The (*k*,*l*) Coredian tree for Ad Hoc Networks

Amit Dvir and Michael Segal
Department of Communication Systems Engineering
Ben Gurion University of the Negev
Israel
azdvir@cse.bgu.ac.il
segal@cse.bgu.ac.il

Abstract

In this paper, we present a new efficient strategy for constructing a wireless tree network containing n nodes of diameter Δ while satisfying the QoS requirements such as bandwidth and delay. Given a tree network T, a coredian path is a path in T that minimizes the centdian function, a k-coredian tree is a subtree of T with k leaves that minimizes the centdian function, and a (k,l)-coredian tree is a subtree of T with k leaves and diameter l at most that minimizes the centdian function. The (k,l)-coredian tree can serve as a backbone for a network, where the internal nodes belong to the backbone and the leaves serve as the heads of the clusters covering the rest of the network. We show that a coredian path can be constructed at $O(\Delta)$ time with $O(n)$ messages and a k-coredian tree can be constructed at $O(k\Delta)$ time with $O(kn)$ messages. We provide an $O(n^2)$ time construction algorithm for the (k,l)-coredian tree that requires $O(n^2)$ messages. We also give upper and lower bounds for a number of nodes covered by the k cluster heads in random geometric graph using critical transmission range of connected network. Finally, simulation is presented for various values of n and k.[1]

Keywords: Ad hoc networks, Sensor networks, Backbone, Centdian

I. INTRODUCTION

A wireless ad hoc network is a network architecture containing a number of nodes distributed across an area using wireless communication links to deliver information between nodes. The network topology changes rapidly because nodes' motion, frequent failures, frequent recoveries,

[1] This research has been partially supported by INTEL and REMON consortium

power limitations and additional problems related to the propagation channels (e.g. obstructions, noise) [19, 25, 27, 36]. This network provides to users with the ability of spontaneously forming a dynamic communication system and allows them to access services of multi-hop communications. However, in order to offer QoS for ad hoc network nodes the wireless networks have to satisfy user requests with minimum service *delay* and *bandwidth*.

A wireless ad hoc sensor network contains a number of sensor nodes limited in power and memory, distributed across an area using the wireless communication links to deliver information between nodes. Unlike the simple ad hoc network, the topology of the sensor network barely changes. Currently, the analysis of wireless ad hoc sensor network gains a lot of attention [17, 18, 20, 21], because this kind of network can be used in a variety of application areas such as health, military and emergence. One of the generic types of applications for these networks is monitoring where all sensors produce relevant information by sensing the area and transmitting it to a central node called sink node by broadcasting request.

Hierarchical structures have been used to provide scalable solutions in many large networking systems [14–16, 37]. These networks are composed of large numbers of low-power nodes that collect some information, which can be sent to the center of the network. Bejerano [16] studied the problem of connecting static wireless networks by a wired backbone ensuring the QoS requirements are fulfilled. The infrastructure provided by [16] is important for low cost and fast deployed access networks and for providing access to sensor networks. In these networks some nodes are selected as the gateways (cluster heads) to access a wired/wireless backbone where each of the cluster heads serves a cluster of nearby wireless users (cluster nodes). In such networks, the aggregation of nodes into clusters controlled by a cluster head provides a convenient framework for the development of important features such as code separation (among clusters), channel access, routing, and bandwidth allocation [40, 42].

In sensor networks, scalability is one of the important issues since they expected to operate with up to millions of nodes. This has implications, particularly with energy, which ideally should not be wasted on sending data to base stations that are potentially far away. Energy waste can be prevented by separating the sensor networks into clusters and nominating nodes that carry out aggregation and forward the data to the sink [41].

In this paper we present a new concept of constructing a wireless subtree network while satisfying the QoS requirements such as bandwidth and delay. The proposed infrastructure can be used to collect information from sensor networks or to provide services to satisfy QoS constrains in mobile ad hoc or sensor networks. The motivation comes from gathering data application by request where the sink node forms a broadcast tree and asks from the sensing nodes to transmit data from the sensing areas for a certain period of time aiming to minimize the bandwidth and delay criterions. Our concept uses a subtree network as the backbone, and partitions the rest of the nodes into clusters in which each leaf of the backbone serves as the head of the cluster. The head of the cluster plays a role of the gateway to access the wireless backbone for all its cluster nodes. By selecting a given number of clusters with a limited number of nodes in each cluster and by bounding the diameter of the backbone, we provide an efficient backbone construction that leads to a balance between the convergecast and delay constraints.

Let $G(V, E)$ be the wireless network where V is the set of wireless nodes ($|V| = n$) and E is the set of undirected edges representing wireless connection between the nodes. Let $T(V, E')$ be the tree network where $E' \subseteq E$ and let U be the set of leaves in the tree network T, $U \subset V$. For any two nodes $u, v \in V$, $d(u, v)$, is the minimal weighted distance path between u and v i.e. the sum of the weighted edges in the tree network between u and v. Let $dist(v)$ be the maximum weighted distance from node v to other nodes in the network, and $sum(v)$ be the sum of weighted distances in the network from all nodes to node v. If P is a path in T, then $d(P)$ denotes the

total weight of weighted edges on P. The total weighted distance from all the nodes to path P is $sum(P) = \sum_{u \in V} d(u, P)$, where $d(u, P) = \min_{v \in P} d(u, v)$ and $dist(P)$ is the weighted distance from the farthest leaf in T to the path P. The *center* of a tree network T is a node $c \in T$, in which the maximal weighted distance from c to any other node in T is minimized, i.e. $dist(c) = \min_{v \in T} dist(v)$. The *median* of a tree network T is a node $m \in T$, in which the sum of weighted distances from other nodes in T to node m is minimized, i.e. $sum(m) = \min_{v \in T} sum(v)$.

The *transport* of a tree T rooted at v is defined as the total weighted distance of packet transmissions required to deliver packets from all nodes to the core node v by a convergecast process on the tree. The maximum delay of the tree T rooted at v is the maximum weighted distance to be traversed by any packet in traveling from core node v to other nodes. The corresponding solution concepts for convergecast and delay constraints have been considered in the literature as median and center [1–3]. By choosing the core to be the median node we minimize the convergecast of the network, but we overlook the nodes at the network peripheral. By choosing the core to be the center node we minimize the delay of the network. However, locating the core at the center may cause a large increase in the total weighted distance from all the nodes to the core. The compromise of using only center or median as a core, lead to a search for the concept called *centdian*, where the centdian function presents a kind of trade-off between the center and the median functions [4, 5, 7]. The centdian function $D_\lambda(v)$ for node v in the network is defined by the following expression:

$$D_\lambda(v) = \lambda \cdot sum(v) + (1 - \lambda) \cdot dist(v),\ 0 \leq \lambda \leq 1 \tag{1}$$

The centdian concept is well known in the facility location field [4, 5, 13, 26, 33–35, 38, 39]. Averbakh and Berman [13] considered the problem for finding an optimal location for a path on a tree network, using combinations of minisum and minimax criteria. They [13] minimized both criteria separately but did not use the above centdian function. Becker et al. [26] considered the same problem as [13] with a path length of at most l on a tree network. Becker et al. [38] dealt with two

related problems. The first was to find a path P on a tree network that minimizes $sum(P)$, when the length of P and the weighted distance from the farthest leaf in T to the path P are bounded by a fixed constant. The second issue was to find a path P on a tree that minimizes $dist(P)$, when the length of P and the total weighted distance from all the nodes to path P are bounded by a fixed constant. Dvir and Segal [7] were first to deal with the centdian function as expressed by Eq. (1) in the context of ad hoc networks.

The center path of tree T is a path P which minimizes $dist(P)$, while a core path X of T is a path which minimizes $sum(X) = \sum_{v \in V} d(v, X)$. Jennings [6] presented distributed algorithms for finding center path and core path in asynchronous networks in $O(\triangle)$ time with $O(n)$ messages, where $\triangle$ is the diameter of the network. Additional results on center/core paths can be found in [6, 8–11, 22–24].

