Power Optimization in Fault-Tolerant Topology Control Algorithms for Wireless Multi-hop Networks

MohammadTaghi Hajiaghayi, Nicole Immorlica, Vahab S. Mirrokni

Abstract— **In ad hoc wireless networks, it is crucial to minimize power consumption while maintaining key network properties. This work studies power assignments of wireless devices that minimize power while maintaining k-fault tolerance. Specifically, we require all links established by this power setting be symmetric and form a k-vertex connected subgraph of the network graph. This problem is known to be NP-hard. We show current heuristic approaches can use arbitrarily more power than the optimal solution. Hence, we seek approximation algorithms for this problem. We present three approximation algorithms. The first algorithm gives an $O(k\alpha)$-approximation where α is the best approximation factor for the related problem in wired networks (the best α so far is $O(\log k)$.) With a more careful analysis, we show our second (slightly more complicated) algorithm is an $O(k)$-approximation. Our third algorithm assumes that the edge lengths of the network graph form a metric. In this case, we present simple and practical distributed algorithms for the cases of 2- and 3-connectivity with constant approximation factors. We generalize this algorithm to obtain an $O(k^{2c+2})$-approximation for general k-connectivity ($2 \leq c \leq 4$ is the power attenuation exponent). Finally, we show that these approximation algorithms compare favorably with existing heuristics. We note that all algorithms presented in this paper can be used to minimize power while maintaining k-edge connectivity with guaranteed approximation factors. Recently, different set of authors (see e.g. [2], [5], [6], [17], [20], [25], [29], [30], [31], [37], [39], [18]) used the notion of k-connectivity and the results of this paper to deal with the fault-tolerance issues for static wireless network settings.**[1]
keywords: Topology control, ad hoc networks, power conservation, graph model, graph properties, approximation algorithms

I. Introduction

In recent years, ad hoc wireless networks have become an increasingly common and important phenomenon due to their applications in battlefield communication and disaster relief communication ([12], [34]). These networks face a variety of constraints that do not occur in wired networks. Nodes in a wireless network are typically battery-powered, and it is expensive and sometimes infeasible to recharge the device. Thus research efforts have focused on designing minimum power algorithms for typical network tasks such as broadcast transmission ([10], [27], [36]) and connectivity/fault-tolerance ([1], [2], [4], [7], [22], [26], [33]).

Ad hoc wireless networks consist of simple mobile devices, or *nodes*, which communicate via radio transmitters. A node can vary its transmission range by varying the power with which it transmits a message. A *range assignment* for a network is a power setting for each node, and the cost of a range assignment is either the average power setting or the maximum power setting in that assignment. The network design goal in this setting is to minimize power consumption while maintaining key network properties such as connectivity. As all messages from a single node in the network reach all nodes within its transmission range, the cost of transmitting a message is not dependent on the number of receiving nodes, but simply a function of their maximum distance r from the sending node. While there may be a cost associated with receiving a message, this cost is typically minimal and we ignore it in this paper. If the network is *multi-hop*, i.e., nodes can forward messages as well as initiate them, then it is possible to maintain network connectivity without every node transmitting at maximum power. This allows us to seek power-optimal range assignments for connectivity and related network issues.

Previous works have addressed the issue of power-optimal range assignments that maintain connectivity. As this problem is NP-hard even in the Euclidean plane [13], some approaches concentrate on heuristics. Rodoplu and Meng [34], Wattenhofer et al. [38], and Li et al. [26] develop cone-based local heuristics for connectivity. In this heuristic, each node increases transmission power until some local conditions are met. This algorithm has a clear advantage of being localized; however we show that the power consumption of the resulting solution can be arbitrarily worse than that of the optimal solution. Other papers have concentrated on providing provable approximation algorithms. Kirousis et al. [22] show the minimum spanning tree of the network graph yields a 2-approximation algorithm for minimum average power connectivity. Calinescu et al. [7] improve the approximation factor to 1.69 with a Steiner tree-based algorithm and also provide a more practical 1.875-approximation algorithm.

A natural generalization of the connectivity requirement is k-connectivity or k-fault tolerance. In a k*-fault tolerant* network, communication should not be disrupted even when up to $k-1$ nodes fail. These networks also provide multi-path redundancy for load balancing or transmission error tolerance. As power-optimal connectivity is NP-hard, power-optimal k-fault tolerance is NP-hard as well. Previous works have investigated heuristics for this problem. Ramanathan and Rosales-Hain [33] consider the special case of 2-fault tolerance and provide a centralized spanning tree heuristic for minimizing the maximum transmit power in this case. Bahramgiri et al. [2] generalize the cone-based local heuristic of Wattenhofer et al. [38], [26] in order to solve the general k-fault tolerant setting. However, both of these works are heuristics and do not have provable bounds on the solution cost. For the heuristics due to Wattenhofer et al. [38], [26] and Bahramgiri et al. [2], we show there are examples for which these heuristics perform arbitrarily worse than the optimal solution. It was recently brought to our attention that Lloyd et al. [28] independently studied algorithms with provable approximation guarantees for this problem. They present a general algorithm which they prove gives an 8-approximation for 2-fault tolerance, but they do not consider general k-fault tolerance.

This work investigates k-fault tolerant range assignments with the objective of minimizing average power. We present

A preliminary version of this paper appeared in the proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MOBICOM 2003).

MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge, MA 02139, Emails: {hajiagha,nickle,mirrokni}@theory.csail.mit.edu.

three approximation algorithms for this problem. The first two algorithms with approximation factors of $O(k \log k)$ and $O(k)$, although centralized, work even in graphs with general edge lengths. For graphs whose edge lengths form a metric, we present simple and practical distributed algorithms with constant approximation factors for the cases of 2- and 3-connectivity. In addition, we generalize this algorithm to obtain an $O(k^{2c+2})$-approximation for k-connectivity ($2 \leq c \leq 4$ is the power attenuation exponent). All algorithms in this paper can be extended to approximation algorithms for power-optimal k-edge connectivity. However, since we are primarily concerned with static settings, node failures (due to lack of power) are more common than edge failures. Therefore, we focus on vertex connectivity in this paper.

Finally, we note that after the conference proceeding version of this paper, Recently, different set of authors (see e.g. [2], [5], [6], [17], [20], [25], [29], [30], [31], [37], [39], [18]) used the notion of k-connectivity and the results of this paper to deal with the fault-tolerance issues for static settings. In particular, $O(\log^4 n + \alpha)$-approximation and $O(\log n + \alpha)$-approximation algorithms for the minimum power k-connected subgraph problem where α is the best approximation factor known for the minimum-cost k-connected subgraph problem, respectively [17], [18]. Currently, $\alpha = O(\ln k)$ for $n \geq 2k^2$ [9] and $\alpha = O(\ln^2 k \cdot \min\{\frac{n}{n-k}, \frac{\sqrt{k}}{\ln n}\})$ [23] otherwise.) Moreover, it has been observed that an β-approximation algorithm for minimum-power k-connected subgraph problem implies a 2β-approximation algorithm for the minimum-cost k-connected subgraph problem.

In Section II, we formally define the k-fault tolerant topology control problem and the underlying wireless network model. In Section III, we discuss two plausible approaches to this problem and provide lower bounds for the approximation factors of these approaches in the worst case. In Section IV, we present our approximation algorithms and prove the approximation factors. In Section V, we evaluate the performance of our approximation algorithms by comparing them to existing heuristics. Finally, in Section VI, we conclude with a discussion of future research directions.

II. Preliminaries and Model

In this paper, we are mainly interested in static symmetric multi-hop ad hoc wireless networks with omni-directional transmitters. This is the model considered by Blough et al. [4], Calinescu et al. [7], Kirousis et al. [22], and others in their works on connectivity. Algorithms developed for this model have important practical considerations. Many existing routing protocols are easily accommodated in this model as links are established in both directions. Furthermore, many of the restrictions imposed by this model can be relaxed at the cost of additional communication. We briefly restate the model here.

Ad hoc wireless networks consist of a set of mobile devices equipped with radio transmitters and receivers. Each radio transmitter is assigned a power setting and an orientation that define the reception area of its transmissions. Oriented transmitters save power by emitting signals in a particular direction. In practice, most transmitters are omni-directional, and this is the model we assume for this paper (and in fact, all cited works assume this model as well). In ideal settings, an omni-directional transmission of power r^2 will reach all receivers within a sphere of radius r. However, interference from other transmissions and background noise may attenuate this signal. Typically, a node must transmit a message at power r^c, $2 \leq c \leq 4$, to attain a transmission range of distance r. The particular exponent c, referred to as the *power attenuation exponent*, depends on the environmental conditions, and may vary from device to device.

We consider multi-hop networks, or networks in which devices cooperate to route each others' messages. In this way, the overall power usage of the network can be minimized. For example, consider the problem of broadcasting a message from device u and assume the transmission power grows like the range squared for all devices (i.e., $c = 2$). Let devices u, v, and w be positioned at the vertices of a triangle such that the distance between u and v is 5 meters, v and w is 6 meters, and u and w is 10 meters. Then if u wants to send the message to w directly, it will take 100 units of power, but by allowing v to forward the message to w, the system uses just 61 units of power.

In most of this paper, we make the further assumptions that our networks are *static* and that all established links are *bidirectional* or *symmetric*. In a static network, the devices are stationary. If a device moves, the range assignment must be recalculated in order to maintain desired network properties. In the symmetric link model, if a device u is assigned to receive transmissions from a device v, then it must also be able to transmit to device v. Although this restriction can theoretically be relaxed, in practice symmetric links greatly simplify routing protocols and thus are desirable.

