the tree is n. The zones A, B and C each have $0.3n$ vertices, the segments L1, L2 and L3 have $0.03n$ vertices, while the segment L4 has $0.01n$ vertices. If k is set at $0.45n$, any cluster will have vertices from at least two of the zones A, B and C. Also, from condition 2, the maximum size of a cluster is $0.9n$, and so there must be at least two clusters to cover all the vertices. Let us choose two such clusters, C_1 and C_2, and let C_1 have vertices from zones A and B (and maybe some other vertices too) and let C_2 have vertices from zones B and C (and some other vertices as might be necessary). To keep each cluster connected, we must have all vertices in segment L2 belong to both the clusters. This would mean $|C_1| \cap |C_2| \geq |L2| = 0.03n$, i.e., overlap, which is linear in the number of vertices.

Hence, we modify our Requirement (3) as follows :
1. $\forall i, |V_i| < 2k$.
2. $\forall i($ except one $) k \leq |V_i|$, we allow one single cluster in the entire graph to have size smaller than k.

Under such a relaxation, it is possible to cluster the graph in Fig. 4, by making C_1 include zones A and B and the segments L1, L2 and L4, and C_2 include zone C and segment L3.

Hence, our exact problem can be refined as stated below :

Given a disk graph $G = (V, E)$, and a positive integer, k, such that, $1 \leq k \leq |V|$, for each *connected* component of G, find a collection of subsets $V_1, \ldots, V_l$ of V, so that
1. $\cup_{i=1}^{l} V_i = V$.
2. $G[V_i]$, the subgraph of G induced by the vertices V_i, is connected.
3. The sizes of the subsets are bounded as follows :
 (a) $\forall i, |V_i| < 2k$.
 (b) $\forall i($ except one $) k \leq |V_i|$, i.e., we allow one single cluster in the entire graph to be smaller than k.
4. $|V_i \cap V_j| \sim O(1)$
5. $|S(v)| \sim O(1)$, where $S(v) = \{V_i | v \in V_i\}$, i.e.. a vertex belongs to a constant number of subsets.

Next, we state and prove the algorithm, first for unit disk graphs, when all nodes have the same transmission radius, R. Subsequently, we show how the same algorithm can be applied for bounded disk graphs, where nodes have varying transmission radii, but with Requirement (5) modified as $|S(v)| \sim O(\log(\frac{R_{\max}}{R_{\min}}))$.

III. Solution

We first outline the algorithm as it applies to a connected graph. If the underlying communication graph is not connected, we can apply this algorithm to each connected component of the graph.

A. Overview of the Solution. : The algorithm proceeds by finding a (rooted) spanning tree of the graph. One could use a Breadth-First-Search tree, or any other tree. The main advantage of a BFS tree is that it has a radius, which is bounded by diameter of the graph.

The algorithm runs in linear time. Let T be the rooted spanning tree, and $T(v)$ denote the subtree of T rooted at vertex v. We use $|T(v)|$ to denote the size of the subtree rooted at v. Let $C(v)$ be the set of children of v in T.

We assume that $|V| \geq 2k$, else we can treat the entire graph as one cluster. First we identify a node u such that $|T(u)| \geq k$ such that for each $v \in C(u)$ we have $|T(v)| < k$. It is clear that such a node always exists. Let $C(u)$ consist of ℓ nodes $v_1, \ldots, v_\ell$.

If we do not worry about Requirement (5), it is easy to group the subtrees rooted at the children of u together into clusters of size between k and $2k$ and to connect them through u [6]. We are left with at most a single subtree of size $< k$. Such a partitioning of subtrees into clusters is easy to achieve since we can keep adding the subtrees to the current cluster as long as the size is less than k. The addition of a single subtree cannot increase the size to more than $2k - 1$.

In fact, this algorithm can be implemented via a post-order traversal of T. When we are visiting a vertex, we can check the size of the subtree rooted at that vertex. If the subtree has size $\geq 2k$ then we can trigger the above scheme. Once we output a set of clusters and consequently, delete the vertices that belong to these clusters, we can update the size of the current subtree and continue with the post-order traversal at the parent of this vertex.

