A Clustering Scheme for Hierarchical Routing in Wireless Networks

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

February 29, 2000

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. We also present simulation results from a distributed implementation of this scheme to demonstrate its convergence and stability properties.

1 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 [Ka 99] and the Wireless Integrated Network Sensors (WINS) project at UCLA [WINS] researchers are attempting to create this 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-head" that is responsible for cluster management. The cluster-head responsibility may be 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 [Es 99], 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 in Routing: 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 [Gu 99] by distributing the responsibility of necessary management in each cluster, to the cluster-heads, and to provide location management of devices for QoS support as in [Ra 98].

Our target application of the clustering mechanism is providing a hierarchical routing infrastructure in the wireless sensor networks. Hierarchical routing schemes have been proposed in the literature [Kl 77], [Be 98]. Scalability in Internet routing, is provided by a manually configured hierarchy of Autonomous Systems (AS). BGP4 [ReLi 95] is the standard inter-AS routing protocol, while other protocols like RIP [He 88], OSPF [Mo 97] and IGRP [IGRP] usually operate inside a single Autonomous System. The inter-AS routing protocol implemented on the border routers, typically, see an abstracted view of the Autonomous Systems.

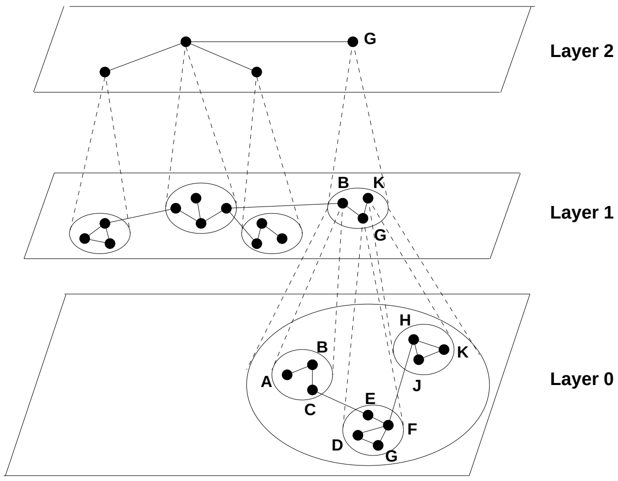

Figure 1: An example of a three layer hierarchy

Our clustering algorithm has been designed for topology aggregation in wireless sensor networks, in a manner similar to peer-groups in ATM PNNI specifications [ATM 96]. The overall routing mechanism that we hope to achieve is illustrated in Figure 1. The network is split into a hierarchy of "layers". By using our clustering scheme, all nodes in a layer are divided into clusters. A representative from each cluster in a layer, is made part of its immediate higher layer in the hierarchy. A local routing protocol is run internal to each cluster, in which each cluster member participates, and has exact topology information of the cluster. The cluster representative of the cluster, in the higher layer, only provides an abstracted view of the cluster to its peers in the higher layer. For example, in Figure 1, nodes D, E, F and G are part of the same cluster, and hence for the routing inside this cluster, each node is aware of the exact internal topology. However, in layer 1, node G has two neighbors. For the routing protocol running in the cluster comprising of B, G and K in layer 1, G advertises a simplified view of its cluster in layer 0 to each of its neighbors in layer 1. For example, G might represent its layer 0 cluster as a star graph, with two radial nodes E' and F' (corresponding to the border nodes E and F of the cluster) with the virtual links (E', G) and (F', G) having some representative costs, that correspond to the costs of using the virtual links. Performance studies on such topology representations have been done by Awerbuch et al [Aw 98] and Lee [Le 95]. Thus, like any hierarchical scheme, we reduce the state information that is required to be stored at a node. Clustering-based hierarchies have been proposed for wireless networks in [Ge 95], [Li 97], [Ba 97] and [Da 97] and we compare them to our clustering scheme in Section 6.

Desired goals of the clustering scheme: For the above hierarchical routing infrastructure to be beneficial, we need the following desirable properties from the clustering scheme that runs in each layer of the hierarchy.

- Each cluster is connected. This is an obvious requirement to do topology aggregation and management of 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 needed for the intra-cluster routing, at each member of the cluster. Ideally, we want all clusters to be of the same size, otherwise different nodes would have different storage and processing needs depending on the size of the cluster they belong to. Additionally, if some cluster is very small in size, the goal of clustering to perform topology aggregation is lost. Hence, we also impose a minimum size constraint. For ease of the clustering scheme design, we set the minimum cluster size to be half the maximum size. An advantage of making the lower and upper bound on the cluster size, to have the same order of magnitude, is that the error in accuracy of aggregated cluster topology is bounded.

