K-Clustering in Wireless Ad Hoc Networks

Yaacov Fernandess
School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel
fery@cs.huji.ac.il

Dahlia Malkhi
School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel
dalia@cs.huji.ac.il

ABSTRACT

Ad hoc networks consist of wireless hosts that communicate with each other in the absence of a fixed infrastructure. Clustering is commonly used in order to limit the amount of routing information stored and maintained at individual hosts. A k-clustering is a framework in which the wireless network is divided into non-overlapping sub networks, also referred to as clusters, and where every two wireless hosts in a sub network are at most k hops from each other. The algorithmic complexity of k-clustering is known to be NP-Complete for simple undirected graphs. For the special family of graphs that represent ad hoc wireless networks, modeled as unit disk graphs, we introduce a two phase distributed polynomial time and message complexity approximation solution with $O(k)$ worst case ratio over the optimal solution. The first phase constructs a spanning tree of the network and the second phase then partitions the spanning tree into subtrees with bounded diameters.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS; C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms

Keywords

wireless ad hoc networks, distributed algorithm, k-clustering, connected dominating set, competitive ratio

1. INTRODUCTION

Ad hoc networks offer the vision of true ubiquitous computing, providing connectivity to everyone, anywhere and from any device. Due to the inherent scale and dynamism of this vision, technical challenges in supporting ad hoc networking are demanding and often daunting. One of the successful approaches for tackling the maintenance of these networks is by decomposing the network into clusters: Since clusters are manageable, intra-cluster maintenance is done tightly, while inter-cluster connectivity is reduced by orders of magnitude this way. In this paper, we enhance existing clustering technology with the first competitive, distributed algorithm for partitioning an ad hoc network into limited-diameter clusters (k-clustering). Our algorithm approximates the best k-clustering partitioning of a graph that represents a mobile communication network. More specifically, if one makes the assumption that each mobile device has a known range of broadcast interception, then, using the well-known *unit disk graph* model, we can show that our algorithm approximates the lowest number k-clustering with a competitive ratio of $O(k)$.

The motivation for our work is the need to maintain routing strategies in the demanding settings of ad hoc networks. Ad hoc networks consist of mobile hosts which can communicate with each other over multi-hop wireless paths without any static network infrastructure. As such, ad hoc networks are characterized by dynamic topology, multi-hop communication and certain strict limitations such as high power consumption, low bandwidth and high error rates. Consequently, routing in ad hoc networks that consist of large numbers of nodes is a challenging problem and much research has been done on the subject. Existing routing protocols can be divided into three categories: proactive, reactive and a hybrid of the two. Proactive routing protocols attempt to maintain consistent, up-to-date information at every node about the route needed to reach every other node in the network. Such routing protocols respond to changes in network topology by propagating route updates throughout the network. Reactive routing protocols use source-initiated on-demand routing. When a source node requires a route to a destination node, it initiates a route discovery process within the network. Once the route has been discovered, it is maintained until the destination becomes inaccessible or the route is no longer required. No conclusions about the definite advantages of either of the categories have yet been made and any such conclusions are likely to be effected by conditions of network mobility.

The third routing category is a hybrid of the previous two. It maintains only partial topology information of the network. One example of this type of routing is a cluster-based routing protocol, proposed in [15]. In a cluster-based routing protocol, the network is divided into non overlapping

POMC'02, October 30-31, 2002, Toulouse, France.

subnetworks, also referred to as clusters. When a source node wishes to send a packet to another network node that is not in the same cluster, the node uses a reactive routing protocol in order to discover the route. One advantage of this is that the route discovery time can be reduced by flooding the discovery packet to the cluster's border hosts only. However, within the confines of the cluster itself, a proactive routing protocol is used since cluster connectivity is maintained by periodically exchanging updates about any changing link information among neighbor nodes.

Until now, clustering algorithms for ad hoc networks concentrated on a single instance of the k-clustering problem, namely finding a clique partition (1-clustering partition). Our objective is to allow more flexibility by choosing k as a dynamic parameter of the network. Note that the same flexibility is achieved by a distributed algorithm for a somewhat similar cluster topology problem, k-dominating set, that was introduced in [4]. In keeping with this goal, we introduce a distributed asynchronous algorithm for k-clustering in a wireless ad hoc network. In essence, we investigate the following partitioning problem.

