CLTC: A Cluster-Based Topology Control Framework for Ad Hoc Networks

Chien-Chung Shen, *Member*, *IEEE*, Chavalit Srisathapornphat, *Student Member*, *IEEE*, Rui Liu, Zhuochuan Huang, *Student Member*, *IEEE*, Chaiporn Jaikaeo, *Student Member*, *IEEE*, and Errol L. Lloyd, *Member*, *IEEE Computer Society*

Abstract—The topology of an ad hoc network has a significant impact on its performance in that a dense topology may induce high interference and low capacity, while a sparse topology is vulnerable to link failure and network partitioning. *Topology control* aims to maintain a topology that optimizes network performance while minimizing energy consumption. Existing topology control algorithms utilize either a purely centralized or a purely distributed approach. A centralized approach, although able to achieve strong connectivity (*k*-connectivity for $k \geq 2$), suffers from *scalability problems*. In contrast, a distributed approach, although scalable, lacks strong connectivity guarantees. We propose a hybrid topology control framework, *Cluster-based Topology Control* (CLTC), that achieves both scalability and strong connectivity. By varying the algorithms utilized in each of the three phases of the framework, a variety of optimization objectives and topological properties can be achieved. In this paper, we present the CLTC framework; describe topology control algorithms based on CLTC and prove that *k*-connectivity is achieved using those algorithms; analyze the message complexity of an implementation of CLTC, namely, CLTC-A, and present simulation studies that evaluate the effectiveness of CLTC-A for a range of networks.

Index Terms—Topology control, clustering, ad hoc networks, transmission power assignment, strong connectivity.

1 Introduction

In ad hoc networks, where nodes are deployed without any preconfigured infrastructure and communicate via multihop wireless links, the *network topology* is autonomously formed based on the nodes' locations and communication ranges. The network topology has a huge impact on the performance of the network. A dense topology may induce high interference, which, in turn, reduces the effective network capacity due to limited spatial reuse and causes unnecessarily high energy consumption. In contrast, a sparse topology is vulnerable to network partitioning due to node or link failures. **Topology control** for ad hoc networks aims to maintain a specified topology by controlling which links should be included in the network to achieve a set of network-wide or session-specific objectives such as reducing interference or probability of detection, reducing energy consumption, increasing the effective network capacity, and reducing end-to-end delay. The primary method of accomplishing topology control is by adjusting the transmission powers of the network nodes.[1]

Several topology control algorithms based on transmission power adjustment have been proposed, where topology control is defined as the problem of assigning transmission powers to the nodes so that the resulting topology achieves certain connectivity properties and so that some function of the transmission powers is optimized. Centralized algorithms [2], [3], [4] rely on the assumption that the locations of all of the nodes are known by a central entity in order to calculate the transmission powers that result in a topology with strong connectivity.[2] However, these algorithms are not scalable for large ad hoc networks where excessive amounts of information would need to be collected by a central entity. Distributed algorithms [4], [5], [6], on the other hand, are generally scalable and adaptive to mobility due to the fact that each node relies on local information collected from nearby nodes to autonomously compute its appropriate transmission power. Considering that the information each node obtains is limited, the major drawback of the distributed approach is that strong connectivity is neither easily achieved nor guaranteed.

As noted above, topology control algorithms achieve a required topology while optimizing some objective functions of the network. In this paper, we study specific instances of the topology control problem by considering the following two optimization functions:

- minimize the maximum power used by any node in the network. We refer to this criterion as MINMAX.
- minimize the total power used by all of the nodes in the network. This is equivalent to minimizing the average power used by the nodes. We refer to this criterion as MINTOTAL.

The main result of this paper is a hybrid topology control framework, termed *Cluster-based Topology Control* (CLTC for short) to facilitate topology control for large scale ad hoc

1. An alternative is to adjust the antenna patterns of the network nodes [1], but that relies on the availability of directional antennas. This alternative is not considered here.

2. In this paper, the term *strong* connectivity is used to denote k-connectivity for $k \geq 2$.

• *The authors are with the Computer and Information Sciences University of Delaware, Newark, DE 19716.*
E-mail: {cshen, srisatha, ruliu, zhhuang, jaikaeo, elloyd}@cis.udel.edu.

Manuscript received 27 Feb. 2003; revised 27 May 2003; accepted 13 Aug. 2003. For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number 0025-0203.

Published by the IEEE CS, CASS, ComSoc, IES, & SPS

networks. Briefly, the CLTC framework consists of three phases:

1. Nodes autonomously form clusters and elect cluster heads.
2. Using the information gathered from the members of its cluster, each cluster head assigns transmission powers to the nodes in its cluster such that strong connectivity is guaranteed within each cluster.
3. Each cluster head selects a set of border nodes, shares information with neighboring cluster heads, and computes appropriate transmission powers for border nodes in order to establish strong connectivity with the neighboring clusters.

In later sections, we will provide the details of this CLTC framework and prove its correctness by showing that, as long as there is strong connectivity within each cluster and between adjacent clusters, then strong connectivity is guaranteed for the entire network.

Note that CLTC utilizes a hybrid approach to topology control. It uses a centralized algorithm within a cluster and between adjacent clusters to achieve strong connectivity and yet achieves the scalability and adaptability of a distributed approach with localized information exchange between adjacent clusters. Further, the CLTC framework allows different topology control algorithms to be applied within clusters and between adjacent clusters, depending on the optimization objective (e.g., MINMAX and MINTOTAL).

The remainder of the paper is organized as follows: Section 2 reviews some existing topology control methods. Section 3 presents the network and communication models and provides some definitions. Section 4 provides the details of the CLTC framework. The correctness of the framework is proven in Section 5. Section 6 describes particular topology control algorithms that can be utilized in the CLTC framework. Section 7 analyzes CLTC in terms of message complexity. Simulation results relative to CLTC and the specific methods of Section 6 are presented in Section 8 and Section 9 concludes the paper with a discussion of future research directions.

2 Existing Topology Control Techniques

This section reviews prior work on topology control using transmission power adjustment, beginning with centralized algorithms and then considering distributed approaches.

2.1 Centralized Algorithms

The initial work on algorithms for topology control using power adjustment appeared in [4]. That paper describes two centralized algorithms where the induced topology is 1-connected or 2-connected, respectively, and where the optimization criterion is MINMAX (i.e., minimize the maximum power utilized by any node). This work is generalized in [3] to show that, if the desired topology property is *monotone*, then, when the optimization goal is MINMAX, the power assignments can be computed in polynomial time. Here, a property is monotone if the property still holds even when a node increases its transmission power. Connectedness properties are monotone, while properties like "*the induced graph is a tree*" are not. A general framework is also presented in [3] for approximation algorithms when the objective is MINTOTAL and the property is monotone. Note that minimizing the total power is NP-hard even for the property of 1-connected [7].

2.2 Distributed Algorithms

In practice, especially in large and dynamic networks, centralized algorithms are not suitable due to the lack of responsiveness and the excessive amounts of information that need to be collected by a central node. Thus, [4] described two *distributed* algorithms, termed *Local Information No Topology* (LINT) and *Local Information Link-State Topology* (LILT). Both of these protocols rely on partial topology information, which could be collected by either a routing protocol or any neighbor discovery protocol available, and adjust transmission powers to maintain the desired number of neighbors of each node. In particular, LINT relies on the neighbor information that is obtained by most routing protocols, while LILT exploits some amount of global topology information collected when the network is operated with a link-state routing protocol. Although both protocols guarantee that the resulting network is 1-connected, neither is able to guarantee strong connectivity.

In [8], a position-based distributed algorithm is proposed to achieve minimum energy paths from all nodes to a master site node. The algorithm relies on the peers' position information and a simple transmission power roll-off model. As with [4], 1-connectivity is guaranteed, but nothing more.

A distributed algorithm is presented in [9] that aims at achieving a particular routing table under the MINMAX criterion. That algorithm results in all nodes having the same transmit power. The algorithm cooperates with proactive routing mechanisms to maintain multiple routing tables, one for each transmit power level. To select the optimum power level, the algorithm considers the smallest power level which results in a routing table with the same number of entries as the routing table obtained at the full transmit power.

Finally, in [5], [6], a distributed topology control mechanism, termed *Cone-based Topology Control*, is proposed where each node autonomously controls its transmission power so that at least one neighbor is found in any surrounding cone of a certain size. They show that $5\pi/6$ is the upper-bound cone size to achieve 1-connectivity. Again, strong connectivity may or may not result.

3 Models and Definitions

This section provides fundamental concepts used throughout the paper. We begin with a description of the wireless network model upon which the results presented here are based and follow with several definitions that formalize connectivity issues in graph theoretic terms.

3.1 Wireless Network and Communication Models

The model of wireless networks utilized in this paper is adapted from [4]. There, $M = (N, L)$ is a multihop wireless network where $N = \{m_1, \ldots, m_n\}$ is a set of nodes and L is a one-to-one mapping from N to planar coordinates. Each node is able to obtain its coordinates by some means (e.g., GPS). Further, each node u is able to adjust its transmission power level p_u within the range $0 \leq p_u \leq P_{max}$, where P_{max} is the

maximum transmission power (we assume this value is the same for all nodes). For a node u to successfully communicate with another node v, the signal received at v must exceed the *receive sensitivity* S. Recall that signals lose their power as a function of distance when propagating through a communication medium. Here, the path loss propagation function is (in dB) $\gamma(l_u, l_v)$, where the signal travels from u to v and l_u, l_v are the coordinates of nodes u and v, respectively. In order to guarantee a successful reception at v, it must be that $p_u \geq S + \gamma(l_u, l_v)$. As in [4], we assume that γ is a monotonically increasing function of the distance between u and v and that a function $\lambda(d)$ maps the distance d to the minimum transmission power required for successful communication at that distance. Thus, a node u is able to determine its minimum transmission power to reach v from $\lambda(d(l_u, l_v)) = S + \gamma(l_u, l_v)$.