- A node in any layer belongs to a constant number of clusters. For generic network topologies, we show that no clustering might exist that can guarantee connected and bounded clusters, as discussed above, with the additional constraint that each node belongs to only a constant number of clusters. We discuss such example network topologies (e.g. the star network) in Section 2. However, we are able to leverage special properties due to the broadcast nature of wireless networks to satisfy this constraint.

- No two clusters (in any layer) have more than one node in common. For generic network topologies and also for wireless network topologies, this is the minimum overlap that any clustering algorithm with the above mentioned properties, can hope to provide, worst case examples for which are discussed in Section 2.

In the wireless sensor network environment, the topology may change dynamically, as new nodes appear and existing nodes move or disappear (e.g., can be due to loss of power). The clustering scheme also needs to maintain clusters across such topology changes and we address such issues in Section 4. Broadly speaking, in mobile networks, there are different routing solutions that have been proposed, and are generally accepted under different mobility models, varying from near static topologies to very rapidly changing topologies. Given that there would be some overheads in cluster creation and maintenance, we expect our clustering-based routing infrastructure to operate in a much higher mobility domain than Mobile IP [Pe 00], but a somewhat slower mobility domain than a flat routing scheme which uses on-demand routing solutions, e.g. Dynamic Source Routing (DSR) [Jo 96], Ad-hoc On-Demand Distance Vector routing protocol (AODV) [Pe 97] and Temporally Ordered Routing Algorithm (TORA) [PaCo 99], as shown in Figure 2, Wireless sensor networks are not usually envisioned as rapidly changing environments, and hence would exist in this domain.

Main contributions: In this paper we propose a clustering scheme to create a layered hierarchy for routing in wireless networks. We define our clustering problem in a graph theoretic framework, and present an efficient distributed solution that meets all our desirable properties. For generic 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.

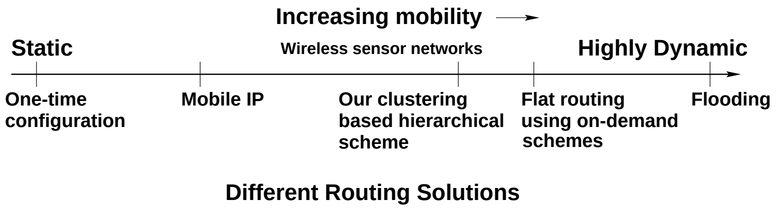

Figure 2: The clustering-based hierarchical routing scheme would be most useful in a domain between the operation points of Mobile-IP and flat routing schemes using on-demand protocols

The rest of the paper is structured as follows. We pose our problem in a graph theoretic framework in Section 2. We discuss the clustering algorithm in Section 3. In Section 4, we demonstrate how our clustering algorithm can be implemented in a distributed environment as the sensor network. Finally, we evaluate our clustering scheme through simulations in Section 5. Finally, we discuss related work in Section 6 and conclude in Section 7.

2 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, $V_1, \ldots, V_l$ of V, so that the following conditions are met.

1. $\cup V_i = V$. All vertices are part of some cluster.
2. $G[V_i]$, the subgraph of G induced by the vertices V_i is connected.
3. $k \leq |V_i| < 2k$. This is the size bound for the clusters.
4. $|V_i \cap V_j| \leq 1$. Two clusters should not have more than one vertex in common. We show, later in the section, why all clusters cannot be guaranteed to be non-overlapping and yet meet the other requirements.
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.

First, 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 have 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 $A \rightarrow B$. For our algorithm, we only consider 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 [Ka 90], MACAW [Va 94], IEEE standard 802.11 [802.11], FAMA [Fu 97] and RIMA [Ga 99].

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 [Cl 90], [Hu 98] 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

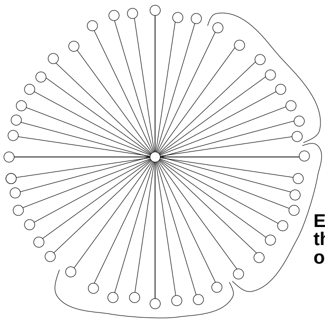

Figure 3: Clustering in a star graph with one central vertex and n-1 radial vertices