Minimum k-clustering: Given an unit disk graph denoted by $G = (V, E)$ and a positive integer k, find the smallest value of ℓ such that there is a partition of V into ℓ disjoint subsets $V_1, ..., V_\ell$ and $diam(G[V_i]) \leq k$ for $i = 1..\ell$. Let $pk(G)$ denote the minimum order of a k-clustering partition for G.

The algorithmic complexity of k-clustering is known to be NP-Complete for simple undirected graphs [8], and no general approximation of it is known. Moreover, no reduction of the difficulty, nor any approximation, is known for the restricted case at hand, i.e., for unit disk graphs. This paper presents the first distributed asynchronous algorithm for k-clustering that has a competitive ratio of $O(k)$ for wireless ad hoc networks.

Our solution consists of two phases. The first phase constructs a spanning tree. Although any spanning tree would suffice for finding a correct k-clustering, for competitive performance we choose to employ a special algorithm for the spanning tree. The second phase partitions the spanning tree into subtrees whose diameter is k bounded, yielding a k-clustering of the network.

Regarding the first phase, we observe that generally, spanning trees might yield a bad k-clustering for the graph. That is, the optimal k-clustering for a particular spanning tree may have far more partitions than the optimal k-clustering partition of the graph. Fortunately, in the case of unit disk graphs, we can make use of the Minimum Connected Dominating Set (MCDS) technique for the creation of a spanning tree, which was recently developed [14]. For a spanning tree generated by MCDS, the k-subtree partition has a worst case ratio of $O(k)$ over the best k-clustering of the graph. This competitive ratio may be derived in the following way. The MCDS used here has the property of containing a maximal independent set IS, such that every path of length ℓ in the tree contains at least $\ell/2$ members of IS. It should then be noted that in a unit disk graph, independence of nodes implies that every node is at a distance greater than 2 from every other node.

Let us look at the second phase. This phase solves the k-clustering problem for trees, a restricted class of graphs. This version of the problem is known as the *minimum k-subtree partitioning* problem. It has been addressed in [9], and has a simple, linear time solution. In a nutshell, the solution works by finding a minimal subtree whose diameter exceeds k, where minimality means that no child subtree already has a diameter that is larger than k. It then detaches the highest child of the subtree and repeats over. We provide a straightforward distributed asynchronous implementation of this strategy, which works in a linear number of messages and has a time proportional to the tree height.

Putting these facts together, we are able to show that each partition formed by the k-subtree partitioning takes up certain geometric space, limiting the number of partitions that can exist as compared with the space taken by an optimal k-clustering partitioning. The result our algorithm achieves is a competitive ratio of $O(k)$ over the best k-clustering partition.

The paper is organized as follows. System model, definitions and preliminaries of graph theory are given in section 2. Section 3 describes a distributed asynchronous algorithm for the k-clustering problem and proves its performance. Section 4 discusses extensions. Section 5 reviews related work and the conclusion is given in Section 6.

2. SYSTEM MODEL AND PRELIMINARIES

A wireless ad hoc network can be modeled as an undirected graph $G = (V, E)$ in which V, $|V| = n$, is the set of nodes and there is an edge (u, v) in E, $|E| = m$, if and only if u and v can mutually receive each other's transmission. Although wireless ad hoc networks are modeled using a special class of graphs, *unit disk graphs*, this section will first focus on some general definitions related to the k-clustering problem on simple undirected graphs, which will then be followed by a more detailed description of unit disk graphs.

Given an undirected graph $G = (V, E)$ the distance between two nodes, u and v, denoted by $d(u, v)$, is the number of edges that together form the shortest path between those nodes. The diameter of G is defined by $diam(G) = max\{d(u, v) : u, v \in V\}$. For any non-empty subset C of V, $G[C]$ denotes the subgraph of G induced by C. A tree, denoted by T, is an acyclic connected graph. The tree $T(u)$ is the subtree of T rooted at u. The height of T, denoted by $h(T)$, is the length of the longest path from the root to a leaf. For a leaf node x, of a tree T, we have $h(T(x)) = diam(T(x)) = 0$.

In [7], Johnson, Clark and Colbourn describe three canonical definitions for modeling a unit disk graph, one of which is called the *intersection model*. In the *intersection model* a set of n unit disks in the plane is represented by an n-node graph, where every node corresponds to a unit disk and there is an edge between two nodes if the corresponding unit disks intersect or tangent.