A wireless network can be modelled as a graph $G(V, E)$ where V is the set of mobile devices and $E \subseteq V^2$ is the set of pairs of devices between which communication is possible. Note E does not necessarily equal V^2 as maximal transmission ranges and environmental conditions may impose constraints on possible pairs of communicating nodes. In general, this graph may be directed, but our symmetric link constraint allows us to eliminate all uni-directional edges. Typically, an edge (i, j) is assigned a distance $d(i, j)$, representing the distances between devices i and j, and cost $p(i, j)$, representing the power setting i and j must use to transmit to each other. In most cases, the edge distances satisfy the triangle inequality, and we refer to these graphs as *geometric graphs*. In the case of a uniform power attenuation exponent, this also implies a relationship between edge costs. In some cases, we place no assumption on the relationship between edge costs, and we refer to these graphs as *general graphs*.

A *range assignment* $\mathcal{R}$ is an assignment of power settings $\mathcal{R}(i)$ to devices i. A subgraph $H = (V, E')$ where $E' \subseteq E$ of the network graph $G = (V, E)$ defines a range assignment $\mathcal{R}_{E'}$ where $\mathcal{R}_{E'}(i) = \max_{\{j \mid (i,j) \in E'\}} p(i, j)$. The cost of a subgraph is the average (or, equivalently, the total) power assigned in its corresponding range assignment. We use the term *power cost* for this quantity to differentiate between this cost and the so-called *normal cost* of a graph, i.e., the cost function which captures the notion of bandwidth usage and which wired network designers typically attempt to minimize. More formally,

Definition 1: In an undirected graph $G = (V, E)$ with edge costs $p(i, j)$, the *power cost* of G is

$$P(G) = \sum_{i \in V} \max_{\{j \mid (i,j) \in E\}} p(i, j).$$

Definition 2: In a graph $G = (V, E)$ with edge costs $p(i, j)$, the *normal cost* of G is

$$C(G) = \sum_{(i,j) \in E} p(i, j).$$

Using these definitions, we can define two main problems. The problem we study in this paper is the undirected minimum power k-vertex connected subgraph problem. A *k-vertex connected* graph has k vertex-disjoint paths between every pair of vertices, or equivalently, remains connected when any set of at most $k-1$ vertices is removed. Hence the subgraphs we find are k-fault tolerant.

Definition 3: An *Undirected Minimum Power k-Vertex Connected Subgraph (k-UPVCS)* of a graph $G=(V,E)$ is a k-vertex connected subgraph $H=(V,F)$, $F\subseteq E$, such that $P(H)\leq P(H')$ for any k-vertex connected subgraph $H'=(V,F')$, $F'\subseteq E$.

This problem is closely related to the standard k-vertex connected subgraph problem which corresponds to k-fault tolerance in wired networks.

Definition 4: An *Undirected Minimum Cost k-Vertex Connected Subgraph (k-UCVCS)* of a graph $G=(V,E)$ is a k-vertex connected subgraph $H=(V,F)$, $F\subseteq E$, such that $C(H)\leq C(H')$ for any k-vertex connected subgraph $H'=(V,F')$, $F'\subseteq E$.

When k is not specified, it is understood that $k=1$. Both the k-UPVCS and k-UCVCS problems are NP-hard, and thus our work as well as previous works have focused on finding approximations for these problems. An α-approximation algorithm is a polynomial time algorithm whose solution cost is at most α times the optimal solution cost.

The k-UCVCS problem has been well-studied. These results are central to our work, for, as in the case of connectivity, a solution to the k-UCVCS problem turns out to be an approximation for the k-UPVCS problem. The problem has been considered both for general and geometric graphs. Frank and Tardos [15] and Khuller and Raghavachari [21] were among the first authors who worked on the k-UCVCS problem. The best known approximation for general graphs with at least $6k^2$ vertices is an $O(\log(k))$-approximation due to Cheriyan et al. [9]. Their results use an iterative rounding method on a linear programming relaxation. Kortsarz and Nutov [23] study combinatorial algorithms for different variants of the problem. They introduce a k-approximation algorithm for general graphs (without any condition on the number of vertices) and a $(2+\frac{k-1}{n})$-approximation for graphs with metric costs. They also consider the special cases of $k\leq 7$ and present a $\lceil\frac{k+1}{2}\rceil$-approximation. We use ideas from their algorithm to design an $O(k)$-approximation for the k-UPVCS problem.

We also consider edge failures and prove similar guarantees for power-optimum k-edge-connected subgraphs. We adapt the centralized algorithm to work in this case. Our distributed algorithm also gives the same performance guarantee for k-edge connected subgraphs.

III. Previous Approaches

As the k-UPVCS problem is NP-hard, an exact solution is infeasible. One line of previous work has focused on approximate solutions. Approximations are often obtained via a linear programming representation of the problem. However, we show that for the k-UPVCS problem, a linear programming approach is unlikely to yield a good approximation in the worst case. Another line of work has focused on providing heuristics which work well in practice. However, heuristics do not have provably good solutions, and in fact, we can show that in the worst case, the current k-UPVCS heuristics perform poorly.

It is important to note that the results in this section make claims about the worst case performance of the proposed algorithms. This does not imply poor behavior on average or in typical situations. The typical cases can only be analyzed through experiments, and those results appear in Section V.

A. Linear Programming Approach

Many known approximation algorithms are based on linear programming approaches. The best known k-UCVCS approximation algorithm (an $O(\log k)$-approximation algorithm by Cheriyan et al. [9]) is such an example. In these algorithms, the problem is formulated as an integer program. Then, the fractional solution of the linear programming relaxation is rounded to an integral solution and its value is used as a lower bound in the analysis. The integrality gap of linear programming formulation, i.e., the ratio between the optimal values of the integral and fractional solutions, is a lower bound on the achievable approximation factor. One might hope for a linear-programming-based approximation algorithm for k-UPVCS with performance similar to that of k-UCVCS. However, we show that the natural linear programming formulation for the k-UPVCS problem has an integrality gap of $\Omega(\frac{n}{k})$, implying that there is no approximation algorithm based on this linear program with an approximation factor better than $\Omega(\frac{n}{k})$.

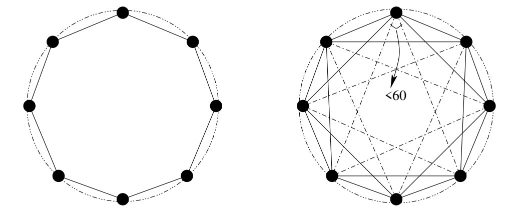
We present a natural linear programming formulation of this problem introduced by Cheriyan et al. [9]. We assign a zero-one variable x_e to each edge e indicating whether edge e is in the k-connected subgraph $G'=(V,E')$ of the input graph $G=(V,E)$. The cost of subgraph G' is $\sum_{v\in V}p_v$ where p_v is the maximum power of all edges adjacent to v in G', i.e., $p_v\geq p(u,v)x_{(u,v)}$ for all $(u,v)\in E$. To guarantee that solutions to the integer program represent k-connected subgraphs, we introduce a constraint ensuring that there are k vertex-disjoint paths between every pair of vertices (in fact, every pair of sets). Define a *setpair* $S=(T,H)$ to be any pair of two disjoint nonempty subsets T and H of vertices. The idea is that any such pair of sets must have k vertex disjoint paths between them in order for G to be k-vertex connected. Let $\delta(S)=\delta(T,H)$ be the set of all edges with one endpoint in T and the other in H. There are $n-|H\cup T|$ vertices outside H and T that can participate in paths between H and T. Thus, there are at most $n-|H\cup T|$ vertex-disjoint paths between H and T that leave $H\cup T$, and so there must be at least $k-(n-|H\cup T|)$ edges in $\delta(T,H)$. The setpair linear programming relaxation is as follows:

$$
\begin{array}{ll}
\text{minimize} & \sum_{v\in V} p_v \\
\text{subject to} & \sum_{e\in\delta(S)} x_e \geq \max\{0, k-(n-|H\cup T|)\} \\
& \qquad \text{for all setpairs } S=(T,H) \\
& p_v \geq p(u,v)x_{vu} \\
& \qquad \text{for all } v\in V,\ (u,v)\in E \\
& 0\leq x_e\leq 1 \\
& \qquad \text{for all } e\in E
\end{array}
$$

The above discussion shows that these constraints are necessary for G' to be a k-connected subgraph. To see that they are also sufficient, we refer the reader to the result of Cheriyan et al. [9].

Lemma 1: If $n\geq 2k$, the integrality gap of the above linear programming is $\Omega(\frac{n}{k})$.

Proof: Consider the complete graph. Assume all edge costs are equal to one. A feasible fractional solution of the linear program is $x_e=\frac{k+1}{n}$ and $p_v=\frac{k+1}{n}$. In order to check that this solution is feasible, we need to prove that for any setpair $S=(T,H)$, $\sum_{e\in\delta(T,H)}x_e=|H||T|\frac{k+1}{n}\geq k-(n-|H\cup T|)$. As $|H||T|\geq|H\cup T|-1$, it is sufficient to show that $k-(n-|H\cup T|)\leq\frac{k+1}{n}(|H\cup T|-1)$. We use the assumption that $2k\leq n$ and the observation that $|H\cup T|\geq(k+1)$ as $k-(n-|H\cup T|)>0$. For clarity of presentation, let $x=|H\cup T|$



(a)The optimum 2-connected subgraph

(b)The output of CBTC for $k = 2$ in which each node increases its power until the angle is less than $\frac{2\pi}{3\times 2} = 60^0$

Fig. 1. The illustration of CBTC lower bound

and note $x \leq n$. Then,

$$\begin{aligned}
k-(n-x) &\leq k-\frac{2k}{n}(n-x) \\
&= \frac{2k}{n}x-k \\
&= (\frac{k}{n}x-k)+\frac{k}{n}x \\
&\leq \frac{k}{n}x \\
&\leq \frac{kx}{n}+\frac{x-(k+1)}{n} \\
&= \frac{k+1}{n}(x-1).
\end{aligned}$$

Here the first inequality follows from our assumption in the statement of the lemma, the second one follows since $x \leq n$ and the third one follows since $x \geq k+1$. As this solution is feasible, the cost of the optimal fractional solution is at most $n\frac{k+1}{n} = k+1$. In the optimal integral solution, there should be at least one edge incident to each vertex; thus the cost of an optimal integral solution is at least n since $p_v \geq 1$ for all v. Therefore, the integrality gap is at least $\frac{n}{k+1} = \Omega(\frac{n}{k})$. ■

B. Heuristic-Based Approach

One approach for the k-UPVCS problem is heuristic-based. Bahramgiri et al. [2] show that the cone-based topology control algorithm of Wattenhofer et al. [38], [26] for UPVCS can be extended to an algorithm for k-UPVCS. In the following, we state this algorithm, and then we construct examples which demonstrate that the approximation factor for this algorithm is at least $\Omega(\frac{n}{k})$.