The (k, l)-core tree of T is a subtree $T' \subseteq T$ that minimizes the sum of the weighted distances from the nodes of T to the subtree T', with precisely k leaves, and a diameter of at most l. The problem of constructing a (k, l)-core tree is a constrained version of the k-core tree problem [8, 11] with an unbounded diameter ($l = \infty$). Peng et al. [8] were first to present an $O(n \log n)$ and $O(kn)$ time algorithms that solve the k-core tree problem. Later, Shioura and Uno [9] improved the results for the k-core tree problem to $O(n)$ time. Becker et al. [11] dealt with the (k, l)-core tree problem and presented an $O(n^2 \log n)$ time algorithm for arbitrary edge lengths and an $O(n^2)$ time algorithm for equal edge lengths. Recently, Wang et al. [10] presented two algorithms for constructing a (k, l)-core tree. The first algorithm has $O(n^2)$ time complexity for the case in which each edge has an arbitrary length. The second algorithm has $O(lkn)$ time complexity for the case in which the lengths of all edges are 1. All the above mentioned algorithms are centralized and therefore, are not applicable for use in networking environment.

The k-center tree is a subtree $T' \subseteq T$ minimizing the weighted distance from the farthest

leaf to the subtree T', with precisely k leaves. The (k,l)-center tree of T is subtree $T' \subseteq T$ that minimizes $dist(T')$, with precisely k leaves and a diameter of at most l. Efficient $O(\triangle)$ time solution for construction of k-center tree has been shown by Wang [12].

The rest of the paper is organized as follows: In Section II we show how to construct a coredian path. Afterwards in Section III, we prove that a coredian path is contained in (k,l)-coredian tree and present a construction of (k,l)-coredian tree. At Section IV we provide an upper and lower bounds for a number of nodes covered by k clusters. Finally, we show simulation results and conclude with future directions for research.

II. FINDING A COREDIAN PATH

In this section we will show how to construct a coredian path in $O(\triangle)$ time with $O(n)$ messages in asynchronous distributed fashion. First we give a number of definitions and afterwards we show some useful properties that allow us to build the coredian path efficiently.

Let $P_{v,u}$ be a path in the tree network between node v and node u. Path $P_{v,u}$ is the coredian path for given $\lambda, 0 \leq \lambda \leq 1$ if its centdian function achieves a minimum over all other paths in the tree network, where the centdian function of a path $P_{v,u}$ is defined as $D_\lambda(P_{v,u}) = \lambda \cdot sum(P_{v,u}) + (1-\lambda) \cdot dist(P_{v,u})$, $0 \leq \lambda \leq 1$. Let node v be a leaf, r an arbitrary node and c the center node in the tree network. Let $P_{r,v}$ be a path in the tree network such that $P_{r,v} =\prec u_1, u_2, \ldots u_t \succ$, where $u_1 = r$ and $u_t = v$. Define $saved_1(P_{r,v}) = sum(v) - sum(P_{r,v})$ and $saved_2(P_{r,v}) = dist(v) - dist(P_{r,v})$. Notice that $saved_1$ and $saved_2$ are positive or equal to zero values. Wang [12] showed that $saved_1(P_{r,v}) = \sum_{1 \leq i \leq t-1} d(u_i, u_{i+1}) \cdot size(u_{i+1})$ where $size(u_{i+1})$ is the number of vertices contained in the subtree rooted at u_{i+1} with T rooted at r. Define $saved_\lambda(P_{r,v}) = D_\lambda(v) - D_\lambda(P_{r,v})$, where $D_\lambda(v)$ is the centdian function of leaf v and $D_\lambda(P_{r,v})$ is the centdian function of the path $P_{r,v}$, where r is an arbitrary node in T.

Lemma II.1: Two endpoints of coredian path are leaves of a tree network T.

Proof: By definition, the coredian path P attempts to minimize the centdian function. Jennings [6] proved that the endpoints of the core path are leaves whereas the endpoints of the center path may not necessarily be leaves. From the centdian function we can observe that the $dist(P)$ value can remain the same for subpath $P' \subseteq P$ during the process of growing P' to $P' = P$, but the $sum(P')$ has to decrease. Thus, the minimum of the centdian function will be achieved when the path touches two leaves from both ends. ■

Lemma II.2: In a tree network T, where c is the center of T and v is any leaf of T, $saved_2(P_{c,v}) = d(P_{c,v})$.

Proof: The diameter of any tree network is the longest path in the tree network, where the center node lies on the diameter. Let $P_{c,x}$ and $P_{c,y}$ be the paths that composing the diameter path $P_{x,y}$ in our tree network. Assume that $d(P_{c,x}) \geq d(P_{c,y})$. If $v \in P_{c,x}$, where v is the end node of the path ($v = x$, as shown in Figure 1(a)) then the path that defines $dist(v)$ value contains the center node c. Otherwise, we can increase our diameter. Therefore $dist(v) = d(P_{v,c}) + d(P_{c,y})$, $dist(P_{c,v}) = d(P_{c,y})$ and $saved_2(P_{c,v}) = d(P_{v,c}) + d(P_{c,y}) - d(P_{c,y}) = d(P_{v,c}) = d(P_{c,v})$. If $v \notin P_{c,x}$ as shown in Figure 1(b), then the path that defines the $dist(v)$ value has to be the path from v containing the center node c, which then combines to $P_{c,x}$. Therefore $dist(v) = d(P_{c,v}) + d(P_{c,x})$, $dist(P_{c,v}) = d(P_{c,x})$ and $saved_2(P_{c,v}) = d(P_{c,v}) + d(P_{c,x}) - d(P_{c,x}) = d(P_{c,v})$ (this case is equivalent to the case where $v \in P_{c,y}$). ■

The following lemma shows that $saved_\lambda$ value is a balanced combination of $saved_1$ and $saved_2$ values. Thus, it shows a relationship between convergecast and delay constraints.

Lemma II.3: Let r be an arbitrary node and v a leaf in T. Then $saved_\lambda(P_{r,v}) = D_\lambda(v) - D_\lambda(P_{r,v}) = \lambda saved_2(P_{r,v}) + (1-\lambda) saved_1(P_{r,v})$.

Proof: By definition $saved_\lambda(P_{r,v}) = D_\lambda(v) - D_\lambda(P_{r,v})$. Therefore $D_\lambda(v) - D_\lambda(P_{r,v}) =$

$[\lambda dist(v) + (1-\lambda)sum(v)] - [\lambda dist(P_{r,v}) + (1-\lambda)sum(P_{r,v})] = \lambda(dist(v) - dist(P_{r,v})) + (1-\lambda)(sum(v) - sum(P_{r,v})]) = \lambda saved_2(P_{r,v}) + (1-\lambda)saved_1(P_{r,v})$. ■

Lemma II.4: Let $P_{c,l}$ and P_{c,l_1} be two rooted paths in T_c from the center c of the tree network T to leaves l and l_1, respectively, when $D_\lambda(P_{c,l}) \leq D_\lambda(P_{c,l_1})$. Let $P_{c,i}$ be a path in which $P_{c,i} \cap P_{c,l} \neq \emptyset$, $P_{c,i} \cap P_{c,l_1} \neq \emptyset$. Then $D_\lambda(P_{i,l}) \leq D_\lambda(P_{i,l_1})$.

Proof: Let $P_{c,x}$ and $P_{c,y}$ be the paths that composing the diameter path $P_{x,y}$ in our tree network T, assuming that $d(P_{c,x}) \geq d(P_{c,y})$. From the definition of the centdian function we have

$$D_\lambda(P_{c,l}) = \lambda \cdot sum(P_{c,l}) + (1-\lambda) \cdot dist(P_{c,l})$$

$$D_\lambda(P_{c,l_1}) = \lambda \cdot sum(P_{c,l_1}) + (1-\lambda) \cdot dist(P_{c,l_1}).$$

Figure 2 shows an example of two paths $P_{c,l}$ and P_{c,l_1} that are rooted at the center of the tree network and have a common path $P_{c,i}$. Since $c \in P_{x,y}$,

$$|dist(P_{c,l}) - dist(P_{c,l_1})| \leq d(P_{c,x}) - d(P_{c,y}) \tag{2}$$

and from $D_\lambda(P_{c,l}) \leq D_\lambda(P_{c,l_1})$ it follows that

$$|sum(P_{c,l}) - sum(P_{c,l_1})| \leq d(P_{c,x}) - d(P_{c,y}) \tag{3}$$

Removing the common path $P_{c,i}$ from both paths $P_{c,l}$ and P_{c,l_1} gives

$$|dist(P_{i,l}) - dist(P_{i,l_1})| \leq d(P_{c,x}) - d(P_{c,y}) \tag{4}$$

However, $sum(P_{i,l}) = sum(P_{c,l}) + X$, $sum(P_{i,l_1}) = sum(P_{c,l_1}) + X$ and therefore, according to Eq. (2),

$$|sum(P_{i,l}) - sum(P_{i,l_1})| \leq d(P_{c,x}) - d(P_{c,y}) + 2X \tag{5}$$

Thus, according to Eq. (4) and Eq. (5), $D_\lambda(P_{i,l}) \leq D_\lambda(P_{i,l_1})$. ■

Lemma II.5: Let v be a leaf of the tree network T_c and c be the center node of the tree network. If $saved_\lambda(P_{c,v}) \geq saved_\lambda(P_{c,u}), \forall u \in U$, then the leaf v is one of the end points of the coredian path.