Given this framework, a network $M = (N, L)$ can be represented by an undirected graph $G = (V, E)$, where $V = \{u_1, u_2, \ldots, u_n\}$ and vertex u_i corresponds to node m_i in M. There is an edge in G between a pair of vertices if their corresponding distance in M enables a successful communication. That is,

$$E = \{(u_i, u_j) | u_i, u_j \in V \text{ and } S + \gamma(m_i, m_j) \leq P_{max}\}.$$

3.2 Definitions

In this section, we define several graph related terms, which will be used in both algorithms and proofs. To facilitate the use of these terms, we use predicate-based notations. For all definitions, we refer to graph $G = (V, E)$ and subgraphs $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$.

- Graph G is **k-connected**, denoted by $\text{CONN}(G, k)$, if and only if there does not exist a set of $k-1$ vertices whose removal partitions G into two or more connected components. If G is k-connected, it follows that, for all $u, v \in V$, there exist at least k disjoint paths between them [10].
- Disjoint subgraphs G_i and G_j of G are **neighboring subgraphs** in G, denoted by $\text{NEIGHBOR}_G(G_i, G_j)$, if $\exists u \exists v (u \in V_i \wedge v \in V_j \wedge (u, v) \in E)$.
- Disjoint subgraphs G_i and G_j of G are **neighboring *k*-connected subgraphs** in G, denoted by $\text{NBRCONN}_G(G_i, G_j, k)$, if $\text{CONN}(G_i, k)$ and $\text{CONN}(G_j, k)$ and $\exists (u_1, v_1), \ldots, (u_k, v_k)$ such that $u_1, \ldots, u_k \in V_i$ and $v_1, \ldots, v_k \in V_j$. Note that $\text{NBRCONN}_G(G_i, G_j, 1)$ is equivalent to $\text{NEIGHBOR}_G(G_i, G_j)$.
- Let G_1, G_2, ..., G_s be a partitioning of G. Then G_i and G_j, $i \neq j$, are **peer**[3] ***k*-connected subgraphs** in G, denoted by $\text{PEERCONN}_G(G_i, G_j, k)$,[4] if either $\text{NBRCONN}_G(G_i, G_j, k)$ or $\exists G_l$ such that ($\text{NBRCONN}_G(G_l, G_j, k) \wedge \text{PEERCONN}_G(G_i, G_l, k)$).
- An **edge-reduction** of a graph $G = (V, E)$ results in a new graph $G' = (V, E')$, where $E' \subset E$. A ***k*-connectivity-preserving edge-reduction** of a k-connected graph is an edge reduction that results in a k-connected graph.

3. In this paper, the terms *adjacent* and *neighboring* are used as synonyms, while the term *peer* is a "multihop" concept involving a sequence of neighboring entities.

4. Note that $\text{PEERCONN}_G(G_i, G_j, k)$ is recursively defined.

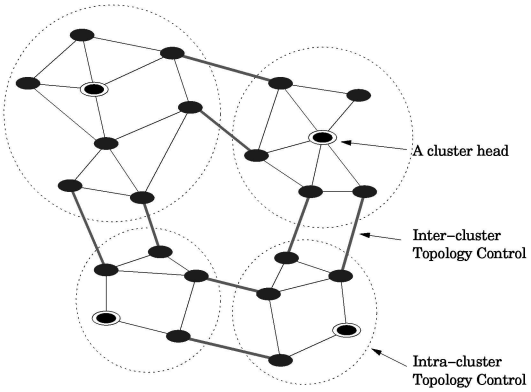


Fig. 1. Cluster-based topology control.

4 A Cluster-Based Topology Control Framework

Recall from the introduction that CLTC is a general framework for designing topology control algorithms. The design goals of CLTC are two-fold: 1) to provide scalability, adaptability, and autonomy through a hybrid approach and 2) to achieve strong connectivity in the resulting network. The CLTC framework does not require the global topology to be known by any entity. Rather, the framework relies on **clustering**[5] where nodes autonomously form groups (clusters) and select a cluster head for each cluster. A centralized topology control algorithm is then applied to each cluster to achieve the desired connectivity within the cluster, while the desired connectivity between adjacent clusters is achieved via *localized* information sharing between adjacent cluster heads, as depicted in Fig. 1. The following subsections detail the three phases of the CLTC framework.

4.1 Phase 1: Cluster Formation

In the first phase, in a distributed fashion, clusters are formed and cluster heads are selected. The cluster heads will take the main responsibility[6] for the subsequent two phases. Note that the operations in those two phases are independent of the specific clustering algorithm used in this phase. Any distributed clustering algorithm that can form nonoverlapping clusters and select cluster heads can be applied[7] ([11], [12], [13]).

4.2 Phase 2: Intracluster Topology Control

In this phase, each cluster head will calculate the power assignments for all of the members of its cluster such that the resulting topology of the cluster meets the given topology constraint (i.e., k-connectivity for a given k). To

5. The final topology generated by CLTC may depend on the characteristics of clusters resulted from the clustering algorithm used.

6. Note that having only a cluster head to perform subsequent operations in a centralized fashion may introduce a single point of failure within a cluster. Being a *framework* for a hybrid topology control algorithm, CLTC allows other alternatives such as a decentralized approach where information is distributed to all member nodes and each node is responsible for its topology control power assignment, hence the use of cluster heads is not necessary. However, the two approaches are equivalent from the theoretical perspective and we chose to proceed with the centralized approach.

7. Although the *operations* of the subsequent two phases do not depend on the specific clustering algorithm used in the Cluster Formation phase, the overall effectiveness of topology control depends on the characteristics of resulting clusters.

accomplish this, we assume that the cluster head has, or can obtain, the position (i.e., coordinates) of each member of the cluster. Thus, the power assignments of the cluster members can be obtained by applying an appropriate centralized algorithm, such as those presented in [3], [4].

The power assignments calculated in this phase will be distributed to the cluster members. But, the cluster members will not yet begin transmitting at these powers. Rather, it may be that these powers will be found to be inadequate as a result of Phase 3 (the intercluster topology control phase). Hence, the final power assignments for all nodes will be computed after Phase 3 is completed. Thus, during Phase 3, all nodes will still utilize their full transmission power. Only after the completion of Phase 3 will nodes start using their assigned transmission power.

Note that some clusters may not be able to achieve a k-connected topology at any legitimate power level. The nodes in these *weak* clusters will transmit at full power. Clusters that are able to achieve a k-connected topology are termed *strong* clusters.

4.3 Phase 3: Intercluster Topology Control

In this phase, connectivity between adjacent clusters is considered. Phase 3 consists of two steps. In the first step, only strong clusters are considered, while weak clusters are considered in the second step. This phase is necessary because the intracluster connectivity resulting from Phase 2 is not sufficient to guarantee global k-connectivity, even if only strong clusters are present in the network. Thus, this phase is aimed at ensuring that there are k disjoint links between adjacent clusters when k such links exist.

4.3.1 Step 1—Adjacent Strong Clusters

Throughout the description of this step (including the algorithm), any reference to a *cluster* will be taken to mean a strong cluster. Weak clusters play absolutely no role in this first step of Phase 3.

For each pair of adjacent (strong) clusters, this step ensures that there are k disjoint links between those clusters when k such links exist. For a given cluster, this calculation relies on information from the nodes of all clusters (remember this means strong clusters) containing nodes adjacent to a node of this cluster. In order to allow adjacent clusters to discover each other, every node periodically broadcasts a *hello* message[8] containing its current coordinates, its ID, and its current cluster ID. When a node A hears a hello message from a node B that belongs to a different cluster, A will place B's information in its *border list*. This border list will subsequently be reported to the cluster head.

With the border list of each cluster member in hand, the cluster head for a cluster A executes Algorithm 1 (see Fig. 2). In that algorithm, for each adjacent cluster C, the cluster head checks if there exist k disjoint links from A to C. That is accomplished by applying an algorithm (*BIMAXMATCHING*) [14] that computes a *maximum cardinality matching* in the bipartite graph defined by the nodes in respective clusters and the edges with one endpoint in each cluster. If k does not exceed the size of the maximum cardinality matching, then the cluster head selects k disjoint links that meet the specified optimization objective. A specific implementation of algorithms for selecting a set of k disjoint links that satisfy minimizing maximum power and minimizing total power, which is the algorithm *FindKLinks()* shown in Algorithm 1 (see Fig. 2), are given in Section 6. When they don't exist, k disjoint links from A to C, the cluster head for A simply assigns full power to each of its members who are adjacent to a node in C (i.e., each node that heard a hello message from a node in C). Thus, in this case, all of the links between clusters A and C are preserved. Note that the connectivity preservation alone does not guarantee k-connectivity between clusters A and C. However, global k-connectivity is guaranteed after the completion of Phase 3 when connectivities with other neighboring clusters are already established, as presented in the proofs in Section 5.