In the *cone-based topology control (CBTC)* algorithm, each node increases its power until the angle between its consecutive neighbors is less than some threshold. In the following, we present a brief description of this algorithm. For details of CBTC and how to implement it in a distributed fashion, we refer to Wattenhofer et al. [38], [26]. Node u sends a **Hello** message to every other node v using power p. Upon receiving a **Hello** message from node u, node v replies with an **Ack** message. After gathering the **Ack** messages, node u constructs the set of its neighbors, $N(u)$, along with a set of vectors indicating the direction of each neighbor. Node u increases its power until the angle between any pair of adjacent neighbors is at most α for some fixed α. Now, let $N_\alpha(u)$ be the final set of neighbors computed by a node u and $E_\alpha = \{(u,v)|v \in N_\alpha(u) \text{ and } u \in N_\alpha(v)\}$. Output graph $G_\alpha = (V, E_\alpha)$.

Wattenhofer et al. [38] have shown that for $\alpha \leq \frac{2\pi}{3}$, the subgraph G_α produced by this algorithm is connected if and only if G is connected. Li et al. [26] show that the theorem does not hold necessarily for $\alpha > \frac{2\pi}{3}$ and they also extend the result to the directed case. Bahramgiri et al. [2] generalize the first result for k-connected subgraphs in the following way: for $\alpha \leq \frac{2\pi}{3k}$, G_α is k-connected if and only if G is k-connected. They also show that the theorem does not hold necessarily for $\alpha > \frac{2\pi}{3k}$ if k is even and $\alpha > \frac{2\pi}{3(k-1)}$ if k is odd. Although this heuristic-based algorithm is very practical in a distributed mobile setting, it does not have a reasonable approximation guarantee. We show that this algorithm's solution can be as much as $\frac{n}{k}$ times the optimal one.

Theorem 1: There are examples for which the approximation factor of the CBTC algorithm for k-connectivity ($k \geq 1$) is at least $\Omega(\frac{n}{k})$, i.e., the ratio between the power of the output of CBTC and the minimum power k-connected subgraph is $\Omega(\frac{n}{k})$.

Proof: Consider the geometric graph G with n nodes evenly spaced around a circle. Figure 1 shows an example when the network has 8 nodes and compares the optimal 2-connected subgraph with the output of CBTC for $k = 2$. In the CBTC algorithm, each node increases its power until the angle between any two consecutive neighbors is at most $\frac{2\pi}{3k}$. As a result, each vertex is connected to $\frac{n}{2} - \frac{n}{3k}$ vertices in each half of the cycle which yields a regular graph of degree $\frac{n}{2} - \frac{n}{3k} = \Omega(n)$. The power of each node is the length of the chord which corresponds to the arc of length $(\frac{1}{2} - \frac{1}{3k})$ of the perimeter. More precisely, the length of this chord is $2R\sin((\frac{1}{2} - \frac{1}{3k})\pi)$.

A feasible solution is to connect each vertex to $\lceil\frac{k}{2}\rceil$ neighbors on each side. The resulting graph, a Harary graph, is k-connected. The power of each node is the length of the chord corresponding to the arc of length $\frac{k}{n}$ of the perimeter. The length of this chord is $2R\sin((\lceil\frac{k}{2}\rceil)\frac{\pi}{n})$. Thus, the ratio between the output of CBTC and the optimum solution is $\Omega(\frac{n}{k})$ when n is large enough and k is small since $\sin(\lceil\frac{k}{2}\rceil\frac{\pi}{n}) \simeq (\lceil\frac{k}{2}\rceil\frac{\pi}{n})$ and $\sin((\frac{1}{2} - \frac{1}{3k})\pi) = \Theta(1)$, i.e., a constant. This example shows that the approximation factor of CBTC is at least $\Omega(\frac{n}{k})$. ■

IV. APPROXIMATIONS

In this section, we present several approximation algorithms for the k-UPVCS problem. We first discuss the relationship between the normal cost and the power cost of a graph, from which an $O(k\alpha)$-approximation for the k-UPVCS problem immediately follows where α is the best approximation factor for the k-UCVCS problem. The k-UPVCS approximation algorithm simply uses the k-UCVCS approximation algorithm as a black box subroutine. We observe that we can actually improve our approximation factor by analyzing a particular k-UCVCS algorithm more precisely.

Although this algorithm yields the best approximation factor known and works even for general graphs, it has the disadvantage of having a high communication overhead. Hence, we also present a simple approximation algorithm with a slightly worse approximation factor which is applicable to geometric graphs and is distributed.

A. Global Approximation

As mentioned above, the normal cost and power cost of graphs are closely related. In fact, Kirousis et al. [22] exploit this relationship to obtain a 2-approximation for the UPVCS problem via a solution for the UCVCS, or minimum spanning tree, problem. As we use these relationships in many of our algorithms and proofs, we present them succinctly here. Lemma 2 states that the power cost of a graph is at most twice the normal cost of the graph. Lemma 3 observes that, for trees, we can also upper bound the normal cost by the power cost. Finally, Lemma 4 uses the preceding two lemmas to show that a forest decomposition of a graph implies a relationship between its normal and power cost.

Lemma 2: For any graph G, $P(G) \leq 2C(G)$.

Proof: The proof is straightforward from the following inequalities.

$$P(G) = \sum_{v \in V} \max_{\{u \mid (u,v) \in E\}} p(u,v)$$
$$\leq \sum_{v \in V} \sum_{(u,v) \in E} p(u,v) = 2 \sum_{e \in E} p_e = 2C(G)$$

■

Lemma 3: For any tree T, $C(T) \leq P(T)$.

Proof: Root T at an arbitrary vertex r. Note the power of each node is at least the cost of its parent edge. The statement follows. ■

Lemma 4: For any graph G which can be written as a union of t forests, $C(G) \leq tP(G)$.

Proof: Write $G = \cup_{i=1}^{t} F_i$ for forests F_i. Then

$$C(G) \leq \sum_{i=1}^{t} C(F_i) \leq \sum_{i=1}^{t} P(F_i) \leq \sum_{i=1}^{t} P(G) = tP(G)$$

where the second inequality follows from Lemma 3 and the third follows since each forest is a subgraph of G. ■

Using these lemmas, we can show that a k-UCVCS subgraph G_C is in fact a $2k$-approximation to a k-UPVCS subgraph G_P. Recall that an edge (u, v) of a k-vertex connected graph H is *critical* if $H - (u, v)$ is not k-vertex connected. Graph G is *critically* k-vertex connected if and only if G is k-vertex connected and all edges of G are critical. We use the following theorem to find a forest decomposition of a critical k-vertex connected graph.

Theorem 2 (Mader [35]) In a k-vertex connected graph, a cycle consisting of critical edges must be incident to at least one node of degree k.

Lemma 5: Any critical k-vertex connected graph, G, can be written as the union of k forests.

Proof: Since all edges of G are critical, by Theorem 2, any cycle in G is incident to at least one node of degree k. Let F_0 be a spanning tree of graph G. Let F_i be a (maximal) spanning forest of graph $G - F_0 - F_1 - \ldots - F_{i-1}$ for $1 \leq i \leq k-2$. Let $F_{k-1} = G - F_0 - F_1 - \ldots - F_{k-2}$. We need to prove that graph F_{k-1} is a forest. Graph G is k-connected, thus there are at least k vertex-disjoint paths from any vertex to any other vertex. Therefore, graph $G - F_0 - F_1 - \ldots - F_i$ for $0 \leq i \leq k-3$ is connected. As a result, each forest F_i for $0 \leq i \leq k-2$ is a tree. Thus, the degree of each vertex in each graph F_i is at least 1. Therefore, the degree of each vertex in $G - F_0 - F_1 - \ldots - F_{i-1}$ for $1 \leq i \leq k-3$ is at most its degree in G minus i. Assume for contradiction that F_{k-1} has a cycle C. The degree of each vertex of C in F_{k-1} is at least 2. Thus, the degree of each vertex of C in G is at least $k+1$. This contradicts the fact that there should be a vertex of degree k in cycle C in graph G (by Theorem 2). Hence, graph F_{k-1} does not have any cycle. Thus, G can be written as the union of k forest $F_0, F_1, \ldots, F_{k-1}$. ■

We can now see that a k-UCVCS subgraph G_C is in fact a $2k$-approximation to a k-UPVCS subgraph G_P:

$$P(G_C) \leq 2C(G_C) \leq 2C(G_P) \leq 2kP(G_P)$$

where the last inequality follows from the fact that there is a subgraph of G_P that is critically k-vertex connected which has the same power-cost as G_P.

Theorem 3: The power of a k-UCVCS subgraph is at most $2k$ times the power of a k-UPVCS subgraph.