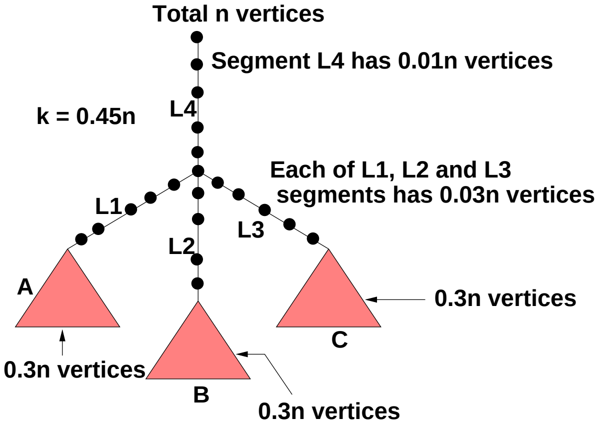

Figure 4: Violation of constant sharing between clusters (with size parameter k), even for disk graphs

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 3.3. We exploit this feature to guarantee that each vertex in the graph is in at most a constant number of clusters [1]. 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 graphs* to classify these underlying topologies. 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 $R_{\max}$) [2]. Even for these graphs, to satisfy requirements (2) and (3), may lead to violation of requirements (4) and (5), as shown below.

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

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

[2] 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 Internet-like topologies.

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 cluster, 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 :

(a) $\forall i, |V_i| < 2k$.

(b) $\forall i($ 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. We call a cluster having size $< k$, a *partial cluster*.
4. $|V_i \cap V_j| \leq 1$
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 graph, 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 the requirement 5 modified as $|S(v)| \sim O(\log(\frac{R_{\max}}{R_{\min}}))$.

3 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.

3.1 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 $|T(u)| \geq 2k$ such that for each $v \in C(u)$ we have $|T(v)| < 2k$. It is clear that such a node always exists. Let $C(u)$ consist of ℓ nodes $v_1, \ldots, v_\ell$. For each v_i with $|T(v_i)| \geq k$ the algorithm outputs a single cluster, namely $T(v_i)$, and removes it from the tree. Thus each remaining child has a subtree of size less than k.

Now it is easy to group the subtrees rooted at the children of u together into groups of size between k and $2k$ and to connect them through u. We are left with at most a single cluster of size $< k$. Such a partitioning is easy to achieve since we can keep adding the subtrees to the current partition 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 node we can check the size of its subtree. If the subtree has size $\geq 2k$ then we can trigger the above scheme. Once we output a set of clusters, we can update the size of the current subtree and return to the parent and continue the algorithm.

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

3.2 Detailed Description of Algorithm: We use the following notations :

- $G[X]$: denotes the subgraph of G induced by the set of vertices X.
- T : A minimum spanning tree of the 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 then visits each vertex in 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, and since $k \leq |T(v)| < 2k$, it has been made into a cluster, A, rooted at v and deleted from the tree (Line 4 of the `GraphCluster` procedure). Hence, at the time u is being processed, there are 5 remaining children of u, and the total subtree size or $T(u)$ is $6k + 1$, that included u. The two clusters, B and C are formed first without including u in any of these clusters (Line 13 of `GraphCluster`), since they have a connecting edges. When `MergePartialClusters` is called (Line 17), 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 all the other vertices are deleted. Hence, when processing at the parent vertex of u is done, $T(u) = F$.