In Algorithm 1 (see Fig. 2), after computing the links to each adjacent strong cluster, the cluster head performs an optimization to allow the links between certain adjacent clusters to be removed, while preserving the k-connectivity of the resulting topology (the correctness of this optimization is provided by Lemma 5.6). In this context, $P_{IC}(C_a, C_b)$ is defined as follows: When the number of disjoint links between C_a and C_b is less than k, $P_{IC}(C_a, C_b)$ is ∞; otherwise, when the optimization objective is MINMAX, $P_{IC}(C_a, C_b)$ is the maximum power of the selected k links and, when the optimization objective is MINTOTAL, $P_{IC}(C_a, C_b)$ is the total power of the selected k links. Then, a strong cluster C_s will not establish the intercluster links with a neighboring strong cluster C_p if it observes that there exists another neighboring strong cluster, say C_q, where C_q is also a neighbor of C_p and both $P_{IC}(C_s, C_q)$ and $P_{IC}(C_q, C_p)$ are strictly less than $P_{IC}(C_s, C_p)$, as illustrated in Fig. 3.

Note that $P_{IC}(C_s, C_q)$ and $P_{IC}(C_s, C_p)$ can be computed directly by the cluster head of C_s (C_s has the complete border information). However, this is not the case for $P_{IC}(C_q, C_p)$. To obtain this information, the cluster heads rely on the local exchange of P_{IC} vectors with adjacent strong clusters.

4.3.2 Step 2—Handling Weak Clusters

Recall that weak clusters (i.e., those that are not able to achieve k-connectivity during the intracluster phase) did not participate in Step 1 nor were they considered by other clusters. It is critical, however, that we maintain the original connectivity associated with a weak cluster in order to guarantee that the network in its entirety is k-connected after applying the CLTC framework (this is proven in Section 5). Thus, in this step, each node A that belongs to a weak cluster will inform each node B within transmission range (e.g., by means of setting a flag in a hello message) that B should utilize their maximum transmission power in order to preserve A's original connectivity to the rest of the network.

After Phase 3 is completed, each node is assigned a transmission power which is the largest of the transmission powers assigned to the node in Phase 2, Phase 3 Step 1, and Phase 3 Step 2.

8. *Hello* messages are always broadcast at full transmission power.

Algorithm 1 Inter-cluster Framework

Input: (at cluster G_s=(V_s, E_s))
k (required connectivity)
Output:
Phase 3 power assignments for nodes in G_s.
Begin:
for all G_i s.t. NEIGHBOR$_G$(G_s,G_i) **do**
  $V' \leftarrow \{v | v \in G_i$ and v is adjacent to $G_s\}$.
  Construct $G_{si} = (V_s \cup V', E_{si})$ where $E_{si} = \{(u,v) | u \in V_s \wedge v \in V'$, and $S + \gamma(l_u, l_v) \leq P_{max}\}$
  if *BIMAXMATCHING*$(G_{si}) \geq k$ **then**
    $(LS(G_s, G_i),\ P_{IC}(G_s, G_i)) \leftarrow$ ***FindKLinks***(G_{si}, k)
  else
    $P_{IC}(G_s, G_i) \leftarrow \infty$
    add all $e = (u,v) \in E_{si}$ with $p_u = P_{max}$ to $LS(G_s, G_i)$
  end if
  Send $P_{IC}(G_s, G_i)$ to neighboring clusters
end for
Collect P_{IC} vectors from neighboring clusters
LIST $\leftarrow \emptyset$
for all G_p s.t. NEIGHBOR$_G$(G_s,G_p) **do**
  if $\sim \exists G_q$ s.t. NEIGHBOR$_G$(G_s,G_q) $\wedge$ NEIGHBOR$_G$(G_s,G_q) $\wedge$
    $(P_{IC}(G_s, G_q) < P_{IC}(G_s, G_p)) \wedge P_{IC}(G_q, G_p) < P_{IC}(G_s, G_p))$ **then**
    add $LS(G_s, G_p)$ to *LIST*
  end if
end for
$Ph3[] \leftarrow 0$
for all p_u in *LIST* **do**
  $Ph3[u] \leftarrow Max(Ph3[u], p_u)$
end for
Return $Ph3$

Fig. 2. Intercluster framework.

5 Proof of Correctness

In this section, we prove that the CLTC framework guarantees k-connectivity. This result is stated as the following theorem.

Theorem 1. *Let $G = (V, E)$ be the induced graph when every node uses its full transmission power. Let $G' = (V, E')$ be the induced graph where every node uses the transmission power assigned by CLTC. Then, CONN(G, k) $\iff$ CONN(G', k).*

Prior to proving Theorem 1, several lemmas utilized in that proof are presented. Throughout this section, the following notations will be used:

- If $G_i = (V_i, E_i)$, $G_j = (V_j, E_j)$, and $G = (V, E)$, then $G_i \cup_G G_j = (V', E')$, where $V' = V_i \cup V_j$ and
$$E' = E_i \cup E_j \cup \{(u,v) \mid u \in V_i \wedge v \in V_j \wedge (u,v) \in E\}$$
- If $G = (V, E)$ and $G_i = (V_i, E_i)$, then $G - G_i = (V', E')$, where $V' = V - V_i$ and
$$E' = \{(u,v) \mid (u,v) \in E \wedge u, v \in V'\}.$$

The proof of Theorem 1 will depend on several lemmas associated with local and global k-connectivity. In this regard, we begin by showing that, in any graph $G = (V, E)$, the union of *neighboring k-connected* subgraphs are k-connected.

Lemma 1. *Let $H_i = (V_i, E_i)$ and $H_j = (V_j, E_j)$ be subgraphs of G. If NBRCONN$_G$(H_i, H_j, k), then $H_i \cup_G H_j$ is k-connected.*

Proof. We need to show that $\forall a$, $b \in V_i \cup_G V_j$, there exist k disjoint paths between them. Given NBRCONN$_G$ (H_i, H_j, k), we have CONN(H_i, k) $\wedge$ CONN(H_j, k). Hence, $\forall a$, $b \in V_i$, or $a, b \in V_j$, a and b have at least k disjoint paths between them.

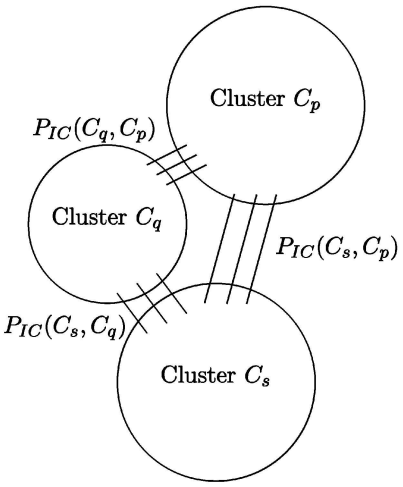


Fig. 3. Intercluster links between C_s and C_p are not necessary if there exists an adjacent cluster C_q such that both $P_{IC}(C_s, C_q)$ and $P_{IC}(C_q, C_p)$ are strictly less than $P_{IC}(C_s, C_p)$.

In the case that $a \in V_i$ and $b \in V_j$, we show that there exists a path between a and b after removing $k-1$ nodes. Given NBRCONN$_G(H_i, H_j, k)$, H_i and H_j share k disjoint links. Therefore, by removing any $k-1$ nodes from $V_i \cup V_j$, there exists at least one link (u, v) that is intact, where $u \in V_i \wedge v \in V_j$. Since CONN$(H_i, k)$, there exist at least k disjoint paths from a to u and, thus, removing any $k-1$ nodes will not disconnect a and u. Similarly, for all $b \in V_j$, b will not be disconnected from v. Therefore, there exists a path between a and b. □

By applying Lemma 1 inductively, it can be shown that, in any graph G, the union of peer k-connected subgraphs are k-connected:

Corollary 1. *Let $\{H_i,\ i = 1, \ldots, m\}$ be a partitioning of G. Let S_c be a maximal set of indices such that $\forall i, j \in S_c$, PEERCONN$_G(H_i, H_j, k)$. Then, $\cup_{G\ i \in S_c} H_i$ is* k*-connected.*

The following lemma shows that, by applying a k-connectivity-preserving edge-reduction to a subgraph of any k-connected graph G, the resulting graph is also k-connected.

Lemma 2. *Let $H_c = (V_c, E_c)$ be a subgraph of G and let H'_c be an edge-reduction of H_c. Let $G' = (V'_c, E'_c)$ be $(G - H_c) \cup_G H'_c$. Then, CONN$(H_c, k) \wedge$ CONN$(H'_c, k) \wedge$ CONN$(G, k) \Rightarrow$ CONN(G', k).*

Proof. By the definition of k-connectivity, it is sufficient to show that $\forall\, u, v \in G'$, u and v are connected after removing $(k-1)$ nodes from G'. There are three cases:

1. $u, v \in V_c$: Trivially, this follows from CONN(H'_c, k).
2. $u \in V_c \wedge v \in V - V_c$: Given CONN$(G, k)$, u and v have k disjoint paths between them in G. Therefore, by removing any $(k-1)$ nodes from G, there exists a path p that is intact. If $p \subseteq E - E_c$, then u and v are connected through p, which is intact in G'. Otherwise, let $p = \{(u, x_1), \ldots, (x_p, g), (g, y_1), \ldots, (y_q, v)\} \equiv \overline{ux_1 \cdots x_p g y_1 \cdots y_q v}$, where $y_1, \ldots, y_q$, $v \in V - V_c \wedge g \in V_c$, as shown in Fig. 4a. Given CONN$(H'_c, k) \wedge u, g \in V_c$, u and g have k disjoint paths between them in H'_c. This implies that the removal of the $(k-1)$ nodes will not disconnect u and g in H'_c. Let this path be $\overline{uz_1 \cdots z_r g}$, where $u, z_1, \ldots, z_r \in V_c$. It is clear that path
$$p' = \overline{uz_1 \cdots z_r g y_1 \cdots y_q v}$$
connects u and v and $p' \subseteq E'$.
3. $u, v \in V - V_c$: Given CONN(G, k), u and v have k disjoint paths between them in G. Therefore, by removing any $(k-1)$ nodes from G, there exists a path p that is intact. If $p \subseteq E - E_c$, then u and v are connected through p, which is intact in G'. Otherwise, let $p = \overline{ux_1 \cdots x_p g_1 y_1 \cdots y_q g_2 z_1 \cdots z_r v}$, where
$$u, x_1, \ldots, x_p, z_1, \ldots, z_r, v \in V - V_c\ \wedge\ g_1, g_2 \in V_c,$$
as shown in Fig. 4b. Given CONN$(H'_c, k) \wedge\ g_1, g_2 \in V_c, g_1$, and g_2 have k disjoint paths between them in H'_c. This implies that the removal of the $(k-1)$ nodes will not disconnect g_1 and g_2 in H'_c. Let this path be $\overline{g_1 w_1 \cdots w_t g_2}$, where $g_1, w_1, \ldots, w_t g_2 \in V_c$. It is clear that path $p' = \overline{ux_1 \cdots x_p g_1 w_1 \cdots w_t g_2 z_1 \cdots z_r v}$ connects u and v and $p' \subseteq E'$.