The main problem with the above scheme is that u may belong to many clusters (in the worst case, proportional to the degree of u even though this bound

[6] If $|T(u)| < 2k$, then the entire subtree $T(u)$ might be grouped into a single cluster.

is unlikely to be achieved in practice). We will now make use of the properties of the disk graphs that arise in this application to avoid this problem.

We will prove that if we take any six vertices in the neighborhood of u, $N(u)$ then there exists a pair of them with an edge between them. Using this property, we can connect the subtrees rooted at nodes in $C(u)$ to each other, *without* using vertex u (except in a small number of clusters). In the pseudo-code description of the algorithm GRAPHCLUSTER, this can be observed as the second required condition to enter the **while** loop at Line 7. This will guarantee that vertex u belongs to at most a constant number of clusters. It is easy to see that the intersection of clusters will also have at most one vertex.

B. Detailed Description of Algorithm. : We use the following notations :

- T : A minimum spanning tree of graph G.
- $T(x)$: Subtree of T, rooted at vertex x.
- $ClusterSet$: The set of clusters created by the algorithm.
- $UnpChildren$: Variable used to store the set of remaining children (i.e. that has not been deleted) that are yet to be processed at a vertex.
- $PartialClusterSet$: Set of temporary clusters that have size $< k$.

The algorithm creates a BFS tree and traverses the the tree in post-order. Figure 5 shows the processing for a vertex, u, in the tree. The vertex v has already been visited. Since $|T(v)| \geq k$, cluster, A, has been created rooted at v and deleted from the tree (Line 4 of the GRAPHCLUSTER proce-

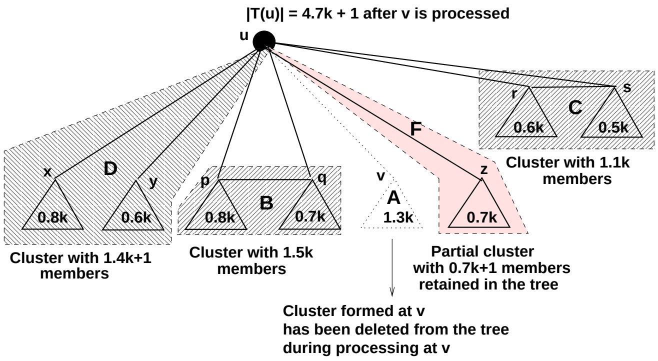

Fig. 5. Example operation at a vertex u for the GRAPHCLUSTER procedure

Proc. 1 : GRAPHCLUSTER(G, k)

```
1:  T ← BFS tree of G;  ClusterSet ← ⊥
2:  for u ∈ G, in post-order traversal of T do
3:    if (|T(u)| ≥ k) then
        {ASSERT : |T(v)| < k, v ∈ Children(u)}
4:      PartialClusterSet ← ⊥
        UnpChildren ← Children(u)
5:      while ∃v ∈ UnpChildren do
6:        TempCluster ← T(v)
          Remove v from UnpChildren
7:        while (|TempCluster| < k) ∧
            (∃x ∈ UnpChildren, s.t. x has an
            edge to w ∈ TempCluster) do
8:          TempCluster ←
              TempCluster ∪ T(x)
9:          Remove x from UnpChildren
10:       end while
11:       if (|TempCluster| ≥ k) then
12:         ClusterSet ←
              ClusterSet ∪ {TempCluster}
13:         Remove all subtrees in TempCluster
14:       else
            {ASSERT : No such x is found}
15:         PartialClusterSet ←
              PartialClusterSet ∪ TempCluster
16:       end if
17:     end while
18:     MERGEPARTIALCLUSTERS(u, k,
            PartialClusterSet, ClusterSet)
19:     if (Children(u) = ⊥) ∧
          (u has been assigned to some cluster) then
20:       Remove u from the tree
21:     end if
22:   end if
23: end for
```