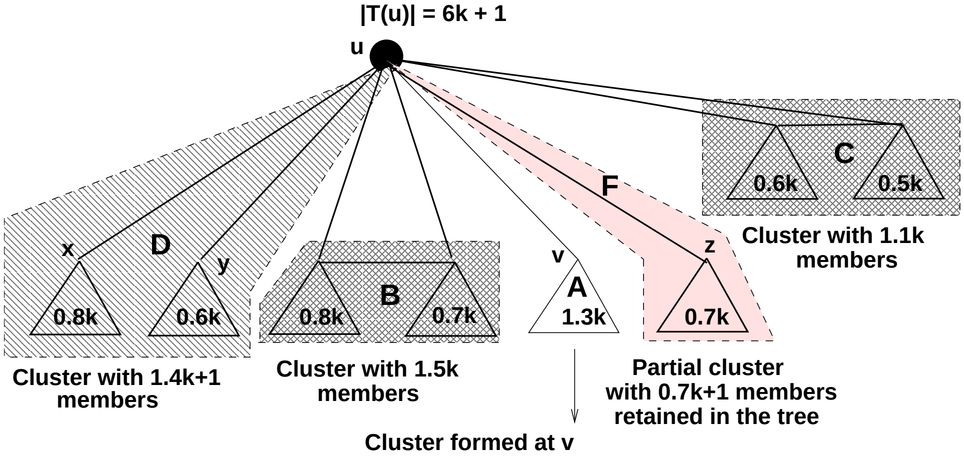

Figure 5: Example operation at a vertex u for the `GraphCluster` procedure

`GraphCluster`(G) —

$T \leftarrow$ BFS tree of G; $ClusterSet \leftarrow \perp$
for $u \in G$, in post-order traversal of T
  if $(k \leq |T(u)| < 2k)$
    $ClusterSet \leftarrow ClusterSet \cup \{T(u)\}$; Remove subtree $T(u)$
  if $(|T(u)| \geq 2k)$ { *Assertion : If true,* $|T(v)| < k, \forall v \in Children(u)$ }
    $PartialClusterSet \leftarrow \perp$; $UnpChildren \leftarrow Children(u)$
    while $\exists v \in UnpChildren$
      $TempCluster \leftarrow T(v)$; Remove v from $UnpChildren$
      while $(|TempCluster| < k) \wedge (\exists x \in UnpChildren$, s.t. x has an edge to $w \in TempCluster)$
        $TempCluster \leftarrow TempCluster \cup T(x)$
        Remove x from $UnpChildren$
      if $(|TempCluster| \geq k)$
        $ClusterSet \leftarrow ClusterSet \cup \{TempCluster\}$
        Remove all subtrees in $TempCluster$
      else { *Assertion : If true,* $(x = \perp)$ }
        $PartialClusterSet \leftarrow PartialClusterSet \cup TempCluster$
    `MergePartialClusters`$(u, PartialClusterSet, ClusterSet)$
    if $(Children(u) = \perp) \wedge$ (u has been assigned to some cluster)
      Remove u from the tree

`MergePartialClusters`$(u, P, ClusterSet)$ —

$C \leftarrow \perp$
while $(P \neq \perp)$
  Pick an arbitrary partial cluster, p from P
  $C \leftarrow C \cup p$; Remove p from P
  if $(|C| \geq k)$
    $ClusterSet \leftarrow ClusterSet \cup \{C \cup \{u\}\}$
    Remove all subtrees in C; $C \leftarrow \perp$

3.3 Proof of Correctness: We now prove the correctness of the algorithm, described above.

LEMMA 3.1. *For unit disk graphs, the maximum independent set,(MIS), in the neighborhood, $N(u)$, of a vertex u has at most 5 vertices.*

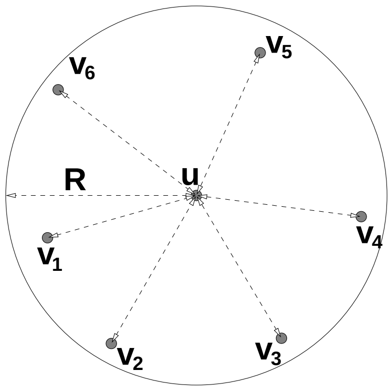

Figure 6: Every six neighbors of a vertex have at least one edge

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 as shown in Figure 6. 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. $j = i(mod 6) + 1$. 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 $\triangle u v_i v_j, (v_i, v_j)$ is the largest side, and so $\angle u$ is the largest angle, which must be $> \frac{\pi}{3}$. Hence, $\sum_{i=1}^{6} \angle v_i u v_j > 6 \times \frac{\pi}{3} = 2\pi$, 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, and only one cluster may have size $< k$.*

Proof. A vertex is removed from the tree T in lines 4,14 or 19 of `GraphCluster` or in line 7 of `MergePartialClusters`. In each such case, they are put in some cluster just prior to that. When the algorithm terminates, all vertices are either deleted, or are left in a single partial cluster, rooted at the root-vertex of the tree. These vertices form the only partial cluster as allowed in requirement (3b) of the problem statement.

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

Clusters created are either subtrees of T, or are sets of subtrees that have other non-tree graph edges to connect them.

OBSERVATION 3. *Any vertex u which satisfies the* **if** *condition in line 5 of* `GraphCluster`*, has all its present children in the tree, of size $< k$.*

This can be shown by induction on the post-order traversal of the tree. Whenever a subtree has size $\geq k$, clusters are formed out of it, and only a partial cluster of size $< k$, is left in tree, with all the other vertices being deleted.

LEMMA 3.2. *All clusters (except one) has size between k and $2k$.*

Proof. A cluster created in line 4 of `GraphCluster` has the required size bound.