The lemma follows from the three cases 1, 2, and 3. □

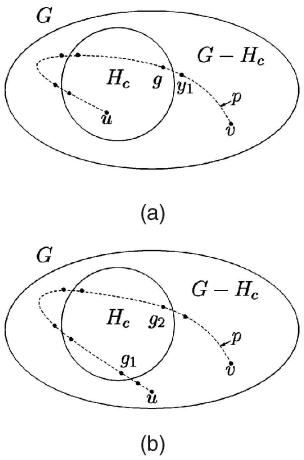


Fig. 4. Diagrams illustrating the proof of Lemma 2.

By induction, it can be shown that a k-connectivity-preserving edge-reduction of multiple subgraphs preserves the k-connectivity of the containing graph:

Corollary 2. *Let $G = (V, E)$ be any* k*-connected graph. Let $G_1, G_2, \ldots, G_m$ be* k*-connected subgraphs in G, where $m > 0, G_i = (V_i, E_i), V_i \subset V$, and $E_i \subset E$. If the subgraphs $G'_i = (V_i, E'_i)$, where $E'_i \subset E_i$ and $i = 1, \ldots, m$, are each* k*-connected, then $G' = (G - (\cup_{G_{i=1}^m} G_i)) \cup_G (\cup_{G_{i=1}^m} G'_i)$, is* k*-connected.*

Recall that CLTC framework includes an optimization step to remove redundant intercluster links. The following lemma shows that the optimization preserves k-connectivity. In the following lemma, G, G', G_i, and G'_i are defined as follows. Let G and G' be as defined in Theorem 1: Let $G_i = (V_i, E_i), i = 1, \ldots, m$, be the clusters resulting from

Phase 1 of the CLTC framework, where V_i is the set of nodes in cluster i and $E_i = \{(u,v) \in E | u,v \in V_i\}$. Let $G'_i = (V_i, E'_i)$, where $E'_i = E_i \cap E'$. That is, G'_i is the resulting subgraph of cluster i at the conclusion of the execution of the CLTC framework.

Lemma 3. $\forall i,j, 1 \le i < j \le m$, $PEERCONN_G(G_i, G_j, k) \Rightarrow PEERCONN_{G'}(G'_i, G'_j, k)$.

Proof. Let G_C be a graph representing the cluster-level connectivity of G. Formally, $G_C = (V_C, E_C)$ where $V_C = \{G_1, \ldots, G_m\}$ and $E_C = \{(G_i, G_j) | \text{NBRCONN}_G(G_i, G_j, k)\}$. Each edge $(G_i, G_j) \in E_C$ is assigned a weight $w(G_i, G_j)$, where $w(G_i, G_j) = P_{IC}(G_i, G_j)$. Let G'_C be the cluster-level representation of G', i.e., $G'_C = (V_C, E'_C)$ where $E'_C = \{(G_i, G_j) | \text{ NBRCONN}'_G(G_i), G_j, k)\}$. Since the intracluster algorithms preserve the k-connectivity of a cluster, CONN$(G_i, k) \Rightarrow$ CONN(G'_i, k). Hence, to prove the lemma, it suffices to show that $(G_i, G_j) \in E_C \Rightarrow G_i$ and G_j are connected in G'_C. By ordering the edges in E_C by nondecreasing weights, we can use induction on the rank of an edge in this ordering. Suppose that the ordering is $(e_1, e_2, \ldots, e_s)$, where $s = |E_C|$. As the base case, the pair of clusters corresponding to the edge e_1 are always directly connected in G'_C. Therefore, $e_1 \in E'_C$. For the inductive step, $\forall t \le s$, suppose that, for all $r < t$, the pair of clusters corresponding to e_r are connected in G'_C (either directly or indirectly). Suppose $e_t = (G_i, G_j)$. The only reason why $e_t \notin E'_C$ (G_i and G_j are not directly connected in G'_C) is that there exists another cluster G_l, where $P_{IC}(G_i, G_l) < P_{IC}(G_i, G_j)$ and $P_{IC}(G_l, G_j) < P_{IC}(G_i, G_j)$. Since the edges (G_i, G_l) and (G_l, G_j) come before (G_i, G_j) in the ordering, by the induction hypothesis, G_i and G_l are already connected in G'_C, as well as G_l and G_j. Therefore, G_i and G_j are also connected in G'_C. □

Finally, we have the proof of Theorem 1. In this proof, G_i and G'_i, $i = 1, \ldots, m$ are as defined in Lemma 3.

Proof of Theorem 1. For the clusters G_i, where CONN(G_i, k) is TRUE, we partition those clusters into sets $S_1, \ldots, S_c$, where each set contains clusters that are peer k-connected in G. Formally, $\forall q = 1, \ldots, c$ $[(G_i \in S_q \wedge \text{PEERCONN}_G (G_i, G_j, k)) \Rightarrow G_j \in S_q]$. Now, define sets $S'_1, \ldots, S'_c$, such that $\forall i$, $G_i \in S_q \Rightarrow G'_i \in S'_q$. By Lemma 3, for each

$$S'_q = \{G'_{q_1}, \ldots, G'_{q_r}\}, \forall\, 1 \le i < j \le r,$$

PEERCONN$'_G$ (G'_{q_i}, G'_{q_j}, k) is TRUE. Then, define a subgraph of G', termed $G_{S'_q} = (V_{S'_q}, E_{S'_q})$, where

$$V_{S'_q} = \{v \mid v \in V_{q_i} \wedge G'_{q_i} \in S'_q\},$$

and

$$E_{S'_q} = \{(u,v) \mid (u,v) \in E' \wedge u,v \in V_{S'_q}\}.$$

Since $G_{S'_q}$ comprises only peer k-connected subgraphs, it follows from Corollary 1 that $G_{S'_q}$ is also k-connected.

As described in Section 4, the intracluster and intercluster algorithms always include in G' all of the edges in G that are adjacent to any vertex that is not in any peer k-connectivity groups (i.e., $\forall (u,v) \in E$, if u or v is not in $G_{S'_i}$, $1 \le i \le c$, then $(u,v) \in E'$) and all of the edges between different peer k-connectivity groups (i.e., $\forall (u,v) \in E$, if $u \in G_{S'_i}$, $v \in G_{S'_j}$ and $i \ne j$, then $(u,v) \in E'$). It follows from Corollary 2 that G' is k-connected as long as the original topology G is k-connected. □

6 Algorithms for Topology Control

In the prior section, we described CLTC, a *framework* for the design of hybrid topology control algorithms that guarantee strong connectivity,[9] while attempting to optimize a given function of the power specifications of the nodes. To utilize the CLTC framework, there are three algorithms that must be specified: the clustering algorithm in Phase 1, the intracluster topology control algorithm of Phase 2, and, the intercluster topology control method of Phase 3. Since there are many algorithms available for clustering, we do not discuss clustering algorithms in detail. However, the characteristics of the clustering algorithm and the resulting clusters may affect the efficiency of topology control. For instance, multihop clusters can provide more flexibility in terms of the trade off between efficiency and scalability. In addition, clusters with larger coverage may result in more efficient (intracluster) topology, while incurring more overhead on the information exchange process within clusters. In contrast, smaller coverage clusters incur less (intracluster) overhead, while resulting in a less efficient topology. In the evaluation of the CLTC framework, we used a specific clustering algorithm, named Adaptive Dynamic Backbone (ADB) [12], in Phase 1. More details on the operations of ADB are presented in Section 7. In this section, we consider specific algorithms for use in Phases 2 and 3 when the optimization criteria are MINMAX and MINTOTAL.

6.1 Intracluster Algorithms

An intracluster topology control algorithm is executed by a cluster head to compute a power assignment for the members of its cluster such that the k-connectivity of that cluster is preserved and specified optimization objectives are achieved. The CLTC framework utilizes different intracluster algorithms for different optimization objectives.

6.1.1 MinMax

In this case, an intracluster algorithm is to compute a power assignment such that the maximum power assigned is minimized. Since k-connectivity is a monotone property and testable in polynomial time, CLTC utilizes the framework in [3] to compute the optimum power assignments in polynomial time. Specifically, for 1-node-connected and 2-node-connected, the algorithms in [4] are utilized.