Our system model includes two general assumptions regarding the state of the network's communication links and topology. The first assumption is that the network topology remains unchanged throughout the execution of the k-clustering algorithm. The second assumption is that the network may be modeled using a unit disk graph, even though communication between network nodes is, in actuality, a dynamic function involving a number of factors such as interference and signal propagation conditions. An ad hoc network can be viewed as a unit disk graph by viewing every

transmitter/receiver in the broadcast network as a point in the graph and by representing the effective broadcast range of each point as a unit disk.

3. ALGORITHM

In this section, we describe the approximation algorithm for the k-clustering partitioning problem. We begin the exposition in Section 3.1 with a particular spanning tree algorithm. The way in which the spanning tree is constructed affects the goodness of the partition induced by the k-subtree partition, as compared with the best k-clustering partition of the original graph. In Section 3.3 we prove that the spanning tree we choose produces a clustering whose order is within factor $O(k)$ of the optimal.

Our exposition then outlines the k-subtree algorithm in Section 3.2. This algorithm finds the optimal partition of a given tree. Note that once a spanning tree is fixed in our graph, there will be but few choices for optimal k-subtree partitions. Our algorithm finds one in a distributed manner with linear complexity.

3.1 The spanning tree algorithm

Any of various known algorithms may be used in order to construct a spanning tree in the first phase of the k-clustering algorithm and achieve a correct k-clustering partition. However, the spanning tree strategy may affect the quality of the clustering substantially. To shed some light on this, consider the following two extreme examples.

BFS spanning ree: Many applications use a Breadth First Search (BFS) tree in order to develop efficient communication between nodes in a network. Unfortunately, using BFS in order to build the spanning tree in the k-clustering algorithm can have a worst case ratio of $n/diam(G)$. Consider a unit disk graph $G = (V, E)$ which corresponds to the following set of n points in the plane:

$S = \{r = (0.5, 1)\} \bigcup \{\bigcup_{j=1}^{k} \bigcup_{i=1}^{f} \{(i/f, 2j)\}\}$

where $n - 1 = fk$. The optimal k-clustering of G has exactly one cluster. However, consider the BFS spanning tree T for G rooted at r. The order of the optimal k-clustering on T is $f+1$. Therefore, the ratio $pk(T)/pk(G) \leq n/diam(G)$.

DFS spanning tree: Using DFS in order to build the spanning tree can have catastrophic results. Consider using the DFS spanning tree on a clique. The resulting spanning tree could be a simple path with a worst-case ratio of n/k.

The approach we have chosen to adopt makes use of a recent algorithm for constructing a Minimum Connected Dominating Set (MCDS) [14].

Preliminaries.. We begin with some notation. A *Dominating Set* (DS) of a graph $G = (V, E)$ is a subset $D \subseteq V$ such that each node in $V \setminus D$ is adjacent to some node in D. If the induced subgraph $G[D]$ is connected, then D is a *Connected Dominating Set* (CDS). The problem of finding a minimal CDS is called the *Minimum Connected Dominating Set* (MCDS) problem. One can construct a spanning tree using D as the 'spine' of the tree and having the nodes in $V \setminus D$ as the leaves of the tree. An *Independent Set* (IS) of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that there is no edge between any pair of nodes in S. An independent set S is maximal if no proper superset of S is also an independent set. It is easy to see that for any graph a maximal independent set is also a dominating set.

MCDS Algorithm. The algorithmic complexity of MCDS is known to be NP-hard for unit disk graphs according to [7]. Therefore, several approximation algorithms have been proposed for MCDS. For example, in [14], Breu et al. present some simple heuristics for a number of classical NP-hard optimization problems on unit disk graphs, including the MCDS problem. The heuristics do not require a geometric representation of a unit disk graph as part of the input. Geometric representations are used only for establishing certain properties of unit disk graphs. These properties, in turn, are used in order to derive the performance guarantees provided by the heuristics.

The sequential approximation algorithm for MCDS presented in [14] has a constant worst case ratio of 10. Given a unit disk graph $G = (V, E)$, the algorithm first constructs a BFS spanning tree T, rooted at a given arbitrary node r. The tree T partitions the nodes of G into disjoint sets S_i, $0 \leq i \leq h(T)$, where S_i is the set of nodes in the tree at height i. The connected dominating set produced by the algorithm is the union of the following two sets of nodes. The first set is a maximal independent set for G, obtained by appropriately selecting an independent set IS_i from each graph $G[S_i]$, $1 \leq i \leq h(T)$. More precisely, after choosing IS_{i-1}, all dominated nodes in S_i are removed, and IS_i is chosen from the remaining nodes. In this way, $\bigcup_{i=0}^{\ell} IS_i$ forms a maximal independent set IS. The second set of nodes is used to ensure connectivity of IS, by connecting members of IS_i with members of S_{i-1}, $1 \leq i \leq \ell$. Note that in this way, each edge is incident to one member of IS. For completeness, Appendix A provides a pseudo code description of the MCDS sequential algorithm.

