Topology Control to Simultaneously Achieve Near-Optimal Node Degree and Low Path Stretch in Ad hoc Networks

Ece Gelal, Gentian Jakllari, Srikanth V. Krishnamurthy and Neal E. Young
Dept. of Computer Science and Engineering University of California, Riverside
{*ecegelal, jakllari, krish, neal*}*@cs.ucr.edu*

***Abstract*— Our objective in this paper is to design topology control algorithms such that (i) nodes have low degree and (ii) paths in the network have few hops. Low node degree is desirable in networks equipped with smart antennas and to reduce access contention. Short paths are desirable for minimizing communication delays and for better robustness to channel impairments and to mobility. Given any arbitrary unit-disc graph G representing all feasible links, our algorithms find a sparse subgraph