6.1.2 MinTotal

When the optimization objective is minimizing total power, an intracluster algorithm is to produce a power assignment to the nodes in a cluster such that the total power is minimized. However, the problem of finding an optimum power assignment is NP-hard even for 1-connectivity [2]. Hence, efficient approximation algorithms are used to produce approximate

9. Strong connectivity is guaranteed under an assumption that a network is strongly connected when nodes communicate using full transmission power.

Algorithm 2 FindKLinks (MINMAX)

1: **Input:** (at cluster G_s)

2: G_{si}, k (as defined in Algorithm 1)

3: **Output:**

$(LS(G_s, G_i), P_{IC}(G_s, G_i))$, where $LS(G_s, G_i) = \{(e_k, p_{u_k}) \mid e_k = (u_k, v_k)$ where $u_k \in G_s \wedge v_k \in G_i$, and p_{u_k} is the least transmission power required to maintain $e_k\}$, $P_{IC}(G_i, G_j)$ is the maximum transmit power among all the k disjoint links in $LS(G_s, G_i)$.

4: **Begin:**

5: $M \leftarrow \emptyset$

6: $L[] \leftarrow$ sort E_{si} in a non-decreasing order of $d(l_u, l_v)$

7: Use binary search to find the smallest j, such that $|M| \geq k$, where $M \leftarrow$ *BIMAXMATCHING*$((V_{si}, E'[j]))$ and $E'[j] = \{L[0], L[1], ..., L[j]\}$.

8: $P_{IC}(G_s, G_i) \leftarrow L[j]$

9: **for all** $e = (u, v) \in M$ **do**

10: $p_u = \lambda(d(l_u, l_v))$

11: $LS(G_s, G_i) \leftarrow LS(G_s, G_i) \cup \{(e, p_u)\}$

12: **end for**

13: **Return** $(LS(G_s, G_i), P_{IC}(G_s, G_i))$

Fig. 5. FindKLinks (MINMAX).

solutions. For the general k-connectivity, the approximation framework in [3] can be used. For 1-node-connected, the 2-approximation-algorithm presented in [2] is utilized and for 2-node-connected, the 8-approximation-algorithm presented in [3] is used.

6.2 Intercluster Algorithms

In Step 1 of Phase 3, after a cluster head collects information from all of the adjacent clusters, it performs the intercluster algorithm described in Section 4.3. Note that different algorithms are utilized at Line 9 of Algorithm 1 depending on the optimization objective. We present algorithms for MINMAX and MINTOTAL below.

6.2.1 MINMAX

With this objective, an intercluster algorithm is to select a set of k disjoint links between two clusters, such that the maximum transmission power of those links is minimized. Algorithm 2 (Fig. 5) gives a method to accomplish this. There, for each candidate maximum power value, a bipartite graph is constructed. The algorithm then computes the maximum cardinality matching (*BIMAXMATCHING*) by using algorithms in [14]. By using binary search, the smallest j such that $E'[j]$ has k disjoint links is found. Thus, the intercluster links that minimize the maximum power are produced. Since binary search considers $O(\log e)$ candidate values and *BIMAXMATCHING* takes $O(\sqrt{n}e)$ time, the running time of this algorithm is $O(\sqrt{n}e \log e)$, where n is the number of nodes in the two clusters and e is the number of edges between them.

6.2.2 MINTOTAL

To achieve k-connectivity and minimize total power, the intercluster algorithm selects k disjoint links whose total power is minimum between two neighboring clusters. When $k = 1$, it is clear that the link with the minimum transmission power is selected. When $k \geq 2$, it is easy to see that this problem is equivalent to a *Minimum Cost Network Flow* (MCNF) problem. Hence, for a general k, an algorithm for minimizing total power can be obtained by replacing Lines 6 and 7 in Algorithm 2 with an algorithm for MCNF. The fastest polynomial time algorithm known for MCNF is due to Ahuja et al. [15] and runs in time $O(ne(\log \log U)(\log nC))$, where U is the maximum capacity and C the maximum cost of an edge. Hence, in the worst case, it takes $O(ne \log n)$ time by using this algorithm, where n is the total number of nodes in the two clusters.

However, when $k = 2$, there is a better method. The method is based on the observation that if (r, t) is the minimum weight (transmission power) edge, then there exist two disjoint edges e_1 and e_2 between the two neighboring clusters such that e_1 and e_2 have the minimum total weight and at least one of e_1 or e_2 is adjacent to either r or t. The reason is the following: Let e'_1 and e'_2 be two disjoint edges with the minimum total weight and suppose that neither is adjacent to r or t. Since $w((r, t)) + w(e'_2) \leq w(e'_1) + w(e'_2)$, (r, t) and e'_2 are

Algorithm 3 Find2Links (MINTOTAL)

1: **Input:** (at cluster G_s)
$G_{si} = (V_{si}, E_{si})$ (as defined in Algorithm 1)
Note that $V_{si} = V_s \cup V_i$.
2: **Output:**
$(LS(G_s, G_i), P_{IC}(G_s, G_i))$ (see Algorithm 2)
3: **Begin:**
4: $S \leftarrow E_{si}$
5: **for all** $u \in V_s$ **do**
6: **if** $|\{(u, v')|(u, v') \in S\}| > 2$ **then**
7: $S \leftarrow S$ - {all $(u, v') \in S$ except two such edges of smallest weight}
8: **end if**
9: **end for**
10: **for all** $v \in V_i$ **do**
11: **if** $|\{(u', v)|(u', v) \in S\}| > 2$ **then**
12: $S \leftarrow S$ - {all $(u', v) \in S$ except two such edges of smallest weight}
13: **end if**
14: **end for**
15: $(r, t) \leftarrow Min_{e \in S}(\lambda(d(e)))$
16: $E' \leftarrow$ {the four smallest weight edges in S other than (r, t)}
17: $P_{IC}(G_s, G_i) \leftarrow \infty$
18: **for all** $(u, v) \in E_{si}$, where $u = r$ or $v = t$ **do**
19: **if** $\exists e \in E'$, e is disjoint from (u, v) **then**
20: $edge \leftarrow$ the minimum weight edge in E' that is disjoint from (u, v)
21: **if** $\lambda(d(u, v)) + \lambda(d(edge)) < P_{IC}(G_s, G_i)$ **then**
22: $e_1 \leftarrow (u, v)$
23: $e_2 \leftarrow edge$
24: $P_{IC}(G_s, G_i) \leftarrow \lambda(d(u, v)) + \lambda(d(edge))$
25: **end if**
26: **end if**
27: **end for**
28: $LS(G_s, G_i) \leftarrow \{(e_1, \lambda(d(e_1))), (e_2, \lambda(d(e_2)))\}$
29: **Return** $(LS(G_s, G_i), P_{IC}(G_s, G_i))$

Fig. 6. Find2Links (MINTOTAL).

disjoint edges with the minimum total weight, and (r, t) is adjacent to r and t. Using this observation, we provide Algorithm 3 (Fig. 6) for use when $k = 2$. In that algorithm, for each edge (u, v) that is adjacent to (r, t), a corresponding $edge$ is found which has the minimum weight among those edges that are disjoint with (u, v). By selecting the (u, v) and $edge$ that have the minimum total weight, Algorithm 3 produces two disjoint intercluster links with the minimum total power. Since the first **for** loop (line 5 to line 9) takes $O(e)$ time and the remaining part of the algorithm takes $O(n)$ time, the running time of Algorithm 3 is only $O(e + n)$.

7 Message Complexity Analysis

We study the message complexity of an implementation of CLTC, namely CLTC-A, by considering the total number of messages exchanged during the three phases of CLTC-A. Then, we compare CLTC-A with a centralized approach, where each node collects information from all other nodes, then computes its power level independently.

In the clustering phase, CLTC-A utilizes the Adaptive Dynamic Backbone (ADB) distributed clustering algorithm. To evaluate the complexity, we describe the major steps of ADB. When starting up, each node broadcasts hello messages containing its ID, cluster ID, the parameters reflecting its appropriateness (*height* or *weight*) of being a cluster head such as node degree, and its geographical coordinates obtained from GPS. Based on the information collected from its neighbors, each node individually evaluates whether it remains the cluster head within its coverage or becomes a member of another cluster. After the clusters are formed, the connecting process starts. When

TABLE 1
Average Message Complexity Analysis of CLTC-A

Steps in CLTC-A	Number of messages per cluster
Phase 1: ADB	
Each node broadcasts a hello message to announce its existence	N_c
Each node announces its role by broadcasting a hello message	N_c
All border nodes report to their cluster head	$N_c R_{BD}$
Cluster head informs connecting nodes	1
Connecting nodes inform connecting nodes of other clusters	C_{NB}
Connecting nodes of other clusters inform their cluster heads	C_{NB}
Phase 2: Intra-cluster algorithm	
Cluster head compute power assignments for members	0
Phase 3: Inter-cluster algorithm	
Border nodes send their border lists to the cluster head	$N_c R_{BD}$
Cluster head distributes the P_{IC} vector	$2C_{NB}+1$
Cluster head sends each member its power assignment	1

node A hears hello messages from node B of neighboring clusters whose cluster IDs are lower than itself (we call these peer connecting nodes), A informs its cluster head regarding that neighboring cluster. Once a cluster head receives information from all border nodes, it selects one connecting node (backbone) corresponding to each neighboring cluster and informs the selected backbone nodes. A chosen backbone node A then notifies its peer connecting node B to serve as connecting nodes on the backbone as well. Further, B notifies its cluster head about the newly formed part of the backbone. Note that ADB was originally designed to operate in a dynamic environment and allow multihop clusters to be formed. However, in the current study of CLTC-A, we consider a static ad hoc network which forms only one-hop coverage clusters. Note that any cluster heads of two neighboring clusters are at most three hops from each other. The extra capabilities of ADB will benefit CLTC-A in future work to accommodate mobile ad hoc networks. Readers are referred to [12] for additional details on ADB.