We use the distributed implementation of MCDS, provided in [2], which has a time complexity of $O(n)$ and a message complexity of $O(n \log n)$. The main property derived from the MCDS based spanning tree construction is the following:

PROPERTY 3.1. *Let T be a spanning tree constructed by the MCDS algorithm above. Then there exists a maximal independent set $IS \subseteq V$, such that every path in T of length ℓ contains at least $\ell/2$ members of IS.*

3.2 The k-subtree algorithm

Preliminaries. We begin the exposition by noting two key properties of trees that facilitate decomposition into bounded diameter k-subtrees. The first one is a simple statement regarding the calculation of the tree diameter.

LEMMA 3.1. *Consider a tree T rooted at a given, arbitrary node r. Let S be a set of children of r. Then the following holds:*

(i) *If S is empty, then $diam(T) = h(T) = 0$.*

(ii) *If $|S| = 1$ and let $S = \{v\}$, then $diam(T) = \max\{diam(T(v)), h(T(v)) + 1\}$.*

(iii) *If* $|S| > 1$ *then* $diam(T) = \max\{\max\{diam(T(v)) : v \in S\}, \max\{h(T(v)) + h(T(u)) + 2 : u, v \in S, u \neq v\}\}$.

PROOF.

(i) This holds by definition.

(ii) For any pair of nodes x and y in T, if x and y belong to $T(v)$, it follows that $d(x,y) \leq diam(T(v))$. Otherwise, either x or y is r. Without loss of generality, assume $y = r$. Then $d(x,r) = d(x,v) + d(v,r) \leq h(T(v)) + 1$.

(iii) For any pair of nodes x and y in T, if x and y belong to any single subtree $T(u)$, it follows that $d(x,y) \leq diam(T(u) \leq \max\{diam(T(v) : v \in S\}$.

If y is r and x belongs to $T(v)$, then $d(x,y) = d(x,v) + d(v,r) \leq h(T(v)) + 1$.

Otherwise, let x belong to $T(rootx)$ and y belong to $T(rooty)$. Therefore,

$$\begin{aligned} d(x,y) &= d(x, rootx) + 2 + d(y, rooty) \\ &\leq h(T(rootx)) + 2 + h(T(rooty)) \end{aligned}$$

□

The second lemma is taken from [9] and we provide the proof for the sake of completeness. The algorithm presented below is a distributed implementation of this lemma and uses a divide and conquer approach to arrive at a k-subtree partitioning of a tree.

LEMMA 3.2. *[9] Consider a tree T rooted at a given, arbitrary node r. Let u be a node in T, such that $u \neq r$ and $diam(T(u)) > k$. Let S be any set of children of u where $\forall s \in S : diam(T(s)) \leq k$. Let t be a node in S such that $\forall s \in S : h(T(t)) \geq h(T(s))$. Then $pk(T) = 1 + pk(T \setminus T(t))$.*

PROOF. Clearly, a k-subtree partition of $(T(t), T \setminus T(t))$ is a valid k-subtree partition of T. Hence, $pk(T) \leq 1 + pk(T \setminus T(t))$. We now show the reverse.

Let $P(u)$ denote the cluster that u belongs to in an optimal k-subtree partition of T. From $diam(T(u)) > k$, it follows that $T(u)$ cannot be contained in its entirety in a single subtree of a k-subtree partition. Hence, $P(u) \not\supseteq T(u)$, and some subtree $T(s) \subseteq T(u)$ must be a partition by itself, where s is a child of u (smaller sub-trees are suboptimal, since by assumption, $diam(T(s)) \leq k$ for all $s \in S$). We claim that we can swap $T(s)$ with $T(t)$. Indeed, by Lemma 3.1 the diameter of $P(u)$ can only decrease. Therefore, the new $P(u)$ (including $T(s)$, excluding $T(t)$) is a valid cluster of diameter up to k. The rest of the tree is not affected by this change. As result, we conclude the following: $pk(T) = 1 + pk(T \setminus T(s)) \geq 1 + pk(T \setminus T(t))$. □

The algorithm. The second phase of the k-clustering algorithm, the k-subtree partition, is a distributed asynchronous algorithm that partitions the spanning tree into subtrees of bounded diameter using a convergecast procedure. Figure 1 depicts the code of each node.