Proof: Define rooted coredian path as a path starting from the root node z to one of the leaves minimizing the centdian function over all the paths from the root node to the leaves. Let us define the coredian path of the tree network $CP = P_{l_1,l_2}$, and $RCP = P_{c,l}$ as the rooted coredian path of T_c, rooted at c. Let us assume that both paths do not intersect as shown in Figure 3(a). Let i be the closest node in RCP to CP and let j represent the closest node in CP to RCP. Let us define path $TP = P_{l,l_2} = P_{l,i} \cup P_{i,j} \cup P_{j,l_2}$. First, we show that RCP and CP have to intersect. Notice that $D_\lambda(CP) \leq D_\lambda(RCP)$, $D_\lambda(RCP) \leq D_\lambda(P_{c,l_1})$, $P_{c,i} \subseteq RCP$ and that $P_{c,i} \subseteq P_{c,l_1}$. According to Lemma II.4, $D_\lambda(P_{i,l}) \leq D_\lambda(P_{i,l_1})$, and following the definition of the coredian path $D_\lambda(P_{i,l_1}) < D_\lambda(P_{j,l_1})$, thus $D_\lambda(P_{i,l}) \leq D_\lambda(P_{i,l_1}) < D_\lambda(P_{j,l_1})$. We get that $D_\lambda(P_{i,l}) < D_\lambda(P_{j,l_1})$ or in other words $D_\lambda(P_{j,l}) < D_\lambda(P_{j,l_1})$. By adding to both paths $P_{j,l}$ and P_{j,l_1} the common path P_{j,l_2} the inequality is still satisfied and becomes $D_\lambda(P_{l,l_2}) < D_\lambda(P_{l_1,l_2})$, i.e. $D_\lambda(TP) < D_\lambda(CP)$, which contradicts to the definition of CP. Therefore $CP \cap RCP \neq \emptyset$.

Next, we assume that $CP \cap RCP \neq \emptyset$ and $l \notin CP$ as shown in 3(b), and we want to show that node l has to be one of the end nodes of the coredian path. By definition, $D_\lambda(P_{c,l}) \leq D_\lambda(P_{c,l_1})$, and therefore by Lemma II.4 $D_\lambda(P_{i,l}) \leq D_\lambda(P_{i,l_1})$. Adding a common part P_{i,l_2} to $P_{i,l}$ and P_{i,l_1} gives $D_\lambda(P_{l,l_2}) \leq D_\lambda(P_{l_1,l_2})$. Thus, $D_\lambda(TP) \leq D_\lambda(CP)$, which is in contradiction to the definition of CP. Therefore, CP has to contain leaf l as one of its end points. ■

Lemma II.6: Let P_{l_1,l_2} and $P_{z,l}$ be a coredian paths in T, i.e. $D_\lambda(P_{l_1,l_2}) = D_\lambda(P_{z,l})$. Then $P_{l_1,l_2} \cap P_{z,l} \neq \emptyset$.

Proof: Let P_{l_1,l_2} and $P_{z,l}$ be a coredian paths in T that do not intersect as shown in Figure 3(a).

By definition $D_\lambda(P_{l_1,l_2}) \leq D_\lambda(P_{l_1,z})$, using Lemma II.4 and the common path $P_{l_1,j}$, it can be concluded that

$$D_\lambda(P_{j,l_2}) < D_\lambda(P_{i,z}) \tag{6}$$

By definition $D_\lambda(P_{l,z}) \leq D_\lambda(P_{l,l_2})$, using Lemma II.4 and the common path $P_{l,i}$, we obtain

$$D_\lambda(P_{i,z}) < D_\lambda(P_{j,l_2}) \tag{7}$$

Eq. (6) and (7) stand in contradiction of each other, and therefore $P_{l_1,l_2} \cap P_{z,l} \neq \emptyset$ ■

Now we will explain how to find the first end point of the coredian path following the results of Lemma II.5. First, we need to find the center node c serving as the root of the tree network, as shown in Figure 4(a). By starting from the leaves and propagating messages to the root we can calculate the maximum $saved_\lambda$ value among the leaves of the rooted tree network and find the end point of the coredian path. Each leaf $z \in U$ sends to his parent a $FindSave(size(z), saved_1(z), saved_2(z))$ message, where $size(z)$ is the number of nodes in the subtree rooted by z (in the case of a leaf, $size$ = 1), $saved_1(z) = d(z, p_z)$, where p_z is the parent of the node z in the rooted tree network and $saved_2(z) = d(z, p_z)$. Any intermediate node u, that obtains $FindSave$ messages from all its sons in the rooted tree network calculates the following values:

- Using $saved_1$ and $saved_2$ of the sons, node u calculates the $saved_\lambda$ values of its sons and marks the maximum value of $saved_\lambda$ son.
- $size(u) = 1 + \sum size_i$, where $size_i$ is the number of nodes in the subtree rooted at the son i of u.
- $saved_1(u) = size * d(u, p_u) + saved_1$, where $saved_1$ is the value of the marked son that gives the maximum value of $saved_\lambda$.
- $saved_2(u) = d(u, p_u) + saved_2$, where $saved_2$ is the value of the marked son that gives the maximum value of $saved_\lambda$ and $p(u)$ is the parent node.

After calculating all these values, intermediate node u sends to its parent in the rooted tree network a $FindSave(size(u), saved_1(u), saved_2(u))$ message. When the root node c receives $FindSave$ message from all its sons, it marks the son yielding the maximum $saved_\lambda$ and propagates it through the marked path all the way to leaf l. Leaf l marks itself as the end point of the coredian path, as shown in Figure 4(b), and begins also to serve as the root of the tree network (c ceases to be the root).

After finding the first end point of the coredian path we compute its second end point. Starting the same process of messaging, as explained above with our newly rooted tree network, we can find the leaf which is the second end point of the coredian path, as shown in Figure 4(c). Thus, we conclude with

Lemma II.7: Given a tree network and a center node serving as the root, a coredian path can be found in $O(\triangle)$ time with $O(n)$ messages.

III. CONSTRUCTING THE (*k*, *l*) COREDIAN TREE

We present a new structure that can serve as a backbone of ad hoc networks. We call this structure a (k, l)-coredian tree of T, which is a subtree $T'(V', E')$ of T that minimizes the centdian function, having precisely k leaves and diameter of at most l.

A subtree $T'' \subseteq T$ is called l-maximal tree if the diameter of T'' is $\delta \leq l$ and any larger subtree containing T'' has a diameter larger than l. Let S be the set of all l-maximal subtrees of T. Clearly, there is a subtree in S that contains a (k, l)-coredian tree of T. In order to construct S we proceed as follows (see [10] for additional explanations):

- Arbitrarily select a node $r \in V$ and orient T into T_r.
- Initially, set $M = \emptyset$. Then, proceed to iterate as follows:
  - Find a leaf q that is farthest from r.
  - Determine $U = \{v | v \in V(T_r), d(q, v) \leq l\}$, and put $\langle U \rangle$ into M.

– If the diameter of the current T_r is larger than l then remove q from T_r and continue the next iteration. Otherwise, stop the algorithm.

The (k, l)-coredian tree problem can be solved as follows:

1. Construct S.
2. Within each subtree $A \in S$ we find a coredian path from which we construct a k-coredian tree, using a direct derivation from the algorithm presented in Section II, where a k-coredian tree is a subtree of T with k leaves that minimizes the centdian function.
3. Compute a (k, l)-coredian tree of T from a set of the k-coredian subtrees obtained at the last step.