The following parameters are used in the analysis. Let N be the total number of nodes in the network, E be the number of links in the network, C be the number of clusters, N_c be the average number of nodes per cluster ($N_c = \frac{N}{C}$), R_{BD} be the average fraction of nodes that are border nodes in a cluster ($0 < R_{BD} < 1$) and C_{NB} be the average number of neighboring clusters ($0 \leq C_{NB} < C$).

Table 1 shows the message overhead incurred by each phase to complete CLTC-A for each cluster. Each phase is broken down into its major steps. The total number of messages (the second column in Table 1) required to complete CLTC is $C \cdot (4C_{NB} + 2N_c + 2N_c R_{BD} + 3))$, which is $O(N + C_{NB}C)$ in the worst case.

It is appropriate to assume that the clustering algorithm is able to maintain the number of clusters, C, and the average number of neighboring clusters, C_{NB}, since these are standard objectives of most clustering algorithms. When C and C_{NB} take the values $O(1)$, $O(\log N)$, $O(\sqrt{N})$, and $O(N)$, the total number of messages is shown in Fig. 7.

Note that, in the majority of the cases, the average number of messages required per node is $O(1)$, which agrees with the statistics obtained in our simulation study (see Fig. 12). The total number of messages is super-linear only when C is $O(N)$ and C_{NB} is greater than $O(1)$. That worst scenario can be avoided by choosing proper parameters (e.g., cluster size) of ADB.

In a centralized approach, each node sends a message that consists of its coordinates to all other nodes in the network. Since no node has global knowledge of the network, we assume the message distribution is done by plain flooding. Thus, a message from one node may travel every link of the network. Therefore, the total number of messages is $O(N^2)$.

C_{NB}/C	$O(1)$	$O(\log N)$	$O(\sqrt{N})$	$O(N)$
$O(1)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
$O(\log N)$	/	$O(N)$	$O(N)$	$O(N \log N)$
$O(\sqrt{N})$	/	/	$O(N)$	$O(N^{\frac{3}{2}})$
$O(N)$	/	/	/	$O(N^2)$

Fig. 7.

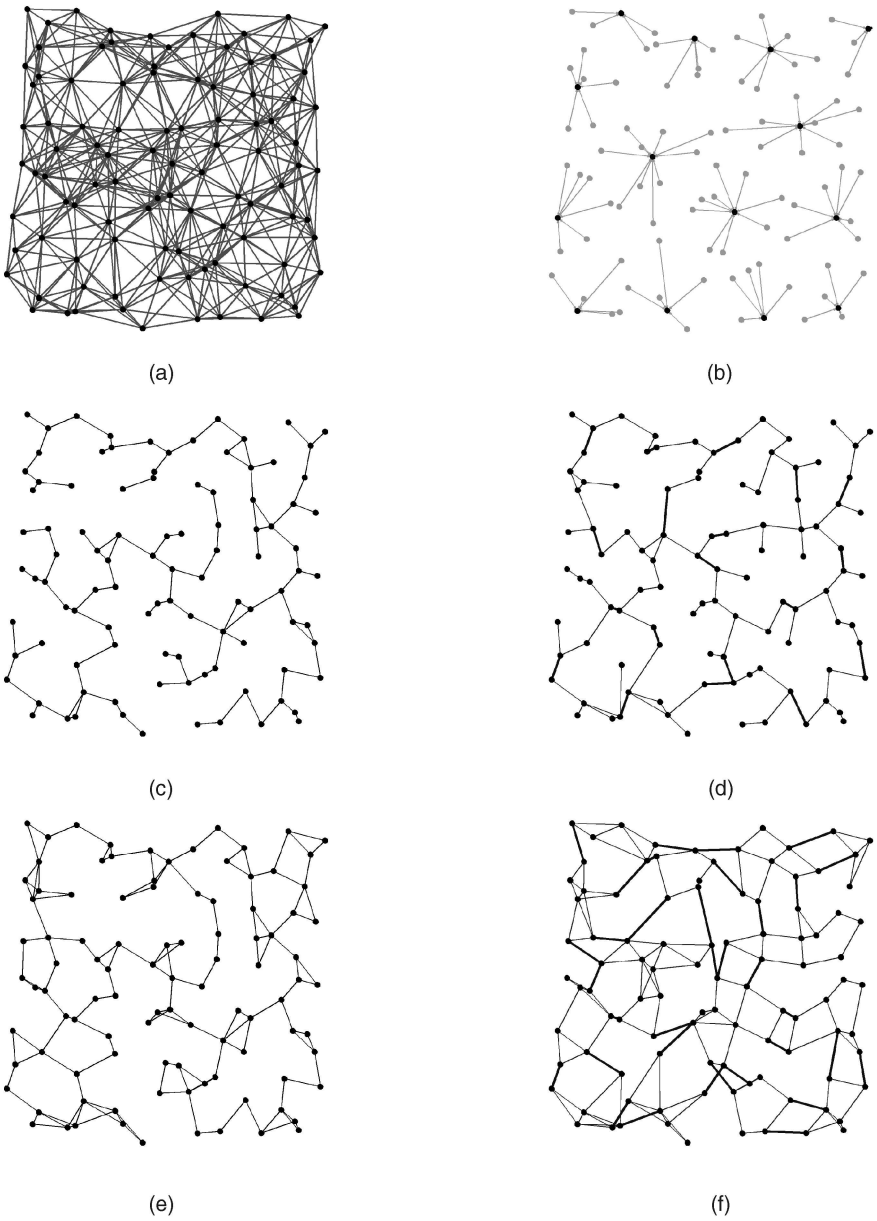


Fig. 8. Network topologies of 100 nodes with different topology control settings (for MINMAX). (a) Without topology control. (b) After applying ADB clustering algorithm. (c) $k = 1$, after applying RR1 algorithm. (d) $k = 1$, after applying CLTC-A (MINMAX). (e) $k = 2$, after applying RR2 algorithm. (f) $k = 2$, after applying CLTC-A (MINMAX).

Thus, CLTC has at least one magnitude less overhead than a centralized approach. In Section 8, we show the experimental results on message complexity collected during our simulation study.

8 Simulation Study

We have conducted a simulation study to determine the effectiveness of CLTC-A and to study the message complexity as described in Section 7. These simulations were carried out using the QualNet simulator [16].

In this study, 20 networks were randomly generated on a terrain of size 1,500×1,500 m^2. In order to study the effect of cluster size on the resulting topologies, we varied the number of nodes inside the terrain, between 100, 125, 150, and 175 nodes. When operating at full transmission power, each node had an effective communication range of 350 m.

For each network, we considered:

- optimization objectives: MINMAX and MINTOTAL,
- k-connectivity: $k = 1$ and $k = 2$,
- algorithms: centralized and CLTC-A.

Thus, for each network, there are eight results. The centralized algorithms utilized were:

- MINMAX k = 1, 2, RR1 and RR2 from [4],
- MINTOTAL k = 1, K3P from [2],
- MINTOTAL k = 2, LLMRR from [3].

Relative to CLTC-A, recall that, in Phase 1, the clustering algorithm used is ADB [12], configured such that each node is only one hop away from a cluster head when all of the nodes

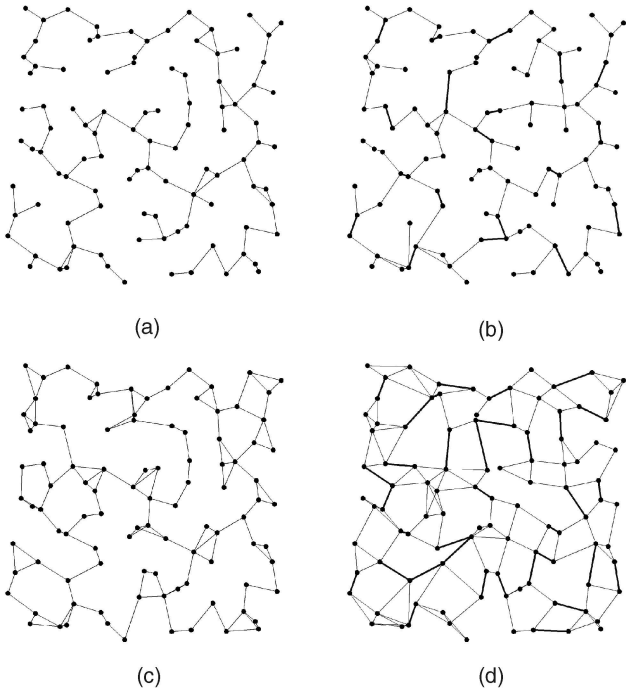


Fig. 9. Network topologies of 100 nodes with different topology control settings (for MinTotal). (a) $k = 1$, after applying K3P algorithm. (b) $k = 1$, after applying CLTC-A (MINTOTAL). (c) $k = 2$, after applying LLMRR algorithm. (d) $k = 2$, after applying CLTC-A (MINTOTAL).

use full transmission power. In Phases 2 and 3, we utilize the algorithms specified in Section 4. In our simulations, for Phase 1 of CLTC-A, ADB generates clusters where the average number of nodes per cluster is 7.79, 9.87, 12.36, and 13.94, where the number of nodes in the network is 100, 125, 150, and 175, respectively. Note that, by varying the number of nodes in the network while maintaining other parameters such as the transmission range and terrain size, we implicitly adjust the degree of centralized and distributed algorithms in CLTC-A. In other words, if there are more nodes in the network, then the clusters are larger and, in CLTC-A, intracluster algorithm has a more significant effect.