Unfortunately, we can not solve the k-UCVCS problem exactly. However, it follows from Theorem 3 that an α-approximation algorithm for the k-UCVCS problem is a $2\alpha k$-approximation for the k-UPVCS problem. In general graphs, Cheriyan et al. [9] give a $\log k$-approximation algorithm for the k-UCVCS problem for general graphs with at least $6k^2$ vertices, implying an $O(k \log k)$-approximation algorithm for the k-UPVCS problem in such graphs. Kortsarz and Nutov [23] give a k-approximation algorithm with no assumption on the size of the graph, implying an $O(k^2)$ algorithm for the k-UPVCS problem in any graph. In geometric graphs, the triangle inequality on edge lengths implies that the edge costs satisfy a *weak* triangle inequality (see Corollary 1 in Section IV-B). In other words, edge costs c_{ij} satisfy $c_{ik} \leq 2^{c-1} \cdot (c_{ij} + c_{jk})$ where $2 \leq c \leq 4$ is the power attenuation exponent. A direct extension of the results in Khuller et al. [21] shows $\alpha = 2 + 2^c(k-1)/n$ for the k-UCVCS problem in these graphs, implying an $O(k)$-approximation for the k-UPVCS problem.

It is worth mentioning that our approach for k-vertex connectivity can also be applied to obtain an $O(k)$-approximation for *k-edge connectivity*, another important concept in fault-tolerant network design. Graph G is k-edge connected if it remains connected after deleting any set of $k-1$ edges. Formally, we can define the *undirected minimum power k-edge connected subgraph* (k-UPECS) and the *undirected minimum cost k-edge connected subgraph* (k-UCECS) similar to the k-UPVCS problem and the k-UPVCS problem, respectively. It turns out that the k-UCECS problem is easier to approximate than the k-UCVCS problem. In fact, constant factor approximations are known even for general graphs ([24], [19]). Khuller and Vishkin [24] first give a 2-approximation for the k-UCECS problem. Later, Jain [19] uses an iterative rounding method to achieve a 2-approximation algorithm for this problem and its generalization. Here, we can design a $2\alpha k$-approximation for the k-UPECS problem from an α-approximation for the k-UCECS problem. As a result we achieve a $4k$-approximation for the k-UPECS problem using 2-approximations for the k-UCECS problem ([24], [19]). The proof is the same as the proof for vertex connectivity except that we need to reprove Lemma 5 for critical k-edge connected graphs.

Lemma 6: Any critical k-edge connected graph, G, can be written as the union of k forests.

Proof: We use the following fact from Frank [14]: Given a k-edge connected graph G, let F_1 be a maximal forest in G and F_i ($2 \leq i \leq k$) be a maximal forest in $G - F_1 - F_2 - \ldots - F_{i-1}$. Then, the union of $F_1, \ldots, F_k$ is k-edge connected [14]. Since G is critically k-edge connected and the union of F_i's is a k-edge connected subgraph of G, $F_1, \ldots, F_k$ should cover all the edges of G. ■

Returning to our algorithm for the k-UPVCS problem, one can see that we use an algorithm for the k-UCVCS problem as a black box. We can improve the approximation factor if we actually analyze the internals of the underlying k-UCVCS algorithm. We follow the k-approximation algorithm introduced by Kortsarz and Nutov [23] to approximate k-UCVCS subgraphs. Their algorithm, which we refer to as Algorithm Global k-UPVCS, first finds a 2-approximation to the cheapest normal cost *k-outconnected subgraph* H rooted at an arbitrary vertex r using a subroutine which we refer to as $A(r, G)$. A k-outconnected subgraph rooted at r is a subgraph with k internal

Algorithm **Global** k-**UPVCS**($G(V,E)$)
// choose arbitrary root r
$r \in V$
// find k-outconnected subgraph H
// and covering set F' using subroutine $A(r,G)$
$H, F' \leftarrow A(r,G)$
for $(u,v) \in F'$
  // find k vertex disjoint paths F_{uv} with the cheapest
  // (normal) cost from u to v in G
  $F_{uv} \leftarrow k$ vertex disjoint paths with cheapest cost
end
// replace edges in cover by the sets of
// cheapest k vertex disjoint paths
for $(u,v) \in F'$
  $H \leftarrow H \cup F_{uv}$
end
output $G_k = H$

Fig. 2. A formal description of Algorithm Global k-UPVCS

k	α
2	8
3	16
4	20
5	24
6	32
7	36

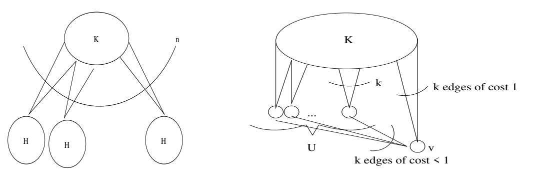
TABLE I
IMPROVED APPROXIMATION FACTOR α OF ALGORITHM GLOBAL k-UPVCS FOR $k \leq 7$

vertex disjoint paths between r and every other vertex $v \in G$. They show that such a graph has a *cover* of size at most $k-2$ where a cover is a set of edges that can be added to a graph to make it k-connected. The algorithm computes a $k-2$ cover F' for H and finally replaces each edge $(u,v) \in F'$ by the k vertex disjoint paths from u to v with the cheapest (normal) cost (as they mention, these paths can be found in polynomial time via a min-cost k-flow algorithm). One can easily observe that adding these k disjoint paths instead of each edge of the cover preserves k-connectivity. For a formal description of this algorithm, see Figure 2.

We show that this algorithm of Kortsarz and Nutov is in fact an $8(k-1)$-approximation for the k-UPVCS problem in general graphs. For the special cases of $k \in \{4,5\}$ and $k \in \{6,7\}$, Kortsarz and Nutov [23] show the covering set of a k-outconnected graph has size 1 and 2 respectively, implying better approximations in these cases. Table I lists the approximation factor of this algorithm for various k, taking into account these special cases. For the important case of $k=2$, this algorithm yields an 8-approximation. Lloyd et al. [28] independently obtained a different 8-approximation algorithm for the 2-UPVCS problem.

Theorem 4: Algorithm Global k-UPVCS returns a k-vertex connected subgraph G_k whose power cost is at most $8(k-1)$ times the power of a k-UPVCS subgraph for $k \geq 2$.

Proof: We decompose G_k into H and $F \equiv \cup_{uv} F_{uv}$ and bound the cost of each part separately. Let G_{opt} be a k-UPVCS subgraph. First we bound $P(H)$ in terms of $P(G_{opt})$. Let H_{opt} be the minimum normal cost graph that has k edge disjoint paths between r and each $v \in V - \{r\}$. We know $P(H) \leq 2C(H) \leq 4C(H_{opt})$ as $A(r,G)$ is a 2-approximation. Notice any k-vertex connected graph also has k edge disjoint paths between r and each $v \in V - \{r\}$. Therefore $C(H_{opt}) \leq C(G_k)$ for any k-vertex connected graph G. Thus $P(H) \leq 4C(G)$. Note that there is a subgraph G'_{opt} of G_{opt} which is critically k-connected and has the same power-cost, and so, by Lemma 5, we can decompose it into k forests. By Lemma 4, $C(G'_{opt}) \leq kP(G'_{opt}) = kP(G_{opt})$. Putting together these inequalities, we see $P(H) \leq 4C(G'_{opt}) \leq 4kP(G_{opt})$.



(a)External structure of G (b)Structure of subgraph H_i of G

Fig. 3. Structure of G

Now we bound $P(F)$ in terms of $P(G_{opt})$. We write F as a union of the $k-2$ sets of edges F_{uv} corresponding to the F_{uv} in the algorithm. Recall each F_{uv} is the minimum normal cost set of k vertex-disjoint paths between u and v where $(u,v) \in F'$. Now $P(F) \leq 2C(F) \leq 2\sum_{(u,v)\in F'} C(F_{uv})$. Let G_{uv} be the minimum power cost set of k vertex-disjoint paths between u and v. Then $C(F_{uv}) \leq C(G_{uv})$. Graph G_{uv} can be written as the union of two trees, $T_u = G_{uv} - \{v\}$ and $T_v = G_{uv} - \{u\}$, so by Lemma 4, $C(G_{uv}) \leq 2P(G_{uv})$. Now G_{opt} must contain k vertex disjoint paths between every pair of vertices, and so $P(G_{uv}) \leq P(G_{opt})$. Combining these inequalities, we see

$$\begin{aligned} P(F) &\leq 2 \sum_{(u,v)\in F'} C(F_{uv}) \\ &\leq 2 \sum_{(u,v)\in F'} C(G_{uv}) \\ &\leq 4 \sum_{(u,v)\in F'} P(G_{uv}) \\ &\leq 4 \sum_{(u,v)\in F'} P(G_{opt}) \\ &\leq 4(k-2)P(G_{opt}). \end{aligned}$$

Our final approximation factor is $P(G_k) \leq P(H) + P(F) \leq 8(k-1)P(G_{opt})$ as stated. ■

We show that, in a sense, this approximation factor is tight. In other words, a k-UCVCS subgraph can have power cost $O(k)$ times the power cost of a k-UPVCS subgraph. Consider the example graph G illustrated in Figure 3. Here we have n copies of a graph H_i which all share a common subgraph K_k, the complete graph on k nodes with zero-cost edges. Each graph H_i contains a set U_i of k nodes disjoint from K_k, all of which are connected to all the nodes in K_k by zero-cost edges. Finally, there is a special node v_i in H_i which is connected to all nodes in K_k by a set of cost 1 edges $F_{i,1}$ and to all nodes in U_i by a set of cost $1-\epsilon$ edges $F_{i,(1-\epsilon)}$ for some $\epsilon \in (0,1)$.