The authors in [10] proved that the upper bound of $|S|$ is n, where n is the number of nodes in the tree network. They also have shown that a set S can be constructed in $O(n^2)$ time. Using their iterative algorithm to construct S, we get l-maximal tree in each iteration. By constructing a k-coredian tree from the l-maximal tree and keeping the best (k, l)-coredian tree over the all obtained results gives us the optimal (k, l)-coredian tree.

The direct derivation includes $k - 2$ iterative operations, starting each iteration from the first end point of the coredian path adds a new path to the subtree, to obtain the k-coredian tree. Leaves propagate to the first end point messages as explained above with a small difference. The subtree nodes (in the first iteration the coredian path is the subtree) only propagate the messages but do not calculate and update the values. In the following lemmas we prove that the coredian path has to be contained in the k-coredian tree. Starting from the coredian path we construct the k-coredian tree in $O(k)$ time with $O(kn)$ messages.

Lemma III.1: Every k-coredian tree network intersects every coredian path, $k \geq 3$.

Proof: Let CP be the coredian path with leaves l_1 and l_2 and KCT a k-coredian tree network. Suppose CP and KCT do not intersect, therefore there is a path $P(i, j)$ between CP and KCT where i is the closest node in CP to KCT and node j is the closest node in KCP to CP, as

shown in Figure 5 (clearly $i \neq j$). Let $l_3 \in U$ be any leaf of KCT. By definition of CP, $D_\lambda(P_{l_1,l_2}) \leq D_\lambda(P_{l_1,l_3})$ and by deleting the common path $P_{l_1,i}$ and using Lemma II.4 we have $D_\lambda(P_{i,l_2}) \leq D_\lambda(P_{i,l_3})$. Adding path $P_{i,j}$ to path P_{i,l_2} decreases $D_\lambda(P_{i,l_2})$ and therefore $D_\lambda(P_{j,l_2}) < D_\lambda(P_{i,l_3})$. Replacing P_{i,l_3} by P_{j,l_2} in KCT results in a better k-coredian tree, contradicting the assumption that KCT is the k-coredian tree. Thus, k-coredian tree network has to intersect the coredian path. ■

Lemma III.2: Every k-coredian tree network contains a coredian path, $k \geq 3$.

Proof: Let CP be the coredian path with leaves l_1 and l_2 and KCT be a k-coredian tree network. Suppose CP is not contained in KCT. According to Lemma III.1, there are two cases:

- CP and KCT intersect at a single vertex u. For example $P_{u,l_3} \in KCT$ and $P_{u,l_2} \in CP$, where $saved_\lambda(P_{u,l_3}) < saved_\lambda(P_{u,l_2})$ and $l_2, l_3 \in U$. By definition of CP we can replace P_{u,l_3} with P_{u,l_2} to create a subtree with k leaves that has a smallest cost (in terms of D_λ). It follows that P_{u,l_2} has to be a part of KCT and in the same way we can show that P_{u,l_1} has to be a part of KCT.
- CP and KCT share a common segment $P_{u,v}$. In this case at least one of the nodes u, v is an intermediate node. We can use the above proof to show that replacing path P_{u,l_3} with P_{u,l_2} create a subtree with k leaves that has a smallest cost (in terms of D_λ), where $P_{u,l_3} \in KCT$.

■

Lemma III.3: Given a tree network, a (k,l)-coredian tree can be found in $O(n^2)$ time with $O(n^2)$ messages.

Proof: From Lemma III.2, we know that KCT contains CP. A KCT can be considered as an extension of a coredian path by addition of $k-2$ paths. Let assume that KCT is the optimal

k-coredian tree. In order to prove that our algorithm works, we need the following facts:

- If CP is a coredian path contained in KCT, using the iterative algorithm starting from CP we obtain KCT. At any iteration of the algorithm, we look for path P, where $D_\lambda(currentTree \cup P) \leq D_\lambda(currentTree \cup P')$, where $p = currentTree \cap P$ and P' is any path in the tree network that $p' = currentTree \cap P'$, $p, p' \in V$. This fact arises easily from the way in which the algorithm works. As a result, if KCT contains CP and the algorithm starts from CP, in the end we will generate a KCT.
- Every coredian path is contained in a k-coredian tree. Let's assume that $CP = P_{a,b}$ is the coredian path contained in KCT of T as shown in Figure 6. Assume that our algorithm starts from C' and obtains k-coredian tree $KCT' \subseteq T$. From Lemma II.6 we know that $C \cap C' \neq \emptyset$. Our goal is to show that the paths chosen by the algorithm to add to C' are also included in KCT. Let us consider the steps at our algorithm after choosing C'. By starting from C' we can add one of the following paths, $P_{a,f}, P_{i,t}, P_{e,h}, P_{g,e}, P_{b,e}$ as the path that minimizes $saved_\lambda$. If the best path that minimizes the centdian function of the subtree is $P_{g,e}$, then we have a contradiction with the fact that C is a coredian path since $D_\lambda(P_{a,g}) > D_\lambda(P_{a,b})$. In the same way we can also prove that adding $P_{e,h}$ to C' leads to a contradiction. Therefore, we have only three candidates paths $P_{a,f}, P_{i,t}, P_{b,e}$ to add. Notice that $P_{b,e}$ and $P_{a,f}$ belong to CP and we can add them to C' because they appear in KCT. Regarding $P_{i,t}$, its addition to C' means that it also appears as an addition to C (when constructing KCT). Therefore constructing the k-coredian tree from C' obtains KCT.

■

IV. THE COVERAGE OF (*k*, *l*) - TREE IN RANDOM GEOMETRIC GRAPH

In ad hoc networks, either wireless or sensor, our goal is to build a backbone that will propagate the messages between the nodes and partition them into clusters, where each cluster has a cluster head connected to the backbone network. In this section we show upper and lower bounds for the number of nodes in the cluster areas after constructing a (k, l) tree in a random geometric graph on the unit square, where k is the number of clusters and cluster heads, l is the diameter bound of the backbone network. When we utilize the (k, l) backbone tree network we deal with a backbone that connects k clusters, where the leaf of the backbone serves as the cluster head and the l parameter controls the propagating time and energy between the clusters. Clearly, the transmission range of the cluster head has a great impact on the number of node inside of the cluster.

It is well known that ad hoc sensor networks have a strong connection to a random geometric graph $G(n, r)$, which is obtained by placing random n points uniformly on the unit square and connecting two points if their Euclidean distance is at most r [28, 29]. Gupta and Kumar [30] conclude that, with high probability, the critical transmission range of nodes placed randomly on a disk of unit area to obtain a connectivity should be $r^2 = \frac{\log n}{n}$, where n is the number of nodes in the network. Suppose we have build a (k, l)-coredian tree from a random geometric graph, where n points are placed uniformly on the unit square of the size $A > \frac{r^2}{\mu}$, $r^2 = \frac{\log n}{n}$. We claim the following:

Lemma IV.1: If the transmission range of the cluster heads is w, where $w^2 \geq \frac{2r^2}{\mu} = \frac{2\log n}{n\mu}$, the upper bound for the number of nodes in k clusters is $\theta(knw^2 + kl)$ and the lower bound is $\theta(nw^2 + k + l)$.

Proof: Avin and Ercal [31] showed that in a random geometric graph with uniform node density across the unit square, a square bin of the size $A > \frac{r^2}{\mu}$, $\mu \geq 1$ in connected network has $\theta(nA) = \theta(n\frac{r^2}{\mu}) = \theta(\log n)$ nodes. Each bin has size of A whereas in our network each cluster head has a

transmission range of w. When $w^2 = \frac{2 \cdot r^2}{\mu}$, the transmission range of each cluster head can cover 4 bin squares and can reach at most 8 bin more squares, as shown in Figure 7(a). Therefore in each cluster we have $\theta(nw^2)$ nodes [31]. Each cluster can stand by itself, therefore we will get k clusters without overlapping $\theta(knw^2 + kl)$ (upper bound) or fully overlapping $\theta(nw^2 + k + l)$ (lower bound), which leaves only 4 bins covered, as shown in Figure 7(b). Therefore, the upper bound is $\theta(knw^2 + kl) = \theta(k\frac{\log^2 n}{n} + kl)$ and the lower bound is $\theta(nw^2 + k + l) = \theta(\frac{\log^2 n}{n} + k + l)$.