Proc. 2 : MERGEPARTIALCLUSTERS$(u, k, P, ClusterSet)$

```
1: C ← ⊥
2: while (P ≠ ⊥) do
3:   Pick an arbitrary partial cluster, p from P
4:   C ← C ∪ p;  Remove p from P
5:   if (|C| ≥ k) then
6:     ClusterSet ← ClusterSet ∪ {C ∪ {u}}
7:     Remove all subtrees in C;  C ← ⊥
8:   end if
9: end while
```

dure). Hence, at the time u is being processed, there are seven remaining children of $u : x, y, p, q, z, r, s$ and the remaining subtree size of $T(u)$ is $4.7k + 1$, that included u. The two clusters, B and C are formed first without including u in any of these clusters (They both satisfy the **if** condition on Line 11 of of GRAPHCLUSTER), and are placed into clusters on Line 12. When MERGEPARTIALCLUSTERS is called (Line 18), there are three partial clusters, the subtrees rooted at vertices x, y and z. In MERGEPARTIALCLUSTERS, the cluster D is formed using the partial clusters from the subtrees rooted at x and y, and vertex u is used to connect the two subtrees, which otherwise do not share common edges. Finally, a single partial cluster (the subtree rooted at vertex z) is left. u is added to this partial cluster to form the partial cluster F, and this is the only subtree that remains in the tree, and each of the other vertices have been placed into some cluster and deleted. Hence, when processing is done at the parent vertex of u, the subtree rooted at u comprises of only the vertices in the partial cluster, F.

C. Proof of Correctness. : We now briefly outline the proof of correctness of the algorithm, described above. Complete details of the proof can be found in [31]. r

LEMMA III.1. *For unit disk graphs, the maximum independent set,(MIS), in the neighborhood, $N(u)$, of a vertex u has at most 5 vertices.*

Proof. The proof follows from simple geometric arguments. Let the distance parameter of the unit disk graph be R. Consider a vertex, u, s.t. $|N(u)| \geq 6$. Let some six of its its neighbors be, $v_1, \ldots, v_6$. Let the vertex indices be labeled in a cyclic order. Since, $v_i \in N(u), d(u, v_i) \leq R$. Consider vertices v_i and v_j, such that they are successive vertices in the cyclic order, i.e. $j = i(mod6) + 1$. If $(v_i, v_j) \notin E$, then $d(v_i, v_j) > R$. Also, $R \geq d(u, v_i)$ and $d(u, v_j)$. So, $d(v_i, v_j) > d(u, v_i)$ and $d(u, v_j)$. Hence, in $\triangle uv_iv_j, (v_i, v_j)$ is the largest side, and so $\angle u$ is the largest angle, which must be $> \frac{\pi}{3}$. Hence, $\sum_{i=1}^{6} \angle v_iuv_j > 6 \times \frac{\pi}{3} = 2\pi$, a contradiction. Hence, $\exists i$, such that $d(v_i, v_j) \leq R$, i.e., $(v_i, v_j) \in E$.

OBSERVATION 1. *When the algorithm terminates, each vertex is part of some cluster.*

OBSERVATION 2. *Each cluster formed by the algorithm is connected.*

OBSERVATION 3. *Only one cluster may have size $< k$ and all other clusters have sizes between k and $2k$.*

OBSERVATION 4. *Any pair of clusters has only one common vertex.*

OBSERVATION 5. *The number of partial clusters, created on exiting the* **while** *loop of lines 8-20 of* **Graph-Cluster**, *is 5.*

Proof. This follows from Lemma III.1.

LEMMA III.2. *Requirement 4 : A vertex v, is part of up to 3 distinct clusters.*

Proof. Refer to [31].

Thus, when the algorithm terminates, all the requirements for the solution is satisfied.

LEMMA III.3. *If $R_{\max}$ and $R_{\min}$ are the maximum and minimum radii respectively, then the maximum independent set in $N(u)$ has cardinality at most $O(\log \frac{R_{\max}}{R_{\min}})$.*

Proof. Let$I(u)$ be the largest independent set in the subgraph induced by $\{u\} \cup N(u)$. We define a "moat" $b(i)$ for $i \geq 0$, which is the annulus defined by circles of radii $c^i R_{\min}$ and $c^{i+1} R_{\min}$, centered at u (see Figure (6). (Note that the constant, c is chosen to be $\sqrt{3}$