From observation (3), an invariant pre-condition of line 10 of `GraphCluster` is : $|TempCluster| < k \wedge |T(x)| < k$. Hence, post-condition of line 10 is $|TempCluster| < 2k$. Hence, when processing exits the **while** loop (lines 9-11), $|TempCluster| < 2k$. The **if** condition on line 12 guarantees that the cluster added to the cluster set in line 13 has size between k and $2k$.

Each partial cluster in the partial cluster set has size $< k$. Hence, the invariant pre-condition of line 4 of `MergePartialClusters` is : $|C| < k \wedge |p| < k$. Hence, the post condition of line 4 of `MergePartialClusters` is $|C| \leq 2k - 2$. Hence, when a cluster is added in line 6 of `MergePartialClusters`, as $\{C \cup \{u\}\}$, its size is $\geq k$ and $< 2k$.

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

OBSERVATION 5. *Number of partial clusters, created on exiting the* **while** *loop of lines 7-16 of* `GraphCluster`*, is 5.*

Proof. Each partial cluster in the $PartialClusterSet$, has at least one child of u in the tree, since $TempCluster$ (line 9 of `GraphCluster`), has subtrees of u rooted at some children of u. If there are at least 6 partial clusters$P_1, \ldots, P_6$, then let $v_1 \ldots v_6 \in Children(u)$ be vertices in these 6 different partial clusters. A partial cluster is added in line 16 of `GraphCluster`, if the inner **while** loop was exited with the second condition on line 9 of `GraphCluster` is **false**. So, at the time P_i was put in $PartialClusterSet$ (line 16 of `GraphCluster`), there would have been no edge from any a vertex in P_i to any other v in $Children(u)$. In particular, $(v_i, v_j) \notin T$, i.e. $v_1 \ldots v_6$ form an independent set of vertices. This contradicts Lemma 3.1. Hence, there can be upto 5 partial clusters.

LEMMA 3.3. *Requirement 4 : A vertex v, is part of upto 3 distinct clusters.*

Proof. A vertex that is part of multiple clusters, would have to satisfy the **if** condition in line 5 of `GraphCluster`. It can be observed that all other vertices will be put into a single cluster. The vertex that satisfies the **if** condition in line 5 would not be placed in any cluster created on line 13 of`GraphCluster`. It will only be placed in the clusters created out of the partial clusters (Line 6 of `MergePartialClusters`). From observation 5, there are only 5 partial clusters. Each partial cluster has size $< k$. Hence, at least two such partial clusters will be merged in line 4 of `MergePartialClusters` to create a cluster in line 6. Hence, the maximum number of clusters created out of the 5 partial clusters, is 3 (upto two clusters of size between k and $2k$, and upto one cluster with size $< k$, that is made into some cluster higher up the tree, or left as is, if u is the root-vertex). Consequently, the vertex u can be a part of upto 3 clusters.

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

LEMMA 3.4. *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 (7). (Note that the constant, c is chosen to be $\sqrt{3}$ 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

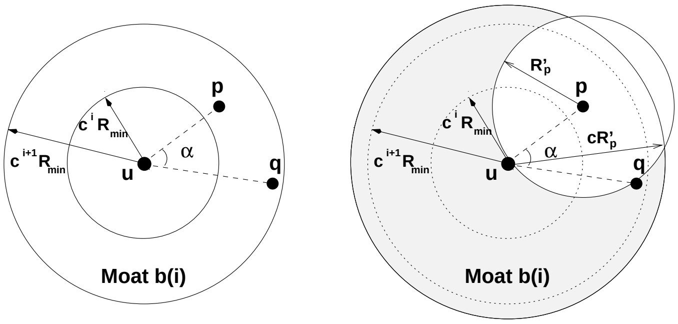

Figure 7: Two vertices p and q in the neighborhood, $N_i(u)$ of u that are part of the maximum independent set in the subgraph induced by $N_i(u) \cup \{u\}$ have an angular separation, $\alpha > \frac{\pi}{6}$. They belong to the same moat $b(i)$, and there are $\log(\frac{R_{\max}}{R_{\min}})$ such moats.

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 7). 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 3.1, there cannot be more than 11 vertices in the moat $b(i)$.

As a consequence of Lemma 3.4, 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.

3.4 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 $O(|E|)$.

4 Distributed Implementation

The algorithm that is described in Section 3.2, is a centralized solution to the problem. In this section, we present a distributed implementation of our clustering technique, using the centralized algorithm for a set of wireless nodes. This implementation can be split into two different parts.

4.1 Cluster Formation: This is a simple distributed extension of the centralized `GraphCluster` algorithm, and is run initially to produce a set of clusters that are connected and have the specified size requirements. It needs to be run infrequently, as described in Section 4.2, to reform the clusters.