Note that $H = G - \{v_1, \ldots, v_n\}$ is a k-connected graph of cost zero. Thus any graph which includes k edges from each v_i to H will be a k-connected subgraph of G. As a k-connected subgraph of G must have minimum degree k, this sufficient condition is also necessary, and so the k-UCVCS subgraph of G is $G_C = H \cup_{i=1}^{n} F_{i,(1-\epsilon)}$. A similar reasoning shows $G_P = H \cup_{i=1}^{n} F_{i,1}$ is the k-UPVCS subgraph. Now we compute

Algorithm **Distributed** 2-**UPVCS**($G(V,E)$)
// *compute the minimum spanning tree*
$T_{\text{MST}} \leftarrow$ Algorithm MST($G(V,E)$)
for node $u \in T_{\text{MST}}$
  // *find neighbors of* u
  $N \leftarrow \{v | (u,v) \in T_{\text{MST}}\}$
  // *add arbitrary path connecting neighbors*
  label vertices in N in an arbitrary order
  $E \leftarrow E \cup \{(v_1,v_2),\ldots,(v_{|N|-1},v_{|N|})\}$
end

Fig. 4. A formal description of Algorithm Distributed k-UPVCS for $k=2$

the power costs of these two subgraphs. In G_C, each node in a set U_i has power cost $(1-\epsilon)$ and each special node v_i has power cost $(1-\epsilon)$. The nodes in the common substructure K_k have power cost 0. Thus

$$P(G_C) = nk(1-\epsilon) + n(1-\epsilon).$$

In G_P, each special node v_i has power cost 1 and all the nodes in the common subgraph K_k have power cost 1. However, the nodes in the U_i sets have power cost 0. Therefore,

$$P(G_P) = n(1) + k(1).$$

Taking the ratio as n goes to infinity and ϵ goes to zero, we see $P(G_C) = (k+1)P(G_P)$ in the limit. Thus the approximation factor of any algorithm that uses the k-UCVCS subgraph as a solution for the k-UPVCS problem is at least $O(k)$.

B. Distributed Approximation

In this section, we assume that our graph is geometric (i.e., the edge lengths satisfy the triangle inequality) and the power attenuation exponent is uniform. In other words, the cost of an edge e of length r_e is r_e^c for some c, $2 \le c \le 4$. As shown in Lemma 7, this implies that the edge costs satisfy a weak triangle inequality.

Lemma 7: If $x_0 \le \sum_{i=1}^k x_i$, then $x_0^c \le k^{c-1}\sum_{i=1}^k x_i^c$.

Proof: Dividing both sides of the inequality by k^c, we see

$$\left(\frac{x_0}{k}\right)^c \le \left(\frac{\sum_{i=1}^k x_i}{k}\right)^c \le \frac{\sum_{i=1}^k x_i^c}{k}$$

by the convexity of the function $f(x) = x^c$. ■

Corollary 1: In a geometric graph with edge lengths r_{ij}, the edge costs $p_{ij} = r_{ij}^c$ satisfy a weak triangle inequality:

$$\forall (i,j),(j,k),(i,k) \in E, p_{ik} \le 2^{c-1}\cdot(p_{ij}+p_{jk}).$$

For simplicity, we will first describe an algorithm for the 2-UPVCS problem. As Theorem 5 states, the algorithm uses just a constant factor more power than the optimal configuration. Our algorithm uses as a subroutine Algorithm MST, an algorithm for computing the minimum spanning tree of the input graph. It then adds a path amongst the neighbors of each node in the returned tree. See Figure 4 for a formal description.

This algorithm has the significant advantage that, after the computation of the minimum spanning tree, it is *distributed*, i.e., each node can compute its power setting with just a small number of messages to other nodes. In wireless networks with no central authority, global computations are quite expensive and so the low communication overhead of this algorithm makes it very attractive in practical settings. In addition, the low communication overhead of this algorithm makes it easier to implement in a mobile setting. Indeed, once the minimum spanning tree has been computed, each node just needs to know its neighbors and their neighbors in order to decide at what power to transmit. The minimum spanning tree itself can be computed by the distributed minimum spanning tree algorithm of Gallager et al. [16] in just $5n\log n + 2m$ messages (where $n = |V|$, the number of devices, and $m = |E|$, the number of valid communication links). The number of required messages can be reduced by finding an approximate minimum spanning tree, although this will affect the approximation factor of the resulting algorithm. Since we only need $O(n)$ messages once we have the minimum spanning tree, the overall number of messages is $O(n\log n + m)$.

Theorem 5: For any geometric graph G, Algorithm Distributed 2-UPVCS returns a 2-vertex connected subgraph G_2 whose power $P(G_2)$ is a $2(4\cdot 2^{c-1}+1)$-approximation of the power of a 2-UPVCS subgraph.

Proof: We use the fact that $P(G) \le 2C(G)$ and bound $C(G)$. Note that for any graph G with subgraphs $H_1,\ldots,H_n$ such that $G = \cup_{i=1}^n H_i$, $C(G) \le \sum_{i=1}^n C(H_i)$. Let T_{MST} be the minimum spanning tree of G computed by Algorithm MST in the first step of our algorithm and $F = G_2 - T_{\text{MST}}$ be the graph we added to T_{MST} in the for-loop of our algorithm. Then $C(G_2) \le C(T_{\text{MST}}) + C(F)$. To bound $C(F)$ in terms of $C(T_{\text{MST}})$, consider edge $(u,v) \in F$. It was added to create a path among the neighbors of some vertex, say, w. Thus (w,u) and (w,v) are in T_{MST}. We say (w,u) and (w,v) *pay for* (u,v). Notice each edge $(x,y) \in T_{\text{MST}}$ pays for at most four edges in F – two edges for which x is the common neighbor and two edges for which y is the common neighbor. These four edges correspond to edges adjacent to y and x on the two paths of neighbor vertices of x and y, respectively. By the weak triangle inequality, it follows that $C(F) \le 4\cdot 2^{c-1}C(T_{\text{MST}})$. Therefore,

$$\begin{aligned} P(G_2) &\le 2C(G_2) \\ &\le 2(4\cdot 2^{c-1}+1)C(T_{\text{MST}}) \\ &\le 2(4\cdot 2^{c-1}+1)C(T_{UPVCS}) \\ &\le 2(4\cdot 2^{c-1}+1)P(T_{UPVCS}) \\ &\le 2(4\cdot 2^{c-1}+1)P(G_{2-UPVCS}) \end{aligned}$$

where $G_{2-UPVCS}$ is a 2-UPVCS subgraph and the last inequality follows since $G_{2-UPVCS}$ is also a solution to the UPVCS problem.

Finally, we note that G_2 is indeed a spanning 2-vertex connected subgraph. Since T_{MST} spans G, clearly G_2 spans G. Furthermore, the removal of any single node leaves the graph connected because of the path amongst its neighbors. ■

It is slightly tricky to generalize this algorithm for $k \ge 3$. The main difficulty arises from the fact that the tree itself is just 1-connected. Thus the neighbor sets of vertices as defined by the tree can be too localized. In order to make the output graph k-connected, we must have an additional step in our algorithm that adds neighbors to guarantee a good intersection of neighbor sets throughout the graph. We would like to add these neighbors without incurring too much cost. We will bound the additional cost in a manner similar to the bound argument for $P(F)$, namely we will charge the additional cost to the edges of T_{MST}. However, we must be careful to charge each edge only a small number of times in order to get a good approximation factor. We can accomplish this by using the extended family of a vertex as its additional neighbors.

Algorithm **Distributed** k**-UPVCS**$(G(V,E))$
// compute the minimum spanning tree
$T_{\text{MST}} \leftarrow \text{Algorithm MST}(G(V,E))$
Root T_{MST} at an arbitrary vertex r
Perform a depth first search,
labeling each node the first and the last time it is visited
(the first node, i.e., r, has labels 0 and $2n-1$)
// add neighbors to each vertex
$G_k \leftarrow (V, \emptyset)$
for each node $v_i \in T_{\text{MST}}$
  Add edges (v_i, v_j) to G_k for $i+1 \le j \le i+2k$
  where indices are computed mod $2n$
  $N_i \leftarrow \{v_{i+1}, \ldots, v_{i+2k}\} \cup \{u : (v_i, u) \in T_{\text{MST}}\}$
end
// add a Harary graph among neighbors of vertices
for each node $v \in G_k$
  Consider an arbitrary cyclic ordering $\sigma(N_i)$
  of vertices in N_i
  for each $u \in N_i$
    Add to G_k edges (u,v) for each of
    the $\lceil \frac{k}{2} \rceil$ vertices before and after u in $\sigma(N_i)$
  end
end

Fig. 5. A formal description of Algorithm k-UCVCS

Specifically, we create a cyclic ordering of the vertices as follows: Root T_{MST} at an arbitrary node r and let r be the first node in the ordering, i.e., $r = v_0$. Perform a depth first search of T_{MST} from vertex r, adding each element v to the ordering both when the search enters the subtree routed v and when it leaves the subtree routed at v. Hence, each element appears in the ordering twice, and $r = v_0 = v_{2n-1}$. Suppose the i'th vertex v_i in this ordering has just $j < k$ neighbors. Then we augment the neighborhood by adding edges from v_i to the $2k-j$ vertices that follow v_i in the ordering, i.e., $v_{i+1}, \ldots, v_{i+2k-j}$ (so as long as k is constant, this step is locally distributed). Now all vertices have at least k neighbors. For each vertex v, we add the following k-connected graph (a *Harary graph*[2]) to its neighbors N: form an arbitrary cycle C amongst the vertices in N; connect each vertex $u \in C$ to the first $\lceil \frac{k}{2} \rceil$ vertices on each side of u. Repeating this procedure for every vertex will make the entire graph k-connected. See Figure 5 for a formal description.