■

V. SIMULATION

This section describes in detail the medium-scale experiment. The objectives of the experiment were to test whether the algorithm actually works and to compare the results with the performance of other (k, l) trees. To test our algorithm we have simulate a number of (k, l) trees (core, center and coredian), where in each simulation we build a backbone, using our algorithm and other well-known algorithms for the center and core trees, and check the performance of each backbone.

A. Environment

We have used OMNET [32] environment with Pentium 4, 1G RAM, 1.8Ghz processor and Windows as OS. The following assumptions were made:

- For each node, transmission and reception ranges are different.
- All the nodes are equal in their functionalities and abilities.
- There is no dependence between the nodes.
- We simulate a network with $n = 30/50/100$ nodes with a variety of λ and l values.

We define the network behavior in a specific scenario based on predefined parameters:

- Number of nodes in the network.

- Number of cluster heads (k parameter).
- Maximum diameter of the subtree (l parameter).
- λ value.

B. Results

The main goal of our simulation is to examine the influence of the new (k, l)-coredian tree as the backbone network. From the obtained results we can learn that when $\lambda \geq 0.5$, the (k, l)-coredian tree is similar to the (k, l)-core tree. Figure 8 shows a number of values for the different simulation parameters for various values of k. The convergecast value (the total weighted distance of the nodes outside of the subtree to the subtree) is inspected for core, center and coredian subtree ($\lambda = 0.25$) and observed to be leveled for the core and the coredian as k grows up. Opposite, the weighted distance to the farthest leaf is the same for center and coredian subtrees. This can be explained by $\lambda = 0.25$ and Eq. (1), thus emphasizing the weight of the center function. It also can be seen from the simulation values ($n = 50$) that for $k = 2$ the coredian path and the center path have the same performance in terms of convergecast and farthest leaf and for $k = 4$ the coredian subtree and the core subtree get the same performance in terms of convergecast and farthest leaf. However, for $k = 6$ we learn that the (k, l)-coredian subtree performance is comparable with the (k, l)-core subtree performance but better than the (k, l)-center subtree performance.

Figures 9–11 show the influence of the network size on the constraints values of the network. Figure 9, in particular, shows the connection between the network size and the total weight of the subtrees. From the results we can conclude that as we increase the network size or the number of the clusters, the total weight of every (k, l) subtree also increases. We show at Figures 10 and 11 the convergecast (transport) and delay (farthest leaf) values of the (k, l)-subtrees. Figure 10 expresses that the (k, l)-coredian tree has obtained lower performance in terms of convergecast than the (k, l)-core tree but has better performance than the (k, l)-center tree for convergecast

criterion. Figure 11 shows that the (k, l)-coredian tree has obtained lower performance in terms of delay than the (k, l)-center tree but has better performance for delay criterion than the (k, l)-core tree. In summary, we can see that the (k, l)-coredian tree gets a balanced combination between the delay and convergecast (transport) parameters and give us the ability to chose the balance factor (either towards the center or the median functions) by setting an appropriate λ value.

VI. Conclusions and Future Work

We developed a new distributed algorithm for constructing a new (k, l)-coredian tree in ad hoc network, based on processing local information of the network. This new subtree can serve as a backbone for a network, where intermediate nodes serve as the backbone subtree and the leaves serve as the heads of the clusters covering the rest of the network. We also give an upper and lower bounds to a number of nodes covered by the k cluster heads in random geometric graph using critical transmission range of connected network. We test our new algorithm using the simulation and have shown that for various network size this algorithm can be used as the backbone tree. An interesting future research direction is to seek for a self-stabilizing solution to the (k, l)-coredian tree problem in ad hoc network, when it get partitioned or connected. Analysis of a model where one assumes some distribution for the velocities of the nodes also seems an attractive research direction. As for the future work, we consider extending our ideas to capture and combine node placement, network lifetime criteria, and data traffic management.

Acknowledgements. We would like to thank anonymous referees for their helpful remarks that improved the presentation of the paper.

References

[1] E. Korach, D. Rotem and N. Santoro, "Distributed Algorithms for Finding Centers and Medians in Networks", *ACM Trans. Program. Lang. Syst.*,Vol 6, (1984), pp. 380-401.

[2] J. H. Lin and J. S. Vitter, "Approximation algorithms for geometric median problems", *Information Processing Letters*, Vol. 44, (1992), pp. 245-249.

[3] M. H. Karaata, S. V. Pemmaraju, S. C. Bruell and S. Ghosh, "Self-stabilizing algorithms for finding centers and medians of trees", 13th Annual ACM Symposium on Principles of Distributed Computing, (1994), pp. 374-395.

[4] J. Halpern, "Finding minimal center-median convex combination (cent-dian) of a graph", *Management Science*, Vol. 16, (1978), pp. 534-544.

[5] J. Halpern, "The location of a centdian convex combination on an undirected tree", *Journal Regional Science*, Vol. 16, (1976), pp. 237-245.

[6] E. Jennings,"Distributed Algorithms for Finding Central Paths in Tree Networks", *Computer Journal*, Vol. 42, No. 7, (1999), pp. 609-612.

[7] A. Dvir and M. Segal, "Placing and Maintaining a Core Node in Wireless Ad Hoc Sensor Networks", *IFIP Networking*, (2007), pp. 500-513.

[8] S. Peng, A.B. Stephen and Y. Yesha, "Algorithms for a core and k-tree core of a tree", *J. Algorithms*, Vol. 15, (1993), pp. 143-159.

[9] A. Shioura and T. Uno, "A linear time algorithm for finding a k-tree core", *J. Algorithms*, Vol. 23, (1997), pp. 281-290.

[10] B. F. Wang, S. Peng, H. Y. Yu and S. C. Ku, "Efficient algorithms for a constrained k-tree core problem in a tree network", *Journal of Algorithms*, Vol. 59, No. 2, (2006), pp. 107-124.

[11] R. I. Becker, I. Lari, G. Storchi and A. Scozzari, "Efficient algorithms for finding the (k, l)-core of tree networks", *Networks*, Vol. 40, No. 4, (2002), pp. 208-251.

[12] B. F. Wang, "Finding a k-Tree Core and a k-Tree Center of a Tree Network in Parallel", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, (1998), pp. 186-191.

[13] I. Averbakh and O. Berman,"Algorithms for path medi-centers of a tree", *Computers and*

Operations Research, Vol. 26, No. 14, (1999), pp. 1395-1409.

[14] L. Kleinrock and K. Faroukh, "Hierarchical routing for large networks", *Computer Networks*, Vol. 1, (1997), pp. 155-174.

[15] Z. Xu, S. Dai and J. J. Garcia-Luna-Aceves, "Hierarchical routing using link vectors", *Infocom*, (1998), pp. 702-710

[16] Y. Bejerano, "Efficient Integration of Multi-hop Wireless and Wired Networks with QoS Constraints", *IEEE/ACM Trans. on Networking*, Vol. 12, No 6., (2004), pp. 1064-1078.

[17] S. Dulman, M. Rossi, P. Havinga and M. Zorzi, "On the hop count statistics for randomly deployed wireless sensor networks", *International Journal of Sensor Networks*, Vol. 1, (2006), pp. 89-102.

[18] J. Lansford and P. Bahl, "The Design And Implementation Of HomeRF: A Radio Frequency Wireless Networking Standard For The Connected Home" *Proceedings of the IEEE*, Vol. 88, (2000), pp. 1662-1676.

[19] Y. Ben-Shimol, A. Dvir and M. Segal, "SPLAST: A novel approach for multicasting in mobile ad hoc networks", *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 2, (2004), pp. 1011-1015.

[20] S. Meguerdichian, F. Koushanfar, G. Qu and M. Potkonjak, "Exposure In Wireless Ad-Hoc Sensor Networks", *Proc. ACM MobiCom*, (2001), pp. 139-150.

[21] H. Zhang and J. C. Hou, "Maximising α-lifetime for wireless sensor networks", *International Journal of Sensor Networks*, Vol. 1, (2006), pp. 64-71.

[22] C. A. Morgan and P. J. Slater, "A Linear Algorithm for a core of a tree", *Journal of Algorithms*, Vol. 1, (1980), pp. 247-258.

[23] E. Jennings, "Distributed Algorithm for Finding a Core of a Tree Network", *Lecture Notes In Computer Science*, (1995), Vol. 1012, pp. 385-390.