For this, we partition the clustering algorithm into two layers, as shown in Figure 8 :

- Tree discovery protocol (TDP) : This is a simple lower layer that is responsible for creating and maintaining the tree T, described in line 1 of the `GraphCluster` procedure. Any simple tree discovery protocol can be used here. Any node can be chosen as the root of the tree, a simple choice being the node with the lowest identifier. Each node stores its distance from the root in number of hops, and transmits a `root-distance`

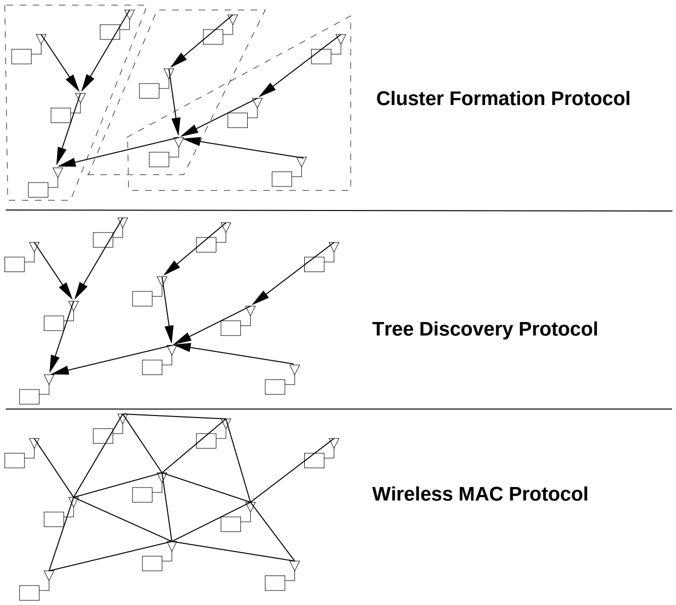

Figure 8: The two-layered architecture : The tree discovery protocol is layered above the wireless MAC protocol (e.g. 802.11) and the cluster formation protocol, is layered above the tree discovery protocol

beacon, it to its neighbors, either periodically, or as changes in the parent information are discovered. On receiving a `root-distance` beacon, a node will choose the source of the beacon with the least distance to the root, as its parent. Changes, are propagated down the tree, as discovered. This basic tree discovery protocol is robust across node mobility, as new nodes appear or existing nodes disappear (which is detected by timeouts at parent and children in the tree) in the topology.

- Cluster formation protocol (CFP) : This protocol uses the abstraction provided by TDP, and runs the distributed computation of the clusters at the different wireless nodes. Each node, initially, forms a single trivial cluster with itself as the only member. Each node reports its partial cluster information to its parent upstream in the tree. Recall, that a partial cluster is a cluster of size less than k, and will be composed of some of its subtrees (as reported by its children). Children never include in their subtree information to their parent, those nodes that have already been placed in clusters that meet the size requirements. A node will either merge all its subtrees of partial clusters, as advertised by its children, to form clusters that meet the size requirements, or will be left with a single partial cluster, which is propagated further up the tree (this is analogous to the processing of lines 4-19 of the `GraphCluster` procedure). At the end of this distributed processing, the set of desired clusters will be created.

4.2 Cluster Maintenance: Clearly, it is expensive to run the cluster formation mechanism for every change in the wireless node topology. To handle new sensor nodes joining the network, existing nodes, migrating or leaving the network (their battery might run out), or link outages happening due to increased channel errors, we propose a simple incremental mechanism to maintain the set of clusters, without significantly perturbing the set of desirable requirements for the set of clusters.

New node joins: A sensor node, v, on joining the network, has to first establish the set the neighbors, $N(v)$, that it can communicate with. If any node $u \in N(v)$ belongs to some cluster of size $< 2k-1$ 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.

If each vertex in $N(v)$ belongs to a cluster of size at least $2k-1$, but has size $< 3k-1$ we add v to that neighbor's cluster, thus relaxing our upper bound for the cluster size to $3k-1$. 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, $3k$. Since this cluster is connected, we re-run the distributed implementation of the `GraphCluster` procedure, *only on the nodes in this cluster*, and thus would create two new clusters of sizes between k and $2k$. As can be observed, in the worst case, only one in every k such node insertions, in that neighborhood, would cause a localized re-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 3.1 and 3.4), 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 $< k$, we can run an insertion algorithm into its neighboring clusters. In this scheme, the component joins its smallest neighboring cluster. If the resulting cluster size exceeds, $3k$, then the distributed implementation of `GraphCluster` procedure is run *only on nodes in the resulting cluster*. This would create all clusters of size between k and $2k$, except maybe one cluster of size less than k. However, this cluster, may now be added to one of the other created clusters so that the combined size is within $3k$.