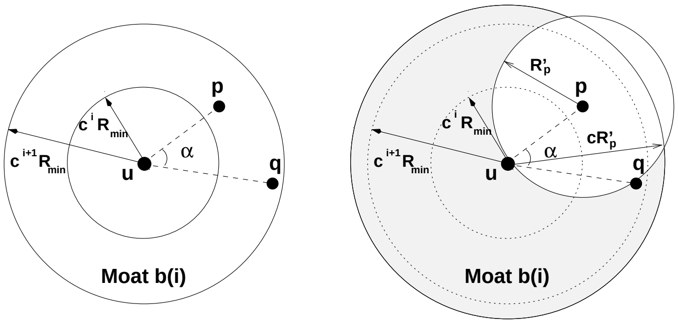

Fig. 6. Two vertices p and q in the neighborhood, $N_i(u)$ of u that are part of the maximum independent set in the subgraph induced by $N_i(u) \cup \{u\}$ have an angular separation, $\alpha > \frac{\pi}{6}$. They belong to the same moat $b(i)$, and there are $\log(\frac{R_{\max}}{R_{\min}})$ such moats.

as a scale factor, for ease of proof using geometric properties.) Let $N_i(u)$ be the neighbors of u that are in moat $b(i)$. Note that, any vertex, $v \in N_i(u)$, must have a transmission radius, $R_v \geq c^i R_{\min}$. This condition is needed for (u, v) to be an edge in the graph. We will prove that within each $N_i(u)$ there are at most 11 vertices in $I(u)$. Since the number of moats that contain vertices from $N(u)$ is $O(\log \frac{R_{\max}}{R_{\min}})$, the result follows.

Let p, q be two vertices that are in $N_i(u)$. Without loss of generality, let $R_p \leq R_q$. The distance between p and q is at least $\min(R_p, R_q) = R_p$ since there is no edge between p and q. We can "shrink" the circle centered at p with radius R_p until u is on the boundary of the circle (see Figure 6). This new radius satisfies $R'_p \leq R_p$ and $R'_p \geq c^i R_{\min}$. Notice that the distance from u to q is at most $c^{i+1} R_{\min} \leq cR'_p$. Draw a circle centered at u with radius cR'_p. Notice that q is inside this circle, but outside the circle centered at p with radius R'_p. This implies that q is in the crescent shaped shaded region.

Under these circumstances, the angle between p and q at u is $> \frac{\pi}{6}$, and by the same arguments as in Lemma III.1, there cannot be more than 11 vertices in the moat $b(i)$.

As a consequence of Lemma III.3, the algorithm `GraphCluster` would be applicable for Bounded Disk graphs, so that each vertex will be a part of $O(\log(\frac{R_{\max}}{R_{\min}}))$ clusters.

D. Algorithm Complexity. : The BFS computation of line 1 `GraphCluster`, takes $O(|E|)$. The computation at each vertex u, in post-order traversal, is $O(deg_T(u))$. i.e. the degree of u in the tree. Hence, the total cost for the entire post-order traversal is $\sum_u deg_T(u) = |V|$. Hence, the complexity of the algorithm is $O(|E|)$.

IV. Distributed Implementation

The algorithm that is described in Section III-B, is a centralized solution to the problem. In the distributed scheme, each wireless node in the network runs an identical protocol. The protocol has two phases : Cluster Creation and Cluster Maintenance. The cluster creation phase of the protocol is invoked very infrequently, when the existing clustering falls below a *quality* threshold. . Cluster maintenance is an inexpensive phase of the protocol that handles node mobility and other usual dynamics of the network.

A. Cluster Creation. This is a simple distributed extension of the centralized GRAPHCLUSTER algorithm. It can be initiated by any node in the network (the initiator will be the root of the BFS tree). if multiple nodes initiate Cluster Creation at the same time, simple tie breaking heuristics (e.g. initiator with least ID) is imposed to allow only one instance to proceed; the rest are not propagated.

There are two parts to cluster creation : Tree Discovery and Cluster Formation. The cluster formation can be easily piggybacked on the Tree Discovery component.