We now consider the message complexity of the algorithm. The algorithm is based on a distributed minimum spanning tree algorithm which can be computed with $O(n \log n + m)$ messages. After the computation of the minimum spanning tree, the remainder of the algorithm is locally distributed. Even the neighbor addition step must query at most $O(k)$ neighbors which are at most a distance of $O(k)$ from the original vertex. Therefore, these remaining steps use just $O(k^2 n)$ messages, and the total message complexity of the algorithm is $O(n \log n + m + k^2 n)$.

Theorem 6: For any geometric graph G, Algorithm Distributed k-UPVCS is a distributed algorithm which outputs a k-vertex connected subgraph whose power is an $O(k^{2c+2})$-approximation of the power of a k-UPVCS subgraph.

Before starting the proof of Theorem 6, we state the following combinatorial lemma concerning k-connectivity.

Lemma 8: Let G be a connected graph whose minimum degree is at least k. Let G' be the graph obtained from G by adding some (possibly empty) set of edges to G such that for each vertex v, the graph $G'[N(v)]$ induced by the neighbors $N(v)$ of v is k-connected. Then the new graph G' is k-connected.

Proof: The proof is by contradiction. Suppose there is a set S of size at most $k-1$ whose deletion disconnects G'. For each pair (u,v) of the remaining vertices, define P_{uv} to be a path in G' from u to v with minimum number of vertices from S and let l_{uv} be this minimum number. We claim that l_{uv} should be zero for all pairs of the remaining vertices, and this finishes the proof of the lemma. Suppose it is not the case. Among all pairs of the remaining vertices, we consider a pair (u,v) for which the l_{uv} value is minimized and non-zero. Let u' be the nearest node to u in P_{uv} whose next vertex s is in S. Let $v' \neq u'$ be the other neighbor of s in P_{uv}. Since s has degree k and $G'[N(s)]$ is k-connected, removing $S - \{v'\}$ from $G'[N(s)]$ does not make $G'[N(s)]$ disconnected. Thus there should be a path P' of vertices not in S which connects u' to v'. Now, if we connect u' and v' in P_{uv} via path P', instead of vertex s, we obtain a new path which decreases l_{uv} by at least one, a contradiction. ■

We are now ready to prove Theorem 6.

Proof: [of Theorem 6] First, we prove that the graph G_k which is the output of Algorithm Distributed k-UPVCS is k-connected. Since each vertex gets two labels in the algorithm, we can observe that graph G_k after adding the set of edges in the first for-loop of the algorithm has minimum degree at least k. In the second for-loop, we construct a k-connected Harary graph on the neighborhood $N(v)$ of each vertex $v \in G$. The k-connectivity of G_k follows immediately from Lemma 8.

Finally we show that the approximation factor of the algorithm is $O(k^{2c+2})$. The proof is very similar to the proof of Theorem 5. Again, we use the fact that $P(G_k) \le 2C(G_k)$ and bound $C(G_k)$. Let N be the set of edges added in the first for-loop to create at least k neighbors for each vertex and O be the set of edges added in the second for-loop to create cycles amongst neighbors. Thus, $G_k = T_{\text{MST}} \cup N \cup O$, and so $C(G_k) \le C(T_{\text{MST}}) + C(N) + C(O)$. We bound $C(N)$ in terms of $C(T_{\text{MST}})$ by charging edges in T_{MST} for edges in N. We claim each edge (u,v) can be charged at most $O(k^2)$ times — $O(k)$ times for each of the $O(k)$ vertices which are at distance at most $2k$ in the cyclic ordering of $2n$ labels of n vertices of the graph. Note that each added edge spans at most $O(k)$ original edges, and so by the weak triangle inequality, this implies $C(N) \le O(k^2 \cdot k^{c-1}) C(T_{\text{MST}})$. Now we bound $C(O)$ in terms of $C(N \cup T_{\text{MST}})$. We again observe that after adding a Harary graph among neighbors of each vertex, each edge in $N \cup T_{\text{MST}}$ can be charged for at most $O(k^2)$ edges in O ($O(k)$ times for each of the k vertices which are at distance at most $k/2$ in the cyclic ordering of neighbors $N(v)$ of a vertex v), and each added edge spans at most $O(k)$ edges from $N \cup T_{\text{MST}}$. Therefore, by the weak triangle inequality, $C(O) \le O(k^2 \cdot k^{c-1})(C(N) + C(T_{\text{MST}}))$, and so

$$
\begin{aligned}
P(G_k) &\le 2C(G_k) \\
&\le 2(C(T_{\text{MST}}) + C(N) + C(O)) \\
&\le O(k^2 \cdot k^{c-1})(C(T_{\text{MST}}) + C(N)) \\
&\le O(k^{c+1}) \cdot O(k^2 \cdot k^{c-1}) C(T_{\text{MST}}) \\
&\le O(k^{2c+2}) P(G_{OPT})
\end{aligned}
$$

where G_{OPT} is a k-UPVCS subgraph and the last inequality follows from a reasoning similar to that in the proof of Theorem 5. ■

[2] In fact, Harary graphs are defined differently when k, the number of nodes, is odd. However, the slightly altered definition provided here enables us to prove a better bound on the power consumption of the resulting graph.

In the Algorithm Distributed k-UPVCS, we do not try to minimize the constants. In Appendix A, we describe a slight modification of Algorithm Distributed k-UPVCS for the special case $k = 3$, which might be used in practice. In this case, we obtain the constants in the approximation factor explicitly.

We note that $O(k^{2c+2})$ is not necessarily the best approximation factor one can prove for Algorithm Distributed k-UPVCS (mainly because we compare our solutions with optimal 1-connected subgraph (MST) and not optimal k-connected subgraphs). In fact our practical results in Section V show that we often perform much better than CBTC algorithm and the performance is comparable to the centralized algorithm. In addition, this algorithm is both distributed and highly localized in the sense that after the distributed computation of the spanning tree and selection of the root, all operations can be performed locally. For this reason, we believe this algorithm is very suitable for practical situations.

We emphasize that after computing the MST, the remaining steps of the algorithm are based on local information and can be implemented locally (as long as k is a constant). To the best of our knowledge there is no locally computable algorithm or approximation algorithm for MST. However, if we are willing to forgo the approximation guarantee, we can make our algorithm completely local by using a local heuristic for MST like CBTC as the initial 1-connected graph in our algorithm.

Finally, we note that since we compare the solution to MST and a k-vertex connected graph is also k-edge connected, this distributed algorithm gives the same approximation guarantee for the power optimum k-edge connected subgraph problem (k-UPECS).

V. Performance Evaluation

In the previous section, we proved a theoretical bound on the performance of our algorithms. In this section, we observe that our algorithms even perform well in practice. In order to understand the effectiveness of our algorithms, we compare them to a previous heuristic, namely the Cone-Based Topology Control heuristic of Wattenhofer et al. [38], Li et al. [26] and Bahramgiri et al. [2].

A. Experimental Environment

We generate random networks, each with 100 nodes. The maximum possible power at each node is fixed at $E_{\max} = (250)^2$. With our assumed *power attenuation exponent* $c = 2$, this implies a maximum communication radius R of 250 meters. We evaluate the performance of our algorithms on networks of varying density. Note that we expect, and in fact observe, that the performance of all algorithms improves as density, and thus the number of extraneous edges, increases. In order to obtain a given density (from 6 nodes per transmission area to 30), we position 100 nodes randomly in an appropriately sized square. We assume the MAC layer is ideal. These networks are similar to the sample networks used by Wattenhofer et al. [38] and Cartigny et al. [8].

As a performance measure, we compute the average expended energy ratio (EER) of each algorithm for these random networks:

$$EER = \frac{\text{Average Power}}{E_{\max}} \times 100.$$

This measure compares the average power of a node in the network to the maximum power of a node in the network; we would like this ratio to be small.

B. Observations

The three algorithms we consider in this experiment are the Cone-Based Topology Control [2] heuristic recapped in Section III-B, the Distributed k-UPVCS algorithm introduced in Section IV-B, and the Global k-UPVCS algorithm introduced in Section IV-A. Figure V-B, Table V-A, and Table V-A depict all these results.

Here, we discuss the results for 2-UPVCS and 3-UPVCS. For 2-UPVCS, the average power assigned by Global k-UPVCS is from 4% to 15% of the maximum possible power, $E_{\max}$ (i.e., the EER is between 4 and 15). The average power for Distributed k-UPVCS is from 7% to 32% of $E_{\max}$ whereas for Cone-Based Topology Control, it is from 58% to 90%. For 3-UPVCS, the average power assigned is from 5% to 20% for Global k-UPVCS, from 9% to 39% for Distributed k-UPVCS, and from 75% to 100% for Cone-Based Topology Control. These numbers show that Global k-UPVCS and Distributed k-UPVCS consistently outperform Cone-Based Topology Control in regards to average power.

As we expect, Global k-UPVCS outperforms Distributed k-UPVCS in most instances. It is not surprising to see that the best algorithm is the totally globalized one, i.e., we can make better choices by ignoring the communication complexity. However, Distributed k-UPVCS is still very competitive with Global k-UPVCS. In fact, while the performance of Global k-UPVCS ranges from 4% of $E_{\max}$ for dense networks to 20% of $E_{\max}$ for sparse networks, the performance of Distributed k-UPVCS ranges from 7% for dense networks to 35% for sparse networks. Hence, Global k-UPVCS spends at most 75% less than Distributed k-UPVCS. In contrast, Distributed k-UPVCS never uses more than twice the power of Global k-UPVCS. Note that the input networks are geometric, thus the theoretical performance guarantee of Distributed k-UPVCS proved in Section IV-B holds.