Prior to providing the experimental results regarding power, we first provide, in Fig. 8, a look at the actual topologies for one simulated network. Three pairs of figures are given there.

- Fig. 8a shows the topology when all nodes communicate using full transmission power. Fig. 8b shows the topology after Phase 1 of CLTC-A. This is the output of the ADB clustering algorithm, prior to the execution of any topology control algorithm.
- Fig. 8c is the topology resulting from RR1 when the topology property is simply that the network be connected (i.e., $k = 1$). Fig. 8d is the topology produced by CLTC-A for MINMAX when $k = 1$, where the thin lines indicate intracluster links and the thick solid lines indicate intercluster links. Note that only those links that are selected by CLTC-A are shown.
- Figs. 8e and 8f are analogous to Figs. 8c and 8d, but $k = 2$.

Fig. 9 also illustrates topologies obtained from various topology control algorithms which aim to achieve minimum total power. Two pairs of figures are organized in a similar fashion, as in Fig. 8.

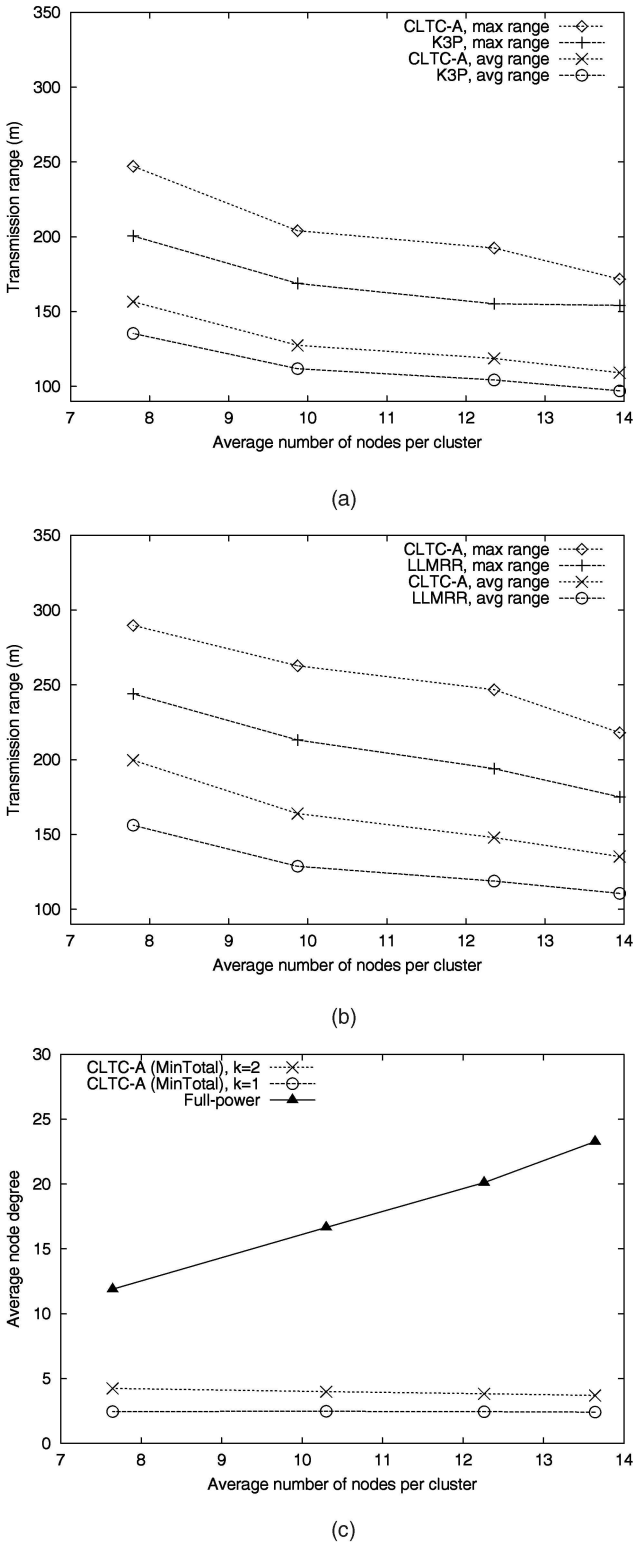
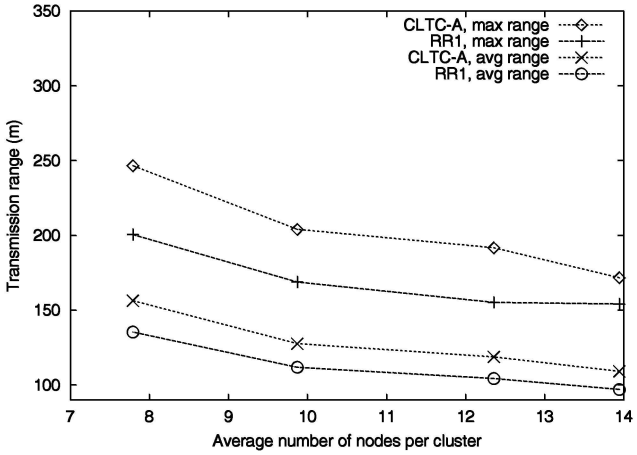
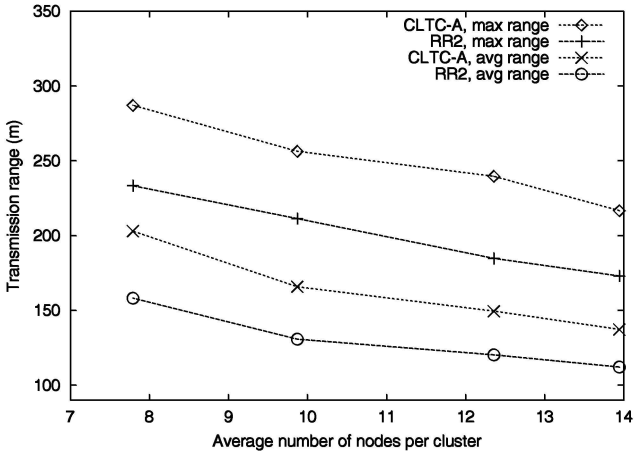


Fig. 10. Results from various topology control algorithms showing average and maximum transmission range when (a) $k = 1$ and (b) $k = 2$, and (c) average node degree when the optimization objective is MinTotal.

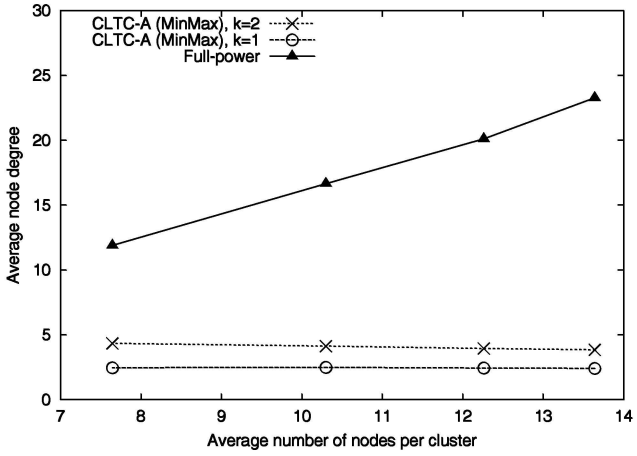
- Fig. 9a shows the topology resulted from applying a centralized algorithm, K3P, to the entire network,



(a)



(b)



(c)

Fig. 11. Results from various topology control algorithms showing average and maximum transmission range when (a) $k = 1$ and (b) $k = 2$, and (c) average node degree when the optimization objective is MinMax.

while Fig. 9b illustrates the topology obtained from CLTC-A when $k = 1$.

- Fig. 9c and 9d depict the topologies obtained from a centralized algorithm, LLMRR, and CLTC-A, respectively, when $k = 2$.

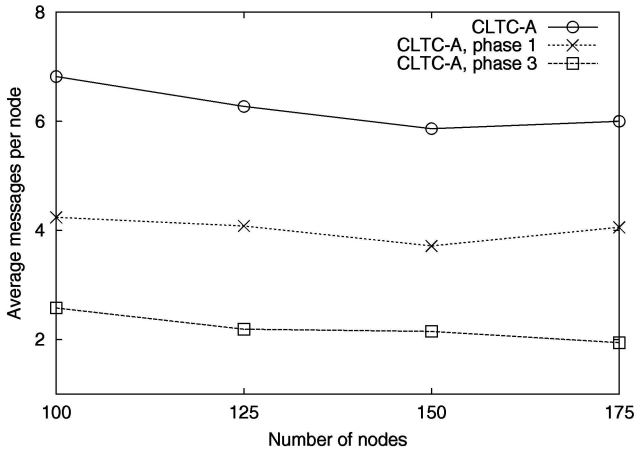


Fig. 12. Number of message exchanges per node in CLTC-A (MINMAX, $k = 1$) when the number of nodes increases.