Tree Discovery : This is a distributed implementation of creating a BFS tree. Each node, u, transmits a beacon, which indicates its shortest distance in number of hops to the root, r. Any neighbor, v of u, will update its hop-distance to the root, if the route through u is a shorter path, and will choose u to be its parent in the tree. The beacon actually contains the following fields :

Some fields in the beacon may be continually updated during the cluster creation process (e.g. *parent-Id* will be initially `NULL`, em root distance and *subtree-size*), until they completely stabilize in their final value. The *root-Id* is used to distinguish between different Cluster Creation initiators, and to choose which of the beacons will not be propagated further. The subtree size at a node is updated as the subtree sizes of the children change as the BFS tree spreads to the entire network.

Cluster Formation : When a node, w, detects that its subtree size has crossed the size parameter, k, it initiates cluster formation on its subtree (e.g. similar to Line 3 of GRAPHCLUSTER). If the entire subtree $T(x)$ is of size $< 2k$, it can create one single cluster for the entire subtree, or else, it will create a set of clusters, (Lines 4-20 of GRAPHCLUSTER). Note, the creation of these set of clusters can be efficiently implemented with only very few further messages exchanged between x and its children. If all nodes in the subtree $T(x)$ has been placed into some clusters, x pretends to its parent in the tree that the subtree does not exist (equivalent to the deletions from the tree, in Lines 13 and 20 of GRAPHCLUSTER and Line 7 of MERGEPARTIALCLUSTERS), and the same process repeats higher up the tree. Node x will set a $Terminate$ flag in its next beacon (a *Terminate* beacon) to indicate its termination of cluster creation part. Cluster formation on the tree thus, proceeds from the leaves upwards, akin to the post-order traversal of the centralized scheme.

The beacons are transmitted by each node once every P units of time, over the duration of the cluster creation procedure. The period P should be chosen depending on the average connectivity of a node in the network and the bandwidth available from the wireless interface. With a high average connectivity of 10 nodes for a 10 Mbps network, assuming the beacons are 100 bits in length, choosing P as $10ms$ would bound the collision probability of beacons to less than 0.1. The terminate beacon ends the cluster creation at that node, and no further beacons need to be sent [7]. The average number of beacons sent by a node for the entire cluster creation phase is approximately bounded by the diameter of the network. Hence, with the above example of a very dense network with $10,000$ nodes, with a diameter of 140 [8], each node on an average sends 140 beacons, and the entire cluster creation phase would terminate in 3 seconds.

As the cluster creation phase ends, only the cluster information needs to be retained by the clusters. The BFS tree does not need to be maintained.

During the cluster creation phase, a few nodes may be missed (could be due to transient channel errors over the duration of the cluster creation phase). Such nodes will be able to join some cluster during the Cluster Maintenance phase, discussed next. The cluster creation protocol does not require a highly reliable or stable network, since

B. Cluster Maintenance. Once the cluster creation phase generates a set of clusters, the cluster maintenance phase is invoked to perform small incremental changes to the clusters to handle node mobility. arrive and existing nodes depart from the network (battery might run out). For the Cluster Maintenance phase, we relax the upper bound of the cluster size to $3k$. Hence, clusters in the maintenance phase can have sizes between k and $3k$ [9].

New node joins : A sensor node, v, on joining the network,establishes its set of neighbors, $N(v)$. If any node $u \in N(v)$ belongs to some cluster of size $< 3k-1$ then we add v to the cluster that u belongs to. This also ensures that we maintain connectivity in the cluster and the size requirement.

However, it is possible that that all neighbors belong to clusters of size $3k-1$. In this case, we add v to one such cluster, thus making its size, $3k$. Hence this cluster is split into two separate clusters. The splitting process involves the use of the same distributed implementation described in Cluster Creation phase. However, in this scenario, it is even simpler, since as soon as a cluster is identified to have size $\geq k$, the remaining $2k$ nodes can immediately be put into another cluster. Note, this splitting involves only the nodes in this cluster od size $3k$ being split. Hence, this splitting effort has a constant communication and time requirements unlike the clustering of the entire network, for which it is proportional to the diameter of the network. It can also be observed, that once a cluster is split, in this manner, it would require at least k other nodes joining one of the clusters (in the worst case) before another split would be necessary.