Global k-UPVCS and Distributed k-UPVCS both outperform Cone-Based Topology Control in all cases. However, the improvement of our algorithms is most obvious in sparse networks. For sparse graphs and especially for 3-UPVCS, the Cone-Based Topology Control average power usage is very close to the maximum power which shows the main flaw of this heuristic and the advantage of our algorithms. The difference between Global k-UPVCS and Distributed k-UPVCS decreases as density increases which implies that Distributed k-UPVCS is more competitive to Global k-UPVCS in dense graphs.

Finally, it is worth mentioning that although our distributed algorithms in this paper have much better performance than the CBTC algorithm, CBTC is locally computable even for dynamic settings (not just the static ones that we considered in this paper) such local approaches are more desirable. Our algorithm which seems more desirable has some maintenance overhead due to the minimum spanning tree computation which needs to be considered further in dynamic settings. This computation is distributed, not local, and so our algorithm can not be implemented locally in its entirety. However, we suspect that for the k-UPVCS problem, locally computable algorithms can not guarantee constant factor approximations.

VI. Conclusion

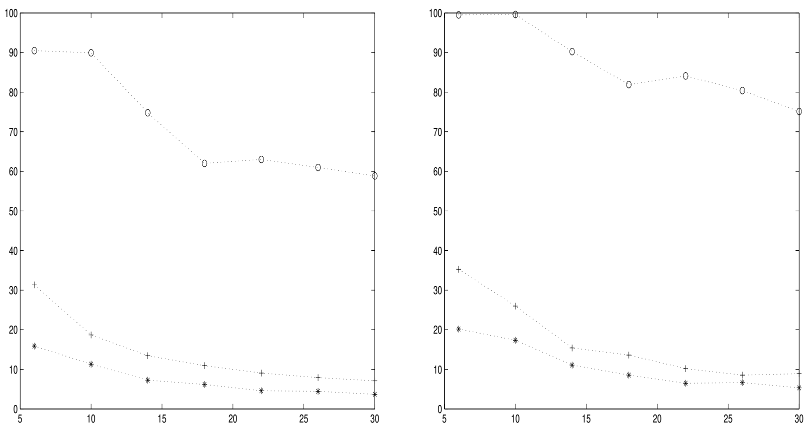
In this paper, we considered power minimization for k-fault tolerant topology control in ad hoc wireless networks. We mentioned the complexity issues of this problem and showed that previous heuristics and approaches do not give us good approximation factors. We demonstrated two approximation algorithms which give us $O(k)$- and $k^{O(c)}$-approximation factors, the sec-

Density	Degree	Cone-Based Topology Control: ERR	Distributed k-UPVCS: ERR	Global k-UPVCS: ERR
6	15.56	90.4726	31.3103	15.8636
10	21.62	89.9237	18.6790	11.2938
14	34.02	74.7904	13.4375	7.2419
18	38.72	62.0195	10.9241	6.1628
22	45.24	63.0056	9.0454	4.5905
26	51.26	60.9590	7.8912	4.4476
30	54.56	58.8282	7.0988	3.6705

TABLE II
EXPENDED ENERGY RATIO $c = 2$ FOR 2-UPVCS (k=2)

Density	Degree	Cone-Based Topology Control: ERR	Distributed k-UPVCS: ERR	Global k-UPVCS: ERR
6	15.56	99.5252	35.2772	20.1612
10	21.62	99.6080	25.9680	17.3236
14	34.02	90.2409	15.4045	11.0623
18	38.72	81.9197	13.5849	8.5273
22	45.24	84.0958	10.1658	6.4635
26	51.26	80.3984	8.5393	6.6278
30	54.56	75.1298	8.3860	5.3084

TABLE III
EXPENDED ENERGY RATIO $c = 2$ FOR 3-UPVCS(k=3)



(a)2-UPVCS (b)3-UPVCS

Fig. 6. Cone-Based Topology Control (o) (*low performance*), Distributed k-UPVCS (+) (*middle performance*), and Global k-UPVCS (*) (*high performance*). These graphs depict EER (Expended Energy Ratio) versus density.

ond of which can be easily implemented in a distributed ad hoc wireless network.

Compared to previous methods, our distributed algorithm is not as practical as CBTC and it is more suitable for static ad-hoc networks. However, it gives us a framework to increase the connectivity of the network using the local information. Furthermore, if we use a good 1-connected subgraph like MST, the practical results and worst-case theoretical comparison show that the performance of this algorithm is much better than that of CBTC.

Obtaining an approximation algorithm with factor better than $8(k-1)$, especially with a factor $\alpha = o(k)$, for undirected minimum power k-vertex connected subgraph (k-UPVCS) is an interesting open question. As we showed, the solution to undirected minimum cost k-vertex connected subgraph (k-UCVCS) can not give an $o(k)$-approximation factor for k-UPVCS. Also, a natural generalization of the $(\log k)$-approximation algorithm for k-UCVCS can not give us better than an $\Omega(\frac{n}{k})$-approximation algorithm. Other interesting open questions include obtaining constant factor approximation algorithms for geometric k-UPVCS and k-UPECS. We give $O(k)$-approximation algorithms for these problems; however we suspect that there are constant factor approximation algorithms for these problems, especially since there are constant factor approximation algorithms for the minimum normal cost variants of these problems. For the directed versions of these problems, to the best of our knowledge, almost nothing is known and any progress in this regard would be interesting. In fact, we believe for geometric graphs, along with the 12-approximation of Wan et al. [36] for the broadcast problem, our Distributed k-UPVCS algorithm from Section IV-B can be generalized for the directed version.

The minimum range assignment problem when the stations are located along a line at arbitrary distance apart have been subject to several recent studies [3], [11], [22], [32]. Kirousis et al. [22] showed an $O(n^4)$ time dynamic programming algorithm to find a minimum cost range assignment of collinear points ensuring that the resulting directed network is strongly connected. We strongly believe that using the same approach, the UPVCS of collinear points can be solved in polynomial time. It would be interesting to know whether or not the result can be generalized to k-UPVCS of collinear points for $k > 1$.

As mentioned before, so far all approximation (not heuristic) algorithms for the range assignment problem are based on minimum spanning trees or approximations of minimal spanning trees, which are globalized. Our approximation for k-UPVCS uses the minimum (or any approximation for minimum) spanning tree as a black box, and the rest of the operations are very simple local ones. Thus using our approach, any localized algorithm for minimum spanning trees in ad hoc wireless networks can result in localized approximation algorithm for k-UPVCS. This leads to another interesting open question — to develop a localized minimum spanning tree algorithm or to prove such an algorithm does not exist.

Finally, in *broadcast oriented protocols*, we have the same objectives of topology control oriented protocols, mentioned in this paper, but we consider the broadcast process from a given source node and we want to have k-disjoint paths from the source to some or all other nodes. Obtaining approximation algorithms for this setting is another possible extension of our results (Notice that for the case of $k = 1$, there exists such an algorithm using a reduction to minimum directed Steiner tree [27].)

VII. Acknowledgement

We would like to thank Joseph Cheriyan, Erik D. Demaine, Michel X. Goemans, Nancy A. Lynch, and Mohammad Mahdian for helpful discussions. We would also like to thank Jennifer Welch for helpful comments on a preliminary version of the paper.

References

[1] E. Althaus, G. Calinescu, I. Mandoiu, S. Prasad, N. Tchervenski, and A. Zelikovsky. Power efficient range assignment in ad-hoc wireless networks. *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, 2003. To appear.

[2] M Bahramgiri, M Hajiaghayi, and V.S. Mirrokni. Fault-tolerant and 3-dimensional distributed topology control algorithms wireless multi-hop networks. *Wireless Networks*, To appear. A preliminary version in *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 392–398. 2002.

[3] M.A. Bassiouni and C. Fang. Dynamic channel allocation for linear macrocellular topology. *Proceedings of the 13th ACM Symposium on Applied Computing (SAC)*, pages 382–388, 1998.

[4] D.M. Blough, M. Leoncini, G. Resta, and P. Santi. On the symmetric range assignment problem in wireless ad hoc networks. *Proceedings of the 2nd IFIP International Conference on Theoretical Computre Science (TCS)*, pages 71–82, 2002.

[5] D.M. Blough, M. Leoncini, G. Resta, and P. Santi. The K-neigh protocol for symmetric topology control in ad hoc networks. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. June 2003.

[6] J.L. Bredin, E.D. Demaine, M.T. Hajiaghayi, and D. Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 309–319, 2005.

[7] G Calinescu, I.L Mandoiu, and A. Zelikovsky. Symmetric connectivity with minimum power consumption in radio networks. In *Proceedings of 17th IFIP World Computer Congress*, pages 119–130. 2002.

[8] Julien Cartigny, David Simplot, and Ivan Stojmenovic. Localized minimum-energy broadcasting in ad-hoc networks. In *Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*. 2003.

[9] J. Cheriyan, S. Vempala, and A. Vetta. Approximation algorithms for minimum-cost k-vertex connected subgraphs. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC),*, pages 306–312. 2002.

[10] A. Clementi, P Crescenzi, P. Penna, G. Rossi, and P. Vocca. A worst-case analysis of an mst based herustic on construct energy-efficient broadcast trees in wireless networks. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 121–131. 2001.

[11] A. Clementi, A. Ferreira, P. Penna, S. Perennes, and R. Silvestri. The minimum range assignment problem on linear radio networks. *Proceedings of 8th Annual European Symposium on Algorithms (ESA)*, pages 143–154, 2000.

[12] A.E.F. Clementi, G. Huiban, P. Penna, G. Rossi, and Y.C. Verhoeven. Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks. *Proceedings of 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE)*, pages 23–38, 2002.

[13] A.E.F. Clementi, P. Penna, and R. Silvestri. Hardness results for the power range assignment problem in packet radio networks. *Proceedings of the 2nd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 197–208, 1999.