[24] S. Peng and W. T. Lo, "A Simple Optimal Parallel Algorithm for a Core of a Tree", *Journal of Parallel Distributed Computing*, Vol. 20, No. 3, (1994), pp. 388-392.

[25] *Mobile Ad Hoc Networks*. http://www.ietf.org/html.charters/manetcharter. html.

[26] R. I. Becker, Y. I. Chiang, I. Lari and A. Scozzari, "The Cent-dian Path Problem on Tree Networks", *Lecture Notes In Computer Science*, Vol 2223, (2001), pp. 743-755.

[27] E. Pagani and G. P. Rossi, "Reliable broadcast in mobile multi-hop packet networks", *IEEE international conference on Mobile computing and networking*, (1997), pp. 34-42.

[28] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors", *Communications of the ACM*, Vol. 43, No. 5, (2000), pp. 51-58.

[29] D. Estrin, R. Govindan, J. Heidemann and S. Kumar,"Next century challenges: Scalable coordination in sensor networks", *In Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking*, (1999), pp. 263-270.

[30] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity in wireless networks", *In Stochastic Analysis, Control, Optimization and Applications*, (1998), pp. 547-566.

[31] C. Avin and G. Ercal, "On the cover time and mixing time of random geometric graphs", *Theoretical Computer Science*, Vol. 380, (2007), pp. 2-22.

[32] OMNET, *http://www.omnetpp.org/*

[33] A. Tamir, "Fully polynomial approximation schemes for locating a tree-shaped facility: A generalization of the knapsack problem", *Discrete Applied Mathematics*, Vol. 87, (1998), pp. 229-243.

[34] A. Tamir, D. Perez-Brito and J. A. Moreno-Perez, "A polynomial algorithm for the p-centdian problem on a tree", *Networks*, Vol. 32, No. 4, (1998), pp. 255-262.

[35] A. Tamir, J. Puerto, D. Perez-Brito and A. M. Rodriguez-Chia, "The Pareto set for the doubly weighted center-median path problem on a tree", *The Institute for Operations Research and*

the Management Sciences ,(2005), pp. 20-30.

[36] L. Qin and T. Kunz, "Survey on Mobile Ad Hoc Network Routing Protocols and Cross-Layer Design", *Technical Report SCE-04-14*, (2004).

[37] J. Y. Yu and P. H. J. Chong, "A Survey of Clustering Schemes for Mobile Ad Hoc Networks", *IEEE Communications Surveys and Tutorials*, Vol. 7, (2005), pp. 32-48.

[38] R.I. Becker, I. Lari and A. Scozzari, "Algorithms for central-median paths with bounded length on trees", *Journal of Operational Research*, Vol. 179, (2007), pp. 1208-1220.

[39] E. J. Carrizosa, E. Conde, F. R. Fernandez and J. Puerto, "An axiomatic approach to the centdian criterion", *Location Science*, Vol. 2, (1994), pp. 165-171.

[40] C. Chiang, H. Wu, W. Liu, and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks", *Proc. IEEE Singapore Intl Conf. Networks*, (1997), pp. 197-211.

[41] E.J. Coyle and S. Bandyopadhyay, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks", *Proc. INFOCOM*, (2003), pp. 1-11.

[42] T. Sheltami and H.T. Mouftah, "Clusterhead Controlled Token for Virtual Base Station On-Demand in Manets" *Proc. Intl Conf. Distributed Computing Systems (ICDCS) Workshops*, (2003), pp. 716-721.