Triggering re-clustering : Using this cluster splitting technique, it might be necessary that the two clusters need to share a node, x, (otherwise one of the clusters will not stay connected). As a consequence $|S(x)|$ would increase. There are cases when $|S(x)|$ decreases too, However, by a pathological sequence of joins by other nodes to the set of clusters $S(x)$, the value of $|S(x)|$ can increase without bounds. We consider a clustering to be of "poor" quality, when the average of $|S(x)|$ exceeds a specified threshold. This triggers the cluster-creation phase to re-create a "good" clustering.

Existing node leaves : When a node leaves, it may cause the cluster(s) it belongs to, become disconnected. However, it may be shown (using the same arguments as in Lemma III.1 and III.3), that the number of remaining connected components of a cluster, due to a node leaving, will be bounded (using the same bounds as before for the Maximum Independent Set in the neighborhood of a single node). Any such connected component, that has its size $\geq k$, is made a cluster. For any component that has a size $< k$, will be disbanded as a cluster, and each node will attempt to join its neighboring clusters – the same mechanism of a node joining a

[7] A few more extra terminate beacons may be sent to handle losses of the beacons. Timeouts can be used by the parent to detect termination of children, if all the terminate beacons are lost.

[8] Assuming a distribution on a square grid

[9] At the end of cluster creation, however, all the clusters had size between k and $2k$. Hence, at the end of cluster creation, if a small cluster of size $< k$ exists (there can be only one such cluster), it is merged with another cluster of size $< 2k$.

cluster, applies for each of these nodes.

Link outages and network partitions : A link outage does not alter the number of nodes in the cluster. It may split the cluster into disconnected components, hence is equivalent to the cluster maintenance mechanism, when an existing node leaves. When a network partition happens, there may be two sets of clusters on each side of the partition. When the partitions join again, therefore, no special mechanisms are needed.

V. Experimental Results

We simulated the operations of our clustering scheme on a set of wireless nodes. For the simulation, we generate arbitrary wireless topologies. We randomly place a set of nodes in a 1000 unit $\times$ 1000 unit grid. We then choose a connectivity for the topology, by specifying the total number of graph edges that we desire, by changing the transmission radii of the node as a parameter. Bigger the transmission radii, higher will be the average degree of the graph. Nodes can arbitrarily join and leave the topology as they desire. In each node, we implement the tree discovery and the cluster formation protocols. In the experiments reported here, we chose to transmit the `root-distance` beacon once every second. The simulated time elapsed in the experiment, can therefore, be treated to have a linear scaling factor equal the `root-distance` beacon period. All nodes in the topology have the same transmission radii in these experiments.

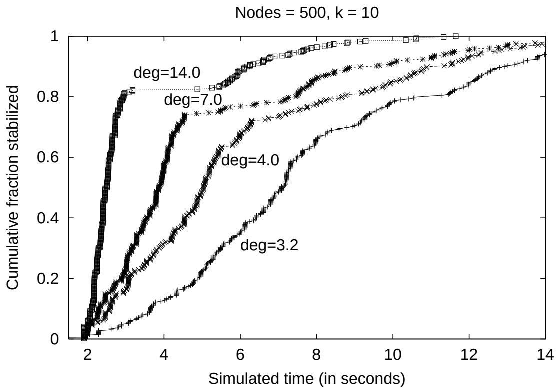

Fig. 7. Cumulative fraction of nodes stabilized in a wireless network with simulated time elapsed, for varying connectivity of the topology

Transmit Radius	Average Degree	Avg. Cluster Dia. for different k			
		$k = 10$	$k = 20$	$k = 25$	$k = 40$
69.1	3.2	4.8	8.2	9.4	12.4
85.0	5.0	4.3	6.7	7.3	10.5
100.6	7.0	3.9	5.5	6.1	7.9
174.0	20.0	2.2	3.4	3.7	4.4

Fig. 8. Cluster diameter for varying connectivity of nodes in the topology and varying k

Time to Stabilize : A wireless node is said to *stabilize* in the clustering scheme, when it undergoes no further changes in its cluster membership. In Figure (7), we plot results from a set of experiments, in a 500-node topology, where all the nodes join the network uniformly distributed between the simulation times of 1–2 seconds. The x-axis shows the simulated time elapsed. The y-axis indicates the cumulative fraction of the nodes that have stabilized. We plot the cumulative fraction stabilized for the same distribution of the nodes in the plane, but with different connectivity of the topology. As can be seen, when the average degree of the graph is low (3.2), due to a small transmission radii of the nodes, the time take by the nodes to stabilize is proportionally larger. When the average degree is very high (14.0), due to a large transmission radii, most of the nodes stabilize into their final clusters, fairly quickly. This is an an artifact of the diameter of the BFS tree formed. For highly connected graphs, the diameter is low, and so the clusters converge quickly.