Fig. 10 illustrates the results of the simulations when the optimization objective is MINTOTAL. In the figures, we show maximum and average transmission range obtained from various topology control algorithms. Note that minimizing total transmission power is equivalent to minimizing average transmission power. It is evident from those results that CLTC-A is *very effective* in reducing the transmission range of each node. Recall that the full-power transmission range is 350 m. When $k = 1$ (Fig. 10a), CLTC-A reduces the maximum transmission range to 245 m when there are about eight nodes per cluster and as low as 175 m when there are 14 nodes per cluster. These transmission ranges are approximately 11 to 23 percent higher than the maximum transmission ranges assigned by K3P centralized algorithm. For the average transmission range, CLTC-A reduces the range to 160 m when there are eight nodes per cluster and as low as 120 m when the density is 14 nodes per cluster, which are approximately 13 to 16 percent higher than those of K3P.

When $k = 2$ (as shown in Fig. 10b), the maximum and average ranges resulting from both the CLTC-A and LLMRR algorithms are larger than when $k = 1$. That is expected because 2-connected is a stronger property than 1-connected. Moreover, the difference between CLTC-A and the centralized algorithm when $k = 2$ is in a greater range than when $k = 1$. This is the consequence of having to maintain one more longer-range link between adjacent clusters and one more additional disjoint path from each node to other nodes within all clusters. The maximum ranges of CLTC-A are approximately 19 to 25 percent higher than those assigned by LLMRR and the average ranges of CLTC-A are approximately 22 to 28 percent higher than those assigned by LLMRR.

Fig. 10c shows the average node degrees produced by CLTC-A versus a full transmission power network. More importantly, the node degree of a network with CLTC-A does not depend on the size or density of the network. The results of MINMAX are similar to those of MINTOTAL and are illustrated in Fig. 11.

Fig. 12 illustrates the number of message exchanges required per node to complete CLTC-A in our simulation environment. The message complexity analysis in Section 7 is confirmed by the result shown in the figure. When the number of nodes in the network increases from 100 to 175 nodes, the average number of messages required per node in CLTC-A does not increase. This shows the CLTC-A has little extra

overhead besides the ADB clustering algorithm and ADB is very effective in maintaining an appropriate number of clusters. This also confirms the scalability of CLTC-A.

9 Conclusion

In this paper, we have proposed the *Cluster-based Topology Control* (CLTC) framework for a hybrid approach to control topology using transmission power adjustment. CLTC employs a clustering algorithm and each cluster locally exchanges information among its own members and with neighboring clusters to achieve k-connectivity for the network in its entirety. We proved that the CLTC framework guarantees global k-connectivity as long as the original topology is k-connected. The framework allows topology control algorithms with different optimization objectives to be utilized. We also conducted simulation studies to demonstrate the effectiveness and message complexity of CLTC. The results, in comparison to a maximum transmission power topology, show that lower average transmission range and node degree and, hence, less interference and energy consumption can be achieved by CLTC while preserving global k-connectivity.

In the current design, a pair of k-connected adjacent clusters will establish either k direct paths or k indirect paths. Research is in progress to allow CLTC to establish k paths of a mix of both direct and indirect paths to further provide route diversity. As mentioned earlier, the operation of CLTC is independent of the clustering algorithm used. We are investigating a topology-control-aware clustering algorithm that forms clusters based on the value of k and allows clusters to overlap. We also plan to study the relationship between the characteristics of clusters and their impact on the efficiency and scalability of topology control. In addition, the capability of a clustering algorithm to adapt to mobility and its effect on the resulting topology will be studied. An experimental study to evaluate the application-layer performance with the existence of topology control would be a crucial step in evaluating the performance benefits of topology control algorithms. The use of directional antennas to limit the range of broadcast and the effects this may have on topology control should also be considered.

Acknowledgments

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory or the US Government. Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the US Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

References

[1] Z. Huang, C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Topology Control for Ad Hoc Networks with Directional Antennas," *Proc. 11th Int'l Conf. Computer Comm. and Networks (ICCCN),* Oct. 14-16 2002.

[2] L.M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc, "Power Consumption in Packet Radio Networks," *Proc. 14th Ann. Symp. Theoretical Aspects of Computer Science (STACS '97),* pp. 363-374, Feb. 1997.

[3] E.L. Lloyd, R. Liu, M.V. Marathe, R. Ramanathan, and S.S. Ravi, "Algorithmic Aspects of Topology Control Problems for Ad Hoc Networks," *Proc. IEEE Mobile Ad Hoc Networking and Computing (MOBIHOC),* June 2002.

[4] R. Ramanathan and R. Rosales-Hain, "Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment," *Proc. Infocom 2000,* pp. 404-413, 2000.

[5] L. Li, J.Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer, "Analysis of a Cone-Based Distributed Topology Control Algorithms for Wireless Multi-Hop Networks," *Proc. ACM Symp. Principle of Distributed Computing (PODC),* Aug. 2001.

[6] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks," *Proc. Infocom 2001,* Apr. 2001.

[7] W.T. Chen and N.F. Huang, "The Strongly Connecting Problem on Multihop Packet Radio Networks," *IEEE Trans. Comm.,* vol. 37, no. 3, Mar. 1989.

[8] V. Rodoplu and T.H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE Trans. Selected Areas in Comm.,* vol. 17, no. 8, Aug. 1999.

[9] S. Narayanaswamy, V. Kawadia, R.S. Sreenivas, and P.R. Kumar, "Power Control in Ad-Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW Protocol," *Proc. European Wireless,* pp. 156-162, Feb. 2002.

[10] R.J. Wilson and J.J. Watkins, *GRAPHS: An Introductory Approach.* John Wiley & Sons, 1990.

[11] M. Chatterjee, S.K. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks," *J. Cluster Computing,* vol. 5, no. 2, pp. 193-204, Apr. 2002.

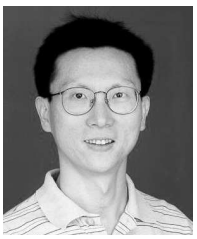
[12] C. Jaikaeo and C.-C. Shen, "Adaptive Backbone-Based Multicast for Ad Hoc Networks," *Proc. IEEE Int'l Conf. Comm. (ICC 2002),* 28 Apr.-2 May, 2002.

[13] U.C. Kozat, G. Kondylis, B. Ryu, and M.K. Marina, "Virtual Dynamic Backbone for Mobile Ad Hoc Networks," *Proc. IEEE Int'l Conf. Comm. (ICC 2001),* June 2001.

[14] J.D. Kececioglu and A.J. Pecqueur, "Computing Maximum-Cardinality Matchings in Sparse General Graphs," *Proc. Workshop on Algorithm Eng. (WAE '98),* pp. 121-132, Aug. 1998.

[15] J. van Leeuwen, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity.* Cambridge, Mass.: Elsevier and MIT Press, 1990.

[16] QualNet, http://www.scalable-networks.com, year?



Chien-Chung Shen (M'92) received the BS and MS degrees from National Chiao Tung University, Taiwan, and his PhD degree from the University of California, Los Angeles, all in computer science. He was a research scientist at Bellcore Applied Research working on control and management of broadband networks. He is now an assistant professor in the Department of Computer and Information Sciences at the University of Delaware. His research interests include ad hoc and sensor networks, control and management of broadband networks, distributed object and peer-to-peer computing, and simulation. He is a member of the IEEE and IEEE Computer Society.

Chavalit Srisathapornphat received the MS degree in computer and information sciences from the University of Delaware in 1999 and the MS degree in computer sciences from Chulalongkorn University, Bangkok, Thailand, in 1995. He is currently a doctoral candidate in the Department of Computer and Information Sciences at the University of Delaware. His research interests include mobile ad hoc and sensor networks, power control, and distributed object computing. He is a student member of the IEEE.

Rui Liu received the BS degree in mathematics from Peking University, Beijing, China, in 1998, the MS degree in applied mathematics from the University of Delaware, in 2000. He is a doctoral candidate in computer and information sciences at the University of Delaware. His research interests include design and analysis of algorithms for combinatorial optimization problems, parallel and distributed computing, and computer networks.

Zhuochuan Huang received the BE degree in computer science and technology from Tsinghua University, Peoples Republic of China, in 1998 and the MS degree in computer science from the University of Delaware in 2000. He is currently a research assistant with the Department of Computer and Information Sciences at the University of Delaware. His current research interest is the design and simulation of protocols for mobile ad hoc networks. He is a student member of the IEEE and IEEE Computer Society.

Chaiporn Jaikaeo (jaikaeo@cis.udel.edu) received the MS degree in computer and information sciences from the University of Delaware in 1999 and the BEng degree in computer engineering from Kasetsart University, Bangkok, Thailand, in 1996. He is currently a PhD student in the Computer and Information Sciences Department at the University of Delaware. His research interests include unicast and multicast routing, topology control, peer-to-peer computing, and network management for ad hoc and sensor networks. He is a student member of the IEEE.

Errol Lloyd received the undergraduate degrees in both computer science and mathematics from Penn State University and the PhD degree in computer science from the Massachusetts Institute of Technology. He is a professor of computer and information sciences at the University of Delaware. Previously, he served as a faculty member at the University of Pittsburgh and as program director for computer and computation theory at the US National Science Foundation. From 1994 to 1999, he was chair of the Department of Computer and Information Sciences at the University of Delaware. Concurrently, from 1997 to 1999, he was interim director of the University of Delaware Center for applied science and engineering in rehabilitation. His research expertise is in the design and analysis of algorithms, with a particular concentration on approximation algorithms for computationally difficult problems. He has published more than 30 journal papers and numerous conference papers. In 1989, he received a US National Science Foundation Outstanding Performance Award and, in 1994, he received the University of Delaware Faculty Excellence in Teaching Award. He is a member of the IEEE Computer Society.