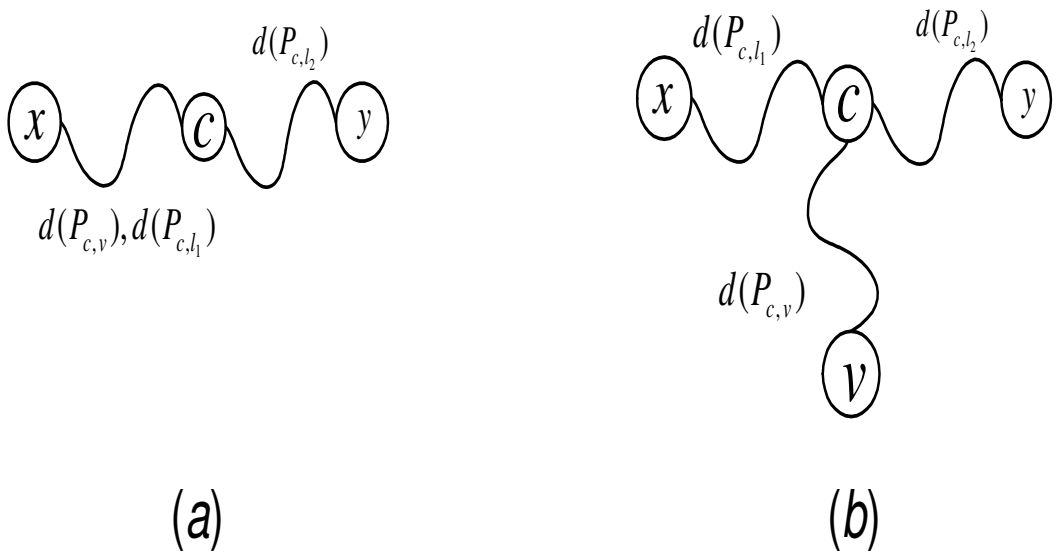


Fig. 1. Cases of Lemma II.2

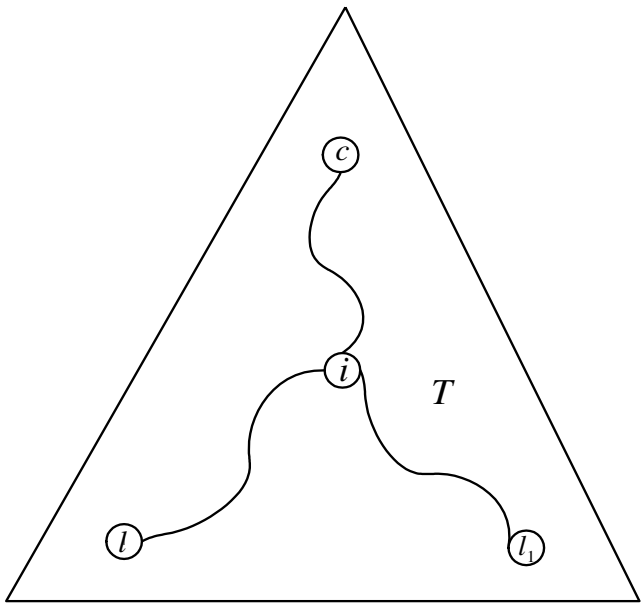


Fig. 2. Rooted paths from the center of the tree

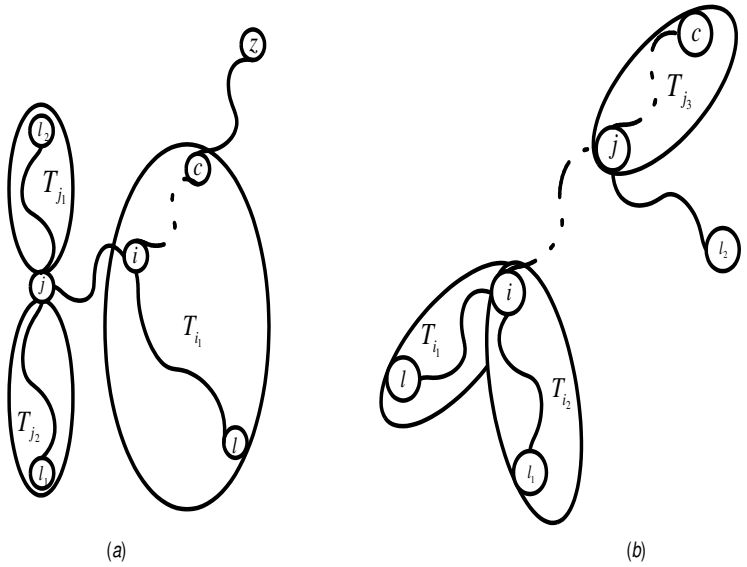
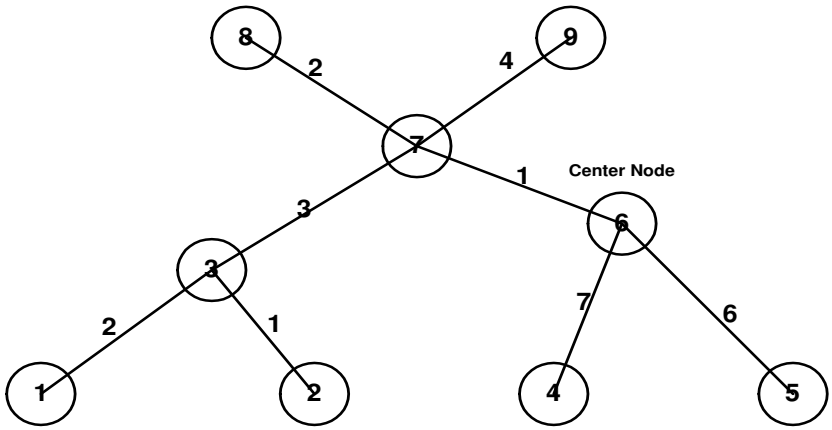
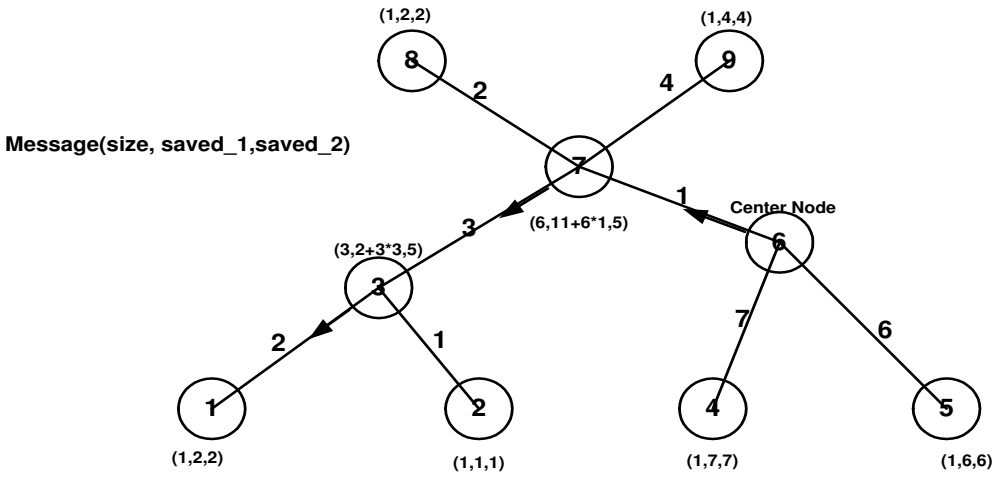


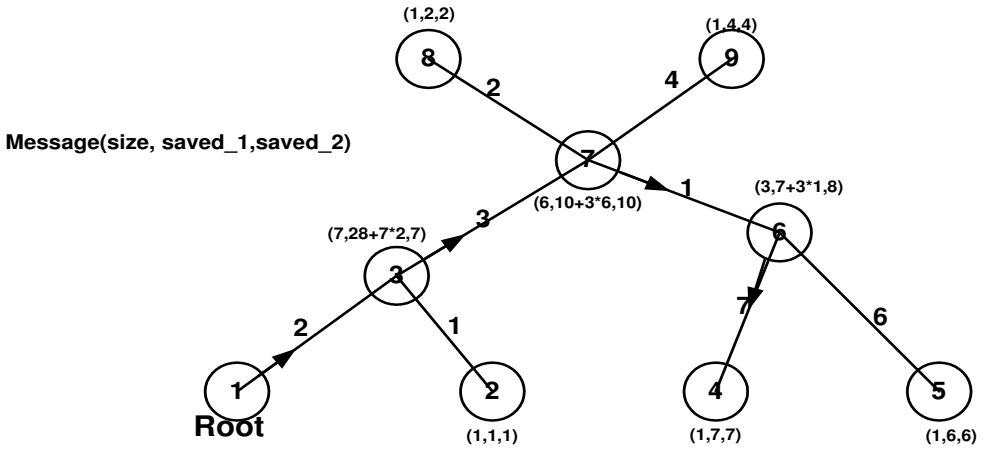
Fig. 3. The coredian and rooted coredian paths in T



(*a*) - The Network



(*b*) - The First End Point



(*c*) - Coredian Path

Fig. 4. Constructing the coredian path

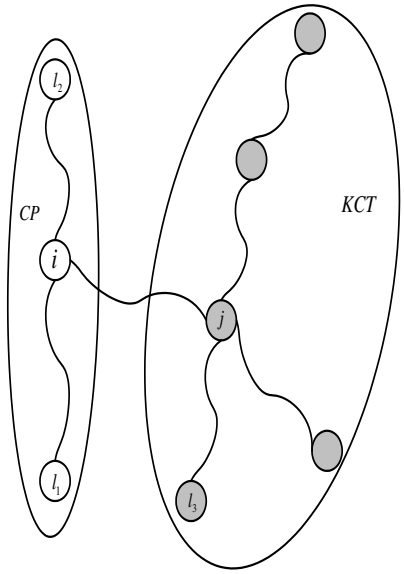


Fig. 5. Shaded nodes present KCT subtree, white nodes present CP path

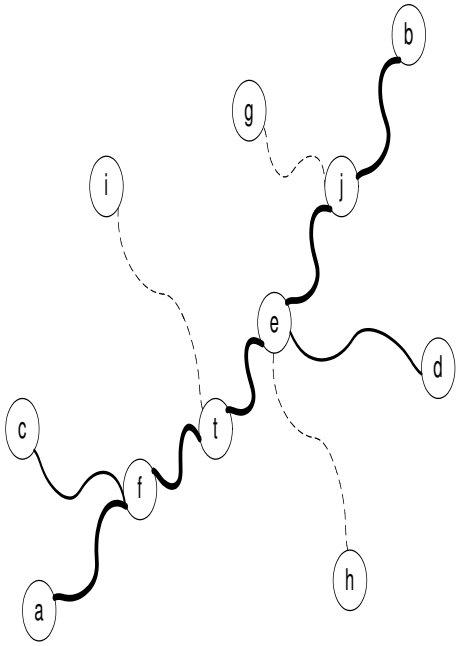


Fig. 6. Two coredian paths intersect

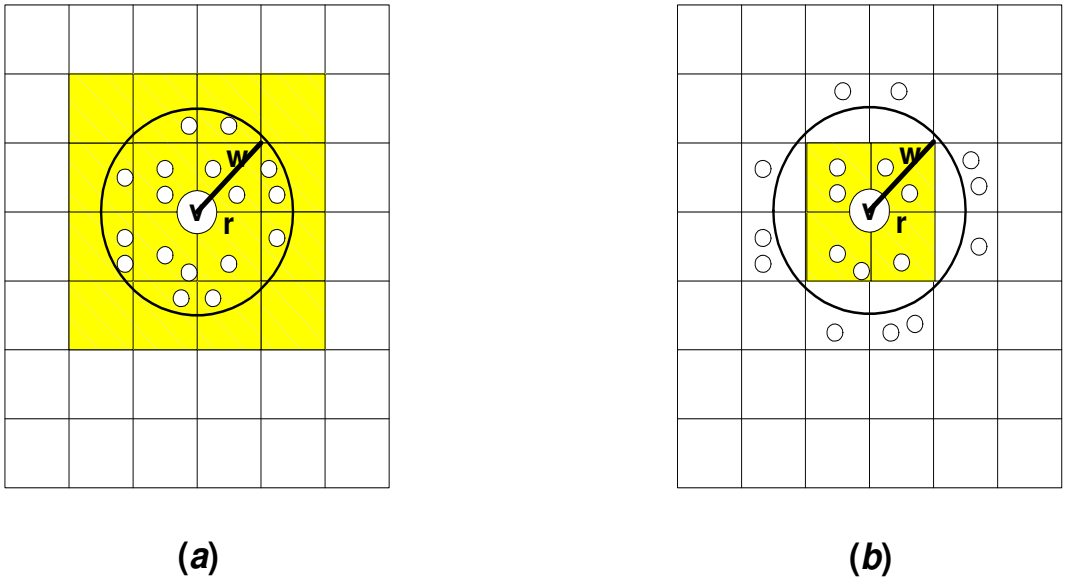


Fig. 7. Random geometric graph with uniform node density across the unit square, node v as cluster head

core tree

k	2	4	6	8
Total weight of the subtree	76	149	196	239
Convergecast	506	289	178	117
Farthest Leaf	40	25	23	17

center tree

k	2	4	6	8
Total weight of the subtree	81	150	198	239
Convergecast	512	304	201	117
Farthest Leaf	36	25	22	17

cordian tree, λ=0.25

k	2	4	6	8
Total weight of the subtree	81	149	197	239
Convergecast	512	289	186	117
Farthest Leaf	36	25	22	17