Cluster diameter : In general, it is desirable to have clusters of low diameter. In the Figure 8, we show the average diameter of the different clusters for the 500-node topology. The cluster diameters increase with increasing value of k and with decreasing connectivity of the topology, as would be expected. An interesting extension to our algorithm would be to optimize a combined metric of the size parameter k, and the diameter of the cluster, d thus permitting a better configuration of the clusters. Such a problem, posed in the graph theoretic framework, would not have a guaranteed solution, and hence some heuristic approaches may be useful.

We ran experiments on networks with up to 1100 nodes using our simulator. In general, we found all clusters on stabilization, reach the desired cluster size bounds (except one, as per Requirement (3b)). Most of the nodes, were part of only a single cluster, and in all our experiments, we found less than 1% of the nodes to belong to multiple clusters.

VI. Related Work

Some routing solutions for the Internet have used hierarchies to provide scalability, e.g. OSPF protocol [18] and ATM PNNI [12]. Additionally, in some cases, ATM PNNI [12], [11] splits the network into clusters, called peer-groups, and only summarized information e.g., of cost of traversal, of the peer-groups is exported to the remaining network.

Krishnan et al [29] have explored different graph partitioning schemes for Internet-like graphs. Their target problem is, as a consequence, somewhat different from ours.

In mobile wireless environments, the Zone Routing Protocol (ZRP) [26], has the weak notion of groups, called zones, which are used to limit the propagation of updates. The notion of clustering has also been used previously for hierarchical routing for packet radio networks in [4] and [5]. In [9], [17] clustering algorithms are described for multi-hop mobile radio network, where the clusters are chosen such that the cluster-heads form a dominating set in the underlying graph topology. This makes the number and size of the clusters, largely dependent on the graph topology. They use it primarily for "spatial reuse" of channel spectrum. A similar mechanism is used in Cluster Based Routing Protocol (CBRP) [27] and a more generalized approach is used in [14] for mobile ad-hoc networks. Das et al [15] uses a connected dominating set to create a routing 'spine' and describes a clustering scheme to create two layered hierarchies. Krishna et al [10] defines a clustering scheme, where each cluster is required to be a clique. These mechanisms are possible by allowing small clusters to exist as may be needed.

VII. Conclusions

In this paper, we demonstrate how certain geometric properties of the wireless networks can be exploited to perform clustering with some desired properties. Generic graph algorithms developed for arbitrary graphs would not exploit the rich geometric information present in specific cases, e.g. the wireless network environment. Even without exact knowledge of node location in different environments, understanding various special properties of the communication graph can lead to better and efficient algorithms.

References

[1] IEEE Computer Society LAN MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. IEEE, 1997.

[2] *The WINS Project*, http://www.janet.ucla.edu/WINS/

[3] L. Kleinrock and K. Faroukh, *Hierarchical Routing for Large Networks*, Computer Networks, Vol. 1, 1997.

[4] D.J. Baker and A. Ephremides, *The architectural organization of a mobile radio network via a distributed algorithm*, IEEE Transactions on Communications. November 1981.

[5] D.J. Baker, J. Wieselthier and A. Ephremides, *A distributed algorithm for scheduling the activation links in a self-organizing, mobile, radio network*, IEEE ICC, 1982.

[6] B.N. Clark, C.J. Colbourn and D.S. Johnson, *Unit Disk Graphs*, Discrete Mathematics, 1990.

[7] P. Karn, *MACA - a new channel access method for packet radio*, ARRL/CRRL Amateur Radio 9th Computer Networking Conference, 1990.