Link outage: A link outage does not alter the number of nodes in the cluster. It may split the cluster into two parts, and the same mechanism used to repair cluster partition when a node leaves, can be applied here.

Re-running cluster formation: As a consequence of the cluster maintenance mechanism, in the worst case, some nodes may belong to increasing number of clusters. To limit this effect, a global measure, like, $\sum |S(v)|$, where, $S(v)$ is the set of clusters, v belongs to, can be periodically estimated, and used to decide the frequency with which the cluster formation mechanism is re-run.

5 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.

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 (9), 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.

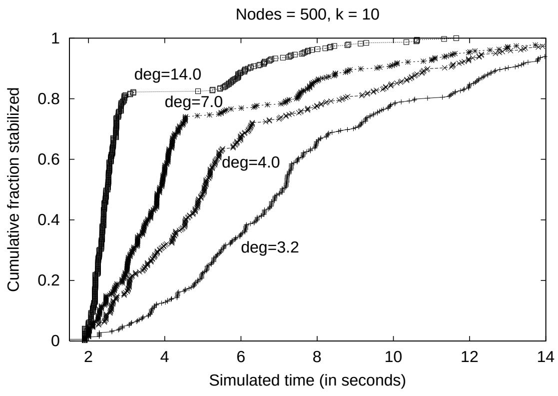

Figure 9: Cumulative fraction of nodes stabilized in a wireless network with simulated time elapsed, for varying connectivity of the topology

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

Figure 10: Cluster diameter for varying connectivity of nodes in the topology and varying k

Cluster diameter: In general, it is desirable to have clusters of low diameter. In the Figure 10, 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 upto 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.

6 Related Work

Some routing solutions for the Internet have used hierarchies to provide scalability, e.g. OSPF protocol [Mo 97] and ATM PNNI [ATM 96], have mechanisms to perform hierarchical routing. All hierarchy based protocols leverage the fact that by aggregating addresses, routers can reduce the size of their routing tables and other routing related data structures. Additionally, in some cases, e.g. ATM PNNI [ATM 96], [Le 95] clustering is used to split 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. The PNNI standard recommends the representation of a peer-group by a star graph, with one virtual central node (nucleus) and weighted spokes between the nucleus and the peer-group's border nodes. Previous work by Awerbuch et al [Aw 98] has focussed on the performance of different topology aggregation schemes for networks, and they conclude that the Minimum Spanning Tree is a good candidate to be the representative of a cluster. However, in most such Internet routing protocols and clustering schemes described, clusters and hierarchies are created by explicit configuration of the routers.

Recent work by Chamlee and Zegura [ChZe 98] proposed methods to perform dynamic hierarchical address assignment for a network. A main concern of their approach has been to provide route aggregation through configuring the network with hierarchical addresses. Krishnan et al [Kr 99] 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) [Ha 99], 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 [Ba 81] and [Ba 82]. In [Ge 95], [Li 97] 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) [Ji 99] and a more generalized approach is used in [Ba 97] for mobile ad-hoc networks. Das et al [Da 97] uses a connected dominating set to create a routing 'spine' and describes a clustering scheme to create two layered hierarchies. Krishna et al [Kr 95] 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.

7 Conclusions

In this paper, we have presented a clustering scheme for wireless networks that can be used for efficiently implementing hierarchical routing. The clustering problem that we solve, is intractable for arbitrary graphs. However, we exploit some geometric properties of the wireless networks to meet our desired requirements. In fact this illustrates that modeling problems arising in the context of mobile communication as arbitrary graphs, loses a lot of information about the special nature of these graphs. Even without knowledge of the precise placement of the nodes, there are many properties we can use, to design good algorithms. For example, our modeling the problem as an abstract graph problem leads to a situation where in fact no solution may exist. However, special properties of these graphs can be exploited to find a solution.

References

[802.11] 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.

[IGRP] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/igrp.htm

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

[Kl 77] L. Kleinrock and K. Faroukh. Hierarchical Routing for Large Networks. Computer Networks, Vol. 1, 1997.

[Ba 81] D.J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. IEEE Transactions on Communications. November 1981.

[Ba 82] 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.

[He 88] C.L. Hendrick. Routing Information Protocol, RFC 1058, June 1988.