The algorithm executed by each node is as follows. Every node's input is its maximum height, which is initially equal to zero. Beginning with the leaves, each leaf sends its height (zero) to its parent. Each parent node p acts as follows. Every node maintains its own height in the variable *height* and the identifier of the child with maximal height in variable *highest_child*. When it receives a new height h from one of its children, it checks whether $h + 1 + height$ exceeds k. If it does, then the subtree rooted at the node has a diameter that is greater than k (by Lemma 3.1). Hence, the node partitions the subtree according to the rule specified in Lemma 3.2. That is, it instructs the child whose height is the largest (*highest_child*) to detach. Otherwise, both *height* and *highest_child* are updated. After receiving the heights of all its children, p sends its own height to its parent.

Correctness. We now argue the correctness of the k-subtree partitioning algorithm above.

THEOREM 3.3. *The k-subtree partitioning algorithm achieves the following: (i) It partitions a tree into subtrees of diameter at most k, and (ii) it obtains the minimum order of such a partition.*

PROOF. For part (i), let $T(\ell)$ be a detached subtree rooted at node ℓ. According to lines 2 and 3 in the algorithm of Figure 1, the execution of the algorithm on node ℓ obtains the following: $height(T(\ell)) \leq k$, and $diam(T(\ell)) \leq k$.

Moving to (ii), we will use an induction on the order of $pk(T)$ in order to prove that the algorithm obtains the optimal value of $pk(T)$. Let R be a set of nodes that become roots of the subtrees of a partition resulting from the execution of the algorithm. The size of this set – $|R|$ – is the order of the algorithm partition. For $pk(T) = 1$ it follows that $diam(T) \leq k$. The algorithm terminates with $R = \{\text{root of } T\}$, since the diameter is never exceeded in line 2 of the code.

Our inductive hypothesis is that the algorithm obtains $pk(T) = |R|$ for $pk(T) < n$. Now consider $pk(T) = n$ and consider the first node ℓ that sets its root state to *true*. Since detaching is done according to the conditions of Lemma 3.2, It follows that $pk(T) = 1 + pk(T \setminus T(\ell))$. Since $pk(T \setminus T(\ell)) < n$, the inductive hypothesis holds and the algorithm obtains a global optimization. □

Complexity. The number of messages sent during an execution of the second phase of the k-subtree partition algorithm is at most $2n$, where n is the number of nodes in the network. This stems from the fact that each edge in the spanning tree is used for sending at most two messages (child to parent and parent to child) and that the number of edges in a tree is at most $n - 1$. The time until all the messages reach the root node is equal to $h(T) \leq n$, since this is the longest path from a leaf in the spanning tree to the root node.

3.3 Solution Analysis

In this section, we analyze the quality of the k-clustering algorithm that combines the MCDS spanning tree with the k-subtree partitioning. We first state and prove the main result: The order of the k-clustering partition of our algorithm approximates the optimal k-clustering solution to within a factor of $O(k)$.

THEOREM 3.4. *Given a unit disk graph, the distributed algorithm for k-clustering using the MCDS spanning tree specified above has a partition order worst case ratio of $4k + o(1)$ over the optimal.*

Input:

node id cn.
$receive(h)_i$, h is a positive integer and $Parent[i] = cn$.
$receive(r)_i$, r is a boolean and $Parent[cn] = i$.

Output:

$send(h)_j$, h is a positive integer and $Parent[cn] = j$.
$send(r)_j$, r is a boolean and $Parent[j] = cn$.

State:

height is a positive integer, initially 0.
highest_child, $Parent[highest\_child] = cn$, initially null.
root is a boolean, initially false except for the root of the spanning tree.

Transitions:

$receive(h)_i$

Effect:

```
[1]  h++;
[2]  if ((h + height) > k) {
         ; must detach some child
[3]      if (h > height)
             ; i is the highest child
[4]          send(true)_i;
[5]      else {
             ; detach the highest recorded child
[6]          send(true)_highest_child;
[7]          height ← h;
[8]          highest_child ← i;
[9]      }
[10] }
[11] else if (h > height) {
         ; record i as highest child
[12]     height ← h;
[13]     highest_child ← i;
[14] }
```

$send(h)_i$

Precondition:

Current node received messages from all its children. Immediately satisfied in a leaf node.