[14] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discrete Math.*, 5(1):25–53, 1992.

[15] András Frank and Éva Tardos. An application of submodular flows. *Linear Algebra Appl.*, 114/115:329–348, 1989.

[16] R. Gallager, P. Humbler, and P. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Programming Language Systems*, 5:66–77, 1983.

[17] M.T. Hajiaghyi, G. Kortsarz, V.S. Mirrokni, and Z. Nutov. Power Optimization for Connectivity Problems. In *Proceedings of the 11th Interntional Integer Programming and Combinatorial Optimization Conference (IPCO)*, pages 349-361. 2005.

[18] G. Kortsarz, V.S. Mirrokni, and Z. Nutov, E. Tsanko. Power Optimization for Network Design Problems, submitted.

[19] K. Jain. A factor 2 approximation for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[20] X. Jia, D. Kim, S. Makki , P. Wan, and C. Yi. Power Assignment for k-Connectivity in Wireless Ad Hoc Networks. *Journal of Combinatorial Optimization*, 9(2):213–222, 2005.

[21] Samir Khuller and Balaji Raghavachari. Improved approximation algorithms for uniform connectivity problems. *J. Algorithms*, 21(2):434–450, 1996.

[22] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Power consumption in packet radio networks. *Theoret. Comput. Sci.*, 243(1-2):289–305, 2000.

[23] G. Kortsarz and Z. Nutov. Approximating node connectivity problems via set covers. In *Proceedings of the third International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 194–205. 2000.

[24] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. In *Proceedings of the 24th ACM Symposium on the Theory of Computing (STOC)*, pages 759–770. 1992.

[25] N. Li and J.C. Hou. FLSS: a fault-tolerant topology control algorithm for wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MOBICOM)*, pages 275–286. 2004.

[26] L. Li, J. Halpern, V. Bahl, Y.M. Wang, and R. Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *Proceedings of ACM Symposium on Principle of Distributed Computing (PODC)*, pages 264–273. 2001.

[27] W. Liang. Constructing minimum-energy broadcast trees in wireless ad hoc networks. *Proceedings of the third ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2002. To appear.

[28] E. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S. Ravi. Algorithmic aspects of topology control problems for ad hoc networks. *Proceedings of the third ACM international symposium on Mobile ad hoc networking and computing*, 2002.

[29] X. Li, W. Song, and Y. Wang. Efficient topology control for wireless ad hoc networks with non-uniform transmission ranges. *Wireless Networks*, To appear.

[30] X. Li, Y. Wang, and W. Song. Applications of k-Local MST for Topology Control and Broadcasting in Wireless Ad Hoc Networks. *IEEE Transactions on Parallel and Distributed Systems*. 15(12):1057–1069, 2004.

[31] X. Li, Y. Wang, P. Wan, and C. Yi. Robust deployment and fault tolerant topology control for wireless ad hoc networks. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. June 2003.

[32] R. Mathar and J Mattfeldt. Optimal transmission ranges for mobile communication in linear multihop packet radio networks. *Wireless Networks*, 2:329–342, 1996.

[33] R. Ramanathan and R. Rosales-Hain. Topology control of multihop radio networks using transmit power adjustment. In *Proceedings of ninetieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 404–413. March 2000.

[34] Volkan Rodoplu and Teresa H. Meng. Minimum energy mobile wireless

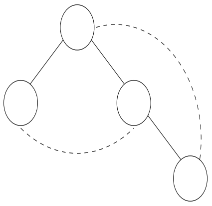


Fig. 7. Adding neighbors to vertices in T_{MST} (the intermediary graph before adding cycles among neighbors)

networks. *IEEE J. Selected Areas in Communications*, 17(8):1633–1639, 1999.

[35] Mader W. Ecken vom grad n in minimalen n-fach zusammenhangenden Graphen. *Arch. Math. (Basel)*, 23:219–224, 1972.

[36] P. Wan, A. Calinescu, X. Li, and O. Frieder. Minimum energy broadcast routing in static ad hoc wireless networks. In *Proceedings of twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 607–617. 2001.

[37] Y. Wang and X.Y. Li. Minimum Power Assignment in Wireless Ad Hoc Networks with Spanner Property. In *Proceedings of IEEE Workshop on High Performance Switching and Routing*, 2004.

[38] R. Wattenhofer, L. Li, V. Bahl, and Y.M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proceedings of twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1388–1397. 2001.

[39] H. Zhang and J. Hou. On the Critical Total Power for Asymptotic k-Connectivity in Wireless Networks. In *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*. 2005. To appear.

APPENDIX

I. DISTRIBUTED APPROXIMATION FOR 3-CONNECTIVITY

In this section, we describe a distributed approximation algorithm for 3-connectivity, in which we obtain the approximation factor explicitly. Similar to Section IV-B, here we assume that our graph is geometric (i.e., the edge lengths satisfy the triangle inequality) and the power attenuation exponent is uniform.

In the case of 3-connectivity, we must add one neighbor to each node. We will find this additional neighbor amongst the siblings (or grandparent if there are no siblings). This process is illustrated in Figure 7. Figure 8 contains a formal description of this algorithm. This algorithm is based on a distributed minimum spanning tree algorithm which can be computed with $O(n \log n + m)$ messages. After the computation of the minimum spanning tree, the remainder of the algorithm is locally distributed. Even the neighbor addition step must query at most one neighbor which is at most a distance of two from the original vertex. Therefore, these remaining steps use just $O(n)$ messages, and the total message complexity of the algorithm is again $O(n \log n + m)$.

Theorem 7: For any geometric graph G, Algorithm Distributed 3-UPVCS returns a 3-vertex connected subgraph G_3 whose power $P(G_3)$ is at most $2(1 + 7 \cdot 2^{c-1} + 12 \cdot 4^{c-1})$ times the power of a 3-UPVCS subgraph.

Proof: The proof is very similar to the proof of Theorem 5. Again, we use the fact that $P(G_3) \leq 2C(G_3)$ and bound $C(G_3)$. Let N be the set of edges added in the first for-loop to create neighbors and O be the set of edges added in the second for-loop to create cycles amongst neighbors. Thus, $G_3 = T_{\mathrm{MST}} \cup N \cup O$, and so $C(G_3) \leq C(T_{\mathrm{MST}}) + C(N) + C(O)$. We bound $C(N)$ in terms of $C(T_{\mathrm{MST}})$ by charging edges in T_{MST} for edges in N. We claim each edge (u, v) can be charged at most 3 times — twice for edges added amongst siblings and once for an edge added from the child of u to its grandparent v. Note that each added edge spans at most two original edges, and so by the weak triangle inequality, this

Algorithm **Distributed** 3-**UPVCS**($G(V, E)$)
// compute the minimum spanning tree
$T_{\mathrm{MST}} \leftarrow \mathrm{Algorithm\ MST}(G(V, E))$
root T_{MST} at arbitrary vertex r
label nodes $v_1, \ldots, v_n \in V$ in an arbitrary order
// add a neighbor to each vertex
$G'_3 \leftarrow T_{\mathrm{MST}}$
for node $u \in T_{\mathrm{MST}} - \{r\}$
  if u has siblings **then**
    add edge (u, v) to G_2 where v is
    successor of u in cyclic ordering induced
    by vertex labelling restricted to sibling set
  else
    add edge (u, v) to G_2 where v is
    grandparent of u
end
// add a cycle among neighbors of vertices
for node $u \in G'_3$
  $N \leftarrow \{v | (u, v) \in G'_3\}$
  label vertices in N in an arbitrary order
  $E \leftarrow E \cup \{(v_1, v_2), \ldots, (v_{|N|-1}, v_{|N|}), (v_{|N|}, v_1)\}$
end

Fig. 8. A formal description of Algorithm k-UCVCS for $k = 3$

implies $C(N) \leq 3 \cdot 2^{c-1} C(T_{\mathrm{MST}})$. Now we bound $C(O)$ in terms of $C(N \cup T_{\mathrm{MST}})$. As argued in the proof of Theorem 5, each edge in $N \cup T_{\mathrm{MST}}$ can be charged for at most four edges in O, and each added edge spans at most two edges from $N \cup T_{\mathrm{MST}}$. Therefore, by the weak triangle inequality, $C(O) \leq 4 \cdot 2^{c-1}(C(N) + C(T_{\mathrm{MST}}))$, and so

$$\begin{aligned} P(G_3) &\leq 2C(G_3) \\ &\leq 2(C(T_{\mathrm{MST}}) + C(N) + C(O)) \\ &\leq 2(1 + 4 \cdot 2^{c-1})(C(T_{\mathrm{MST}}) + C(N)) \\ &\leq 2(1 + 4 \cdot 2^{c-1})(1 + 3 \cdot 2^{c-1}) C(T_{\mathrm{MST}}) \\ &\leq 2(1 + 7 \cdot 2^{c-1} + 12 \cdot 4^{c-1}) P(G_{OPT}) \end{aligned}$$

where G_{OPT} is a 3-UPVCS subgraph and the last inequality follows from a reasoning similar to that in the proof of Theorem 5.

Finally, we note that G_3 is indeed a spanning 3-vertex connected subgraph. Since T_{MST} spans G, clearly G_3 spans G. Furthermore, the removal of any two nodes leaves the graph connected. More precisely, we can consider two cases. In the first case, we remove two non-adjacent vertices u and v in T_{MST}. Here because of the cycles amongst the neighbors and the path from u to v in T_{MST}, the graph remains connected. In the second case, we remove two adjacent vertices u and v in T_{MST} (thus without loss of generality, we can assume u is the parent of v in T_{MST}.) Again in this case, because of adding a sibling or grandparent of each vertex to the set of its neighbors and then adding the cycle amongst its neighbors, we have connectivity of the remaining graph. ∎