[Cl 90] B.N. Clark, C.J. Colbourn and D.S. Johnson. Unit Disk Graphs, Discrete Mathematics, 1990.

[Ka 90] P. Karn. MACA - a new channel access method for packet radio. ARRL/CRRL Amateur Radio 9th Computer Networking Conference, 1990.

[Va 94] V Bhargavan, A. Demers, S. Shenker and L. Zhang. MACAW : A Media Access Protocol for Wireless LANs. Proceedings of Sigcomm, September 1994.

[Ge 95] M. Gerla and J.T.-C. Tsai. Multicluster, mobile, multimedia radio network. ACM-Baltzer Journal of Wireless Networks, Vol 1. No. 3, 1995.

[Kr 95] 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.

[Le 95] W.C. Lee. Topology Aggregation for Hierarchical Networks. ACM Computer Communications Review, Vol 25, No. 1, 1995.

[ReLi 95] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4), RFC 1771, March 1995.

[ATM 96] ATM Forum Technical Committee. Private Network-Network Interface, Version 1.0, May 1996.

[Jo 96] D.B. Johnson, D.A.Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks, Mobile Computing, 1996.

[Ba 97] S. Basagni, I. Chlamtac, and A. Farago. A generalized clustering algorithm for peer-to-peer networks. Workshop on Algorithmic Aspects of Communication July 1997.

[Da 97] B. Das, R. Sivakumar, and V. Bharghavan. Routing in Ad-Hoc Networks Using a Spine. International Conference on Computers and Communications Networks, September 1997.

[Fu 97] C.L. Fullmer and J.J. Garcia-Luna-Aceves. Solutions to Hidden Terminal Problems in Wireless Networks. Proceedings of Sigcomm, September 1997.

[Li 97] C.R. Lin and M. Gerla. Adaptive clustering for mobile, wireless networks. Journal on Selected Areas of Communication, 15, 7, September 1997.

[Mo 97] J. Moy. Open Shortest Path First Version 2, RFC 2178, July 1997.

[Pe 97] C.E. Perkins. Ad-hoc On-demand Distance Vector Routing. Internet-Draft, draft-ietf-manet-aodv-00.txt, November 1997.

[Aw 98] B. Awerbuch, Y. Du, B. Khan and Y. Shavitt. Routing Through Networks with Hierarchical Topology Aggregation, Journal of High Speed Networks, 7(1), 1998.

[Be 98] Z. Xu, S. Dai, and J.J. Garcia-Luna-Aceves. Hierarchical Routing Using Link Vectors. Proceedings of Infocom, March 1998.

[ChZe 98] M.E. Chamlee and E.W. Zegura. Clustering Algorithms for Multi-level Address Hierarchies, Proceedings of the International Conference on Computer Communications and Networks, October 1998.

[Hu 98] 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.

[Ra 98] 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.

[Es 99] D. Estrin, R. Govindan, J. Heidemann and S. Kumar. Next Century Challenges : Scalable Coordination in Sensor Networks.

[Ga 99] J.J. Garcia-Luna-Aceves and A. Tzamaloukas. Reversing the Collision-Avoidance Handshake in Wireless Networks. Proceedings of Mobicom, August 1999.

[Gu 99] E. Guttman, C. Perkins J. Veizades and M. Day Service Location Protocol, Version 2, RFC 2608, June 1999.

[Ha 99] Z.J. Haas and M.R. Pearlman. The Zone Routing Protocol for Ad Hoc Networks, Internet-Draft, draft-ietf-manet-zone-zrp-02.txt.

[Ji 99] M. Jiang, J. Li and Y.C. Tay. Cluster-Based Routing Protocol (CBRP), Internet-Draft, draft-ietf-manet-zone-zrp-02.txt, August 1999.

[Ka 99] J.M. Kahn, R.H. Katz and K.S.j. Pister. Next Century Challenges : Mobile Networking for "Smart Dust", Proceedings of Mobicom, August 1999.

[Kr 99] R. Krishnan, R. Ramanathan and M. Steenstrup. Optimization algorithms for Large Self-structuring Networks, Proceedings of IEEE Infocom, March 1999.

[PaCo 99] V. Park. S. Corson. Temporally Ordered Routing Algorithm Version 1, Functional Specification. Internet-Draft, draft-ietf-manet-tora-spec-02.txt, October, 1999.

[Pe 00] C.E. Perkins, editor. IP Mobility Support for IPv4, revised. Internet Draft, draft-ietf-mobileip-rfc2002-bis-01.txt, January 2000.