$receive(r)_i$

Effect:

if $(r = true)$ $root \leftarrow true$;

Figure 1: Pseudo code of k-subtree algorithm.

PROOF. Let *opt* denote the order of the optimal k-clustering on G and let $\{opt_i\}_{i=1..opt}$ denote the clusters in it. Let *approx* denote the order of our k-clustering approximation on G and let $\{approx_i\}_{i=1..approx}$ denote its clusters. The following lemmas will help us prove the above theorem.

LEMMA 3.5. *Every cluster opt_i, $1 \leq i \leq opt$, is bounded by a $2k \times 2k$-square.*

PROOF. Consider the path between the node with the smallest x coordinate and the node with the largest x coordinate. The length of the path is at most k. Therefore, the distance between the two points is at most $2k$. The case of the y coordinate is symmetric and the same logic applies. □

LEMMA 3.6. *Every cluster $approx_i$, $1 \leq i \leq approx$, except possibly one, includes at least $\lceil \frac{k}{4} \rceil$ independent members of IS.*

PROOF. It is easy to see from lines 2 and 3 that the k-subtree partition algorithm obtains clusters with *diam* $(approx_i) \geq \lceil \frac{k}{2} \rceil$, with the exception of the cluster rooted at the root of the spanning tree. Let ρ be the longest path in the cluster $approx_i$, with a length of at least $\lceil \frac{k}{2} \rceil$. It follows from the construction of MCDS that half of the nodes in ρ, i.e., at least $\lceil \frac{k}{4} \rceil$, are part of the maximal independent set. □

We are now ready to complete the theorem's proof. Without lose of generality, assume that the smallest cluster in the approximate solution is $approx_1$, and, as such, we omit it from the calculation. The proof is by contradiction. More specifically, we show that there exists an opt_i that contains more than k^2 elements of IS, which contradicts Lemma 3.5 – a $(2k \times 2k)$-square cannot contain more than k^2 disjoint unit disks. To show that such an opt_i exists, note that the nodes in IS are partitioned according to $\{opt_i\}$ and likewise they are partitioned differently according to $\{approx_i\}$. There are two cases. The first case is that there are more IS members in some $\{opt_i\}$ than some $approx_j \neq approx_1$ by a factor of $4k$. In this case, opt_i contains more than $4k(k/4) = k^2$ members of IS, and the contradiction follows. The second case is that there is no such opt_i, and it follows that $approx - 1 \leq (4k)opt$. In this case, the competitive ratio holds.

More specifically, let us denote by $a_i = |approx_i \cap IS|, i = 1..approx$. Assume w.l.o.g. that $a_1 = \min_{i=1..approx} a_i$, and that $a_2 = \min_{i=2..approx} a_i$. Similarly, denote by $o_i = |opt_i \cap IS|, i = 1..opt$, and let $o_{max} = \max_{i=1..opt} o_i$. We have that $|IS| = \sum_{i=1}^{approx} a_i = \sum_{i=1}^{opt} o_i$. We assume by contradiction $approx > (4k)opt + 1$. We now have two cases. (i) The first case is that $o_{max} > (4k)a_2$. Hence, by Lemma 3.6, $o_{max} > (4k)(k/4) = k^2$. But as we already noted, this contradicts Lemma 3.5 – a $(2k \times 2k)$-square cannot contain more than k^2 disjoint unit disks.

(ii) The second case is $o_{max} \leq (4k)a_2$. In this case we have:

$$\begin{aligned} |IS| &= \sum_{i=1}^{approx} a_i \geq a_1 + a_2(approx - 1) \\ &\geq a_1 + (approx - 1)o_{max}/(4k) \\ &> a_1 + opt \times o_{max} \\ &\geq a_1 + \sum_{i=1}^{opt} o_i \geq |IS| . \end{aligned}$$

A contradiction.

The complexity of the algorithm is dominated by the complexity of the spanning tree algorithm. Therefore, we have the following:

THEOREM 3.7. *The message complexity of the k-clustering algorithm using the MCDS spanning tree building block is $O(n \log n)$, and its time complexity is $O(n)$.*

4. EXTENSIONS

4.1 MST Spanning Tree [1]

Using MST [10] in order to build the spanning tree will not improve the theoretical worst-case ratio, but may have more practical advantages. By setting the weight function according to specified preferences, MST allows for the construction of a spanning tree with predefined attributes for a mobile ad hoc network. Below we illustrate how to utilize such a paradigm. A topic for further research is to explore whether the advantages of both the MCDS and the MST paradigms can be combined.

Power Control. Power usage in wireless ad hoc networks is a cause of some concern and has motivated research into power-aware and power-efficient wireless protocols. Algorithms designed to conserve power currently address data-link, network and transport layer protocols. Wireless transmission reception, retransmission and beaconing operations all consume power, making advanced power conservation techniques an essential factor in the design of wireless protocols.

Consider using Maximum Spanning Tree (MST) to construct the spanning tree during the first phase of our algorithm, with the weight of an edge reflecting the strongest of the battery power of the two nodes that are incident to the edge. Using MST will enhance the stability of the clustering by reducing the amount of power-consuming activities on nodes that have a low battery life. The latter holds since the edges belonging to the subtrees of the resulting partition will each contain at least one node with a long battery life.

5. RELATED WORK

Contemporary research done in the area of clustering in ad hoc networks has concentrated on two main topologies. In the first topology, the more common cluster-based scheme for ad hoc networks, a set of nodes, referred to as cluster-heads, forms a dominating set of the network. This set of nodes is also an independent set of the network. The role of cluster-heads is to control channel access, perform power measurements, maintain time division frame synchronization and guarantee bandwidth for real time traffic, as illustrated in [13]. In this type of centralized hierarchical architecture, cluster-heads play an important role by coordinating their one-hop neighbors. However, unlike the base station in a conventional cellular system, cluster-heads have no special hardware and may change dynamically. Thus, a radio station that acts both as a local radio node and as a cluster-head can easily become the system bottleneck. A generalized load-balancing solution which resolves the asymmetrical loads induced on cluster-heads is given in [3]. This is achieved by a circular queue that distributes the responsibility of acting as a cluster-head evenly among all of the cluster nodes. A number of algorithms have been proposed for dealing with cluster formation in ad hoc networks. Most of the algorithms that have been proposed for this topology are based on two fundamental algorithms, the lowest-ID [5] and the highest degree [16]. A generalization of this topology, where each node is at most k hops from a cluster-head, was proposed in [4]. A different approach [6] is to elect cluster-heads from a Weakly Connected Dominating Set (WCDS) of the input network, which is modeled as a simple undirected graph. The authors in [6] presented a distributed approximation for the WCDS problem with a worst case ratio of $O(lan\Delta)$, where Δ is the maximum degree of the input graph.

The second topology that has been proposed for clustering in ad hoc networks is the one that is dealt with in this paper - the k-clustering. The k-clustering approach maintains an infrastructure where the network is decomposed into clusters of nodes. This approach is more symmetric than the previous one since there are no specific nodes that are designated cluster-heads. This sort of clustering infrastructure serves to create a dynamic backbone of a network for the purpose of minimizing the amount of data to be exchanged in order to maintain routing and control information in a mobile environment. Although k-clustering was suggested and defined, until now only the case of 1-clusters (i.e., cliques) have been explored in the context of ad hoc networks. In [12], Krishna et al. presented algorithms for forming clique partitioning, as well as algorithms for maintaining the cliques in the face of various network occurrences. A slightly similar approach, the (a, t)-cluster proposed in [15], is a framework for dynamically organizing mobile nodes in wireless ad hoc networks into clusters for which the probability of path availability can be bounded. Combining our k-clustering algorithm for cluster formation with the (a, t)-cluster for cluster maintenance will provide a full paradigm for clustering in ad hoc networks.

The problem of k-clustering in general undirected graphs is known to be NP-Complete [8]. No previous research has been done on the subject of approximating k-clustering on simple undirected graphs and unit disk graphs. Contemporary (mostly theoretical) research relating to graph clustering is restricted to special classes of graphs. For example, in [9], the k-clustering problem is investigated on trees and a linear time algorithm solving the k-clustering problem is presented. In addition, in [1], Abbas and Stewart give a linear-time algorithm for solving the k-clustering problem on restricted special classes of graphs, such as interval graphs and bipartite permutation graphs. Furthermore, they demonstrate that the k-clustering problem remains NP-Complete when restricted to bipartite graphs for any fixed k, such that $k > 1$. In [8], Deogun et al. introduce a polynomial-time approximation algorithm of constant worst case ratio that computes a k-clustering for graphs having a domination diametric path. Our work differs from all of the above in providing an approximation for general unit disk graphs, and in providing a distributed algorithm realizing it.

Another problem that is closely related to k-clustering is geometric clustering. In geometric clustering, the objective

[1] Since the objective according to the definition of our weight function is to maximize the spanning tree weight, we will assume our MST calculates a Maximum Spanning Tree, instead of the conventional minimum.

is to minimize the size of the clusters while the order of the partition is given. The problem of geometric k-clustering was shown to be NP-Complete by Gonzales in [11] and a sequential approximation algorithm with a worst case ratio of 2 was given. Although the k-clustering and the geometric k-clustering problems are similar, we cannot use an approximate solution to one in order to solve the other. In addition, geometric k-clustering makes use of geometric coordinates, which means that deployment on mobile devices necessitates the use of Global Positioning Systems (GPS).

6. CONCLUSION

K-clustering is a framework in which the wireless network is divided into non-overlapping clusters and where the size of such clusters is at most k. The purpose of this network decomposition is to support a hybrid approach to routing that maintains only partial information of network topology. This type of routing scheme was proposed in order to overcome the mobility of ad hoc networks by adjusting cluster size according to network stability, and it is capable of supporting routing in large ad hoc networks with high mobility rates.

This paper investigates k-clustering formation, a fundamental requirement of this type of routing scheme, and proposes a distributed asynchronous algorithm for the problem that approximates the optimal solution with a worst case ratio of $O(k)$. The algorithm has polynomial time and message complexity. Our approach offers a generic two-stage algorithm for k-clustering that may be used to set various cluster properties, and which causes the resulting clusters to have a virtual backbone.

7. REFERENCES

[1] N. Abbas and L. Stewart. Clustering bipartite and chordal graphs: Complexity, sequential and parallel algorithms. *Discrete Applied Mathematics*, pages 1–23, 1999.

[2] K. Alzoubi, P.-J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS)*, January 2002.

[3] A. Amis and R. Prakash. Load-balancing clusters in wireless ad hoc networks. In *Proceedings of ASSET 2000*, March 2000.

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

[5] D. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communication*, pages 1694–1701, November 1981.

[6] Y. P. Chen and A. L. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *Proceedings of the third ACM international symposium on Mobile ad hoc networking and computing*, pages 165–172. ACM Press, 2002.

[7] B. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

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

[9] A. Farley, S. Hedetniemi, and A. Proskurowski. Partitioning trees: matching, domination, and maximum diameter. *International Journal Computing Information Science*, 10(1):55–61, February 1981.

[10] R. Gallager, P. Humblet, and P. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions and Decision Systems*, 5(1):66–77, January 1983.

[11] T. Gonzalez. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science, NorthHolland*, 38:293–306, 1985.

[12] P. Krishna, N. Vaidya, Chatterjee, and Pardhan. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, pages 49–64, 1997.

[13] C.-H. Lin and M. Gerla. A distributed control scheme in multi-hop packet radio networks for voice/data traffic support. In *Proceeding of IEEE GLOBECOM*, pages 1238–1242, 1995.

[14] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.

[15] A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in ad hoc networks. *IEEE Journal on Selected Areas in communications*, 17(8):1466–1487, August 1999.

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

APPENDIX

A. MCDS ALGORITHM PSEUDO-CODE

In this section, we provide for completeness a pseudo-code description of the Minimum Connected Dominating Set (MCDS) algorithm, which is employed in the first stage of the k-clustering partitioning algorithm.

---

[1] Arbitrarily pick a node $v \in V$.
[2] Construct the BFS tree T of G rooted at v.
[3] Let ℓ be the height of T.
[4] Let Si denote the nodes at height i in T
[5] Set $IS_0 = \{v\}; NS_0 = \emptyset$.
[6] for $i = 1$ to ℓ do begin
[7] $DS_i = \{v | v \in S_i$ and
[8] v is dominated by some node in $IS_{i-1}\}$.
[9] Pick a MIS IS_i in $G[S_i \setminus DS_i]$.
[10] $NS_i = \{u | u$ is the parent (in $T)$
[11] of some $v \in IS_i\}$
[12] (Note that $NS_i \subseteq S_{i-1}$.)
[13] end
[14] output $(\bigcup_{i=0}^{\ell} IS_i) \bigcup (\bigcup_{i=0}^{\ell} NS_i)$ as the CDS.

Figure 2: Pseudo code of sequential minimum connected dominating set.