Fig. 8. Summary results of the simulation, n=50, λ=0.25

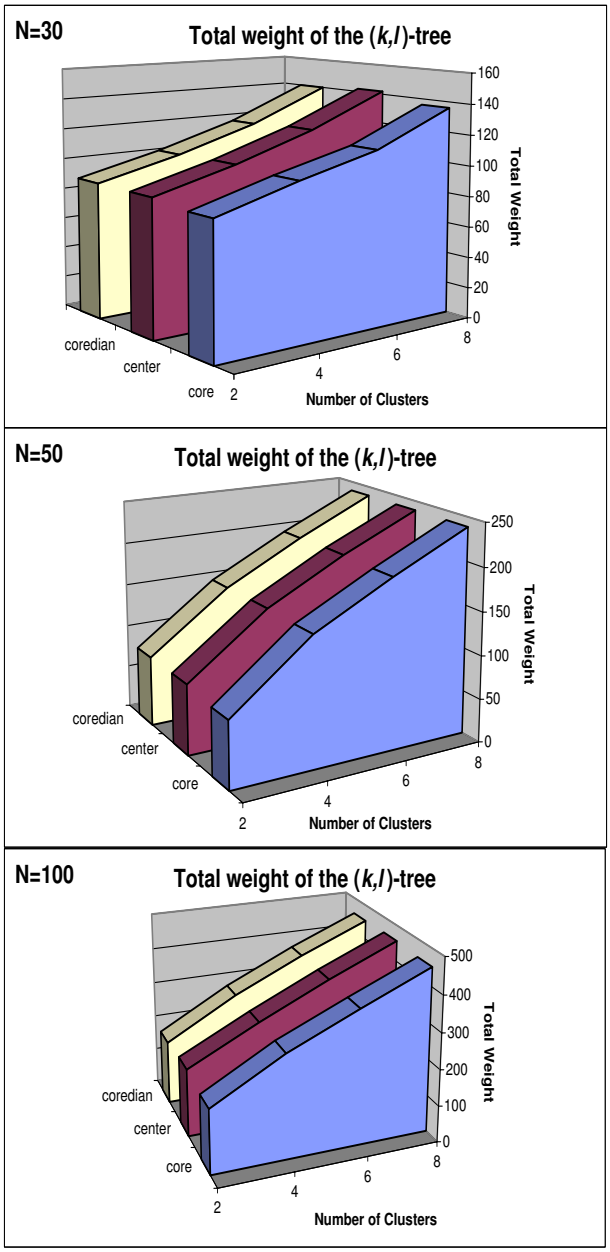


Fig. 9. Total weight of the (k, l) trees, n= 30, 50, 100

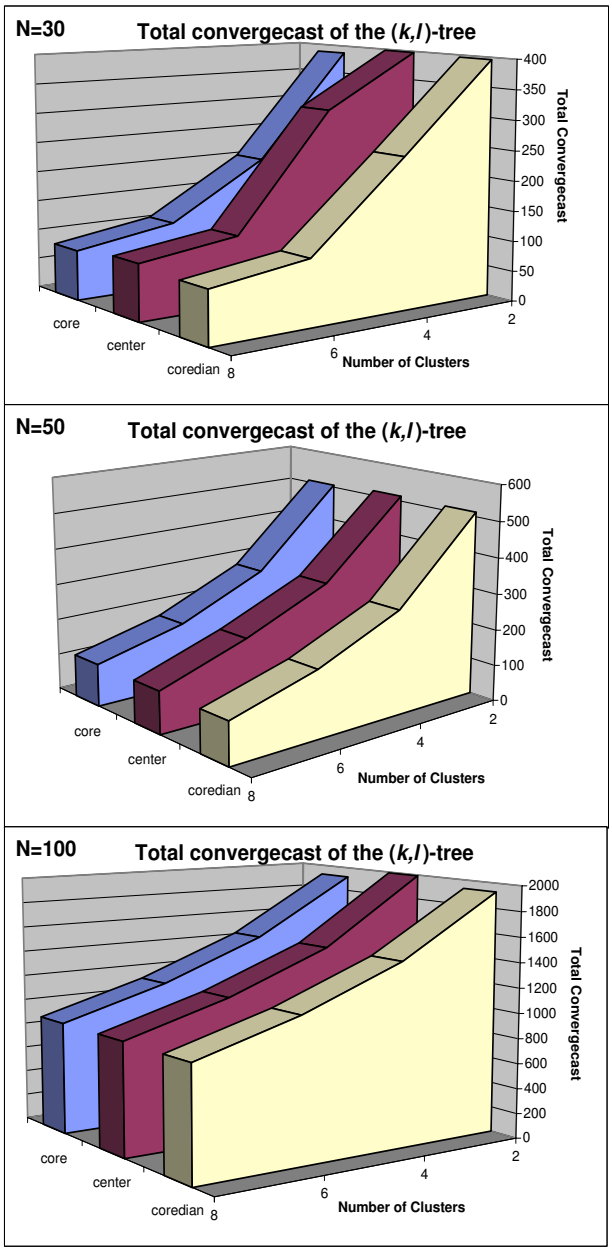


Fig. 10. Total convergecast of the (k, l) trees, n= 30, 50, 100

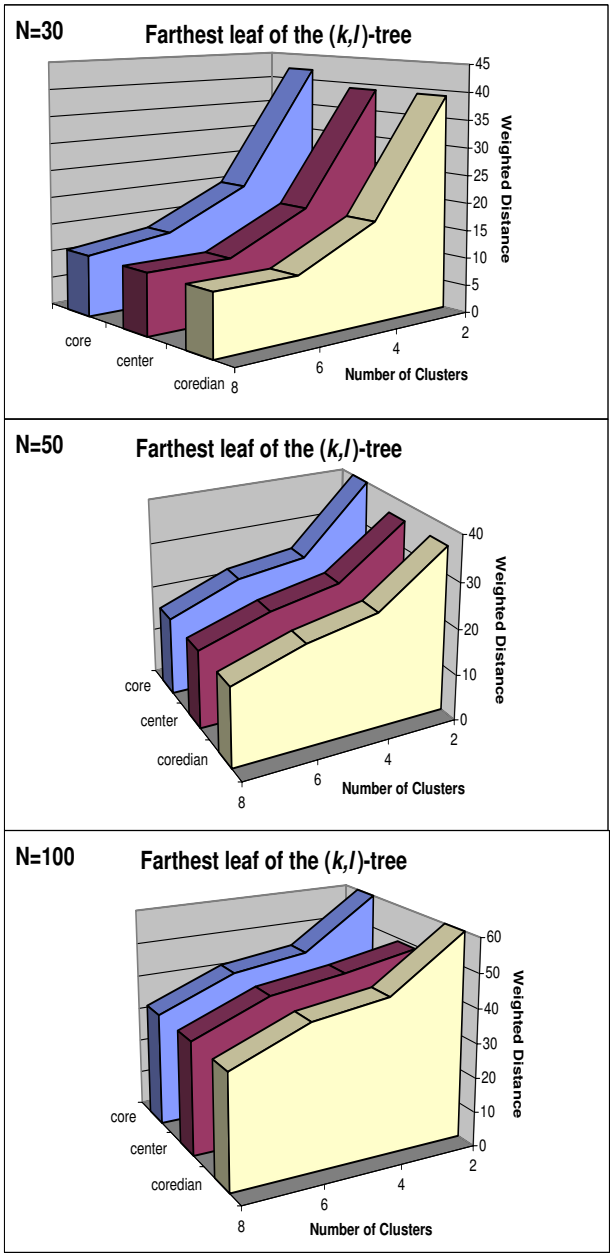


Fig. 11. Farthest leaf from the (k, l) trees, n= 30, 50, 100