[8] V Bhargavan, A. Demers, S. Shenker and L. Zhang, *MACAW : A Media Access Protocol for Wireless LANs*, Proceedings of Sigcomm, September 1994.

[9] M. Gerla and J.T.-C. Tsai, *Multicluster, mobile, multimedia radio network*, ACM-Baltzer Journal of Wireless Networks, Vol 1. No. 3, 1995.

[10] P. Krishna, M. Chatterjee, N.H. Vaidya and D.K. Pradhan, *A Cluster-based Approach for Routing in Ad-hoc Networks*, 2nd Usenix Symposium, April 1995.

[11] W.C. Lee, *Topology Aggregation for Hierarchical Networks*, ACM Computer Communications Review, Vol 25, No. 1, 1995.

[12] ATM Forum Technical Committee, *Private Network-Network Interface, Version 1.0*, May 1996.

[13] D.B. Johnson and D.A.Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, Mobile Computing, 1996.

[14] S. Basagni, I. Chlamtac, and A. Farago, *A generalized clustering algorithm for peer-to-peer networks*, Workshop on Algorithmic Aspects of Communication July 1997.

[15] B. Das, R. Sivakumar, and V. Bharghavan, *Routing in Ad-Hoc Networks Using a Spine*, International Conference on Computers and Communications Networks, September 1997.

[16] C.L. Fullmer and J.J. Garcia-Luna-Aceves, *Solutions to Hidden Terminal Problems in Wireless Networks*, Proceedings of Sigcomm, September 1997.

[17] C.R. Lin and M. Gerla, *Adaptive clustering for mo-*

bile, wireless networks, Journal on Selected Areas of Communication, 15, 7, September 1997.
[18] J. Moy, *Open Shortest Path First Version 2*, RFC 2178, July 1997.
[19] C.E. Perkins, *Ad-hoc On-demand Distance Vector Routing*, Internet-Draft, draft-ietf-manet-aodv-00.txt, November 1997.
[20] Z. Xu, S. Dai, and J.J. Garcia-Luna-Aceves, *Hierarchical Routing Using Link Vectors*, Proceedings of Infocom, March 1998.
[21] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz and R.E. Stearns, *NC-approximation schemes for NP- and PSAPCE-hard problems for geometric graphs*, Journal of Algorithms, 26(2), February 1998.
[22] R. Ramanathan and M. Steenstrup, *Hierarchically organized, multihop mobile wireless networks for Quality-of-Service Support*, Mobile Networks and Applications, Vol 3, No.1, June 1998.
[23] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, *Next Century Challenges : Scalable Coordination in Sensor Networks*, Proceedings of Mobicom, August 1999.
[24] J.J. Garcia-Luna-Aceves and A. Tzamaloukas, *Reversing the Collision-Avoidance Handshake in Wireless Networks*, Proceedings of Mobicom, August 1999.
[25] E. Guttman, C. Perkins J. Veizades and M. Day, *Service Location Protocol, Version 2*, RFC 2608, June 1999.
[26] Z.J. Haas and M.R. Pearlman, *The Zone Routing Protocol for Ad Hoc Networks*, Internet-Draft, draft-ietf-manet-zone-zrp-02.txt.
[27] M. Jiang, J. Li and Y.C. Tay, *Cluster-Based Routing Protocol (CBRP)*, Internet-Draft, draft-ietf-manet-zone-zrp-02.txt, August 1999.
[28] J.M. Kahn, R.H. Katz and K.S. Pister, *Next Century Challenges : Mobile Networking for "Smart Dust"*, Proceedings of Mobicom, August 1999.
[29] R. Krishnan, R. Ramanathan and M. Steenstrup, *Optimization algorithms for Large Self-structuring Networks*, Proceedings of IEEE Infocom, March 1999.
[30] V. Park and S. Corson, *Temporally Ordered Routing Algorithm Version 1, Functional Specification*, Internet-Draft, draft-ietf-manet-tora-spec-02.txt, October, 1999.
[31] S. Banerjee and S. Khuller, *A Clustering Scheme for Hierarchical Routing in Wireless Networks* CS-TR 4103, University of Maryland, College Park, USA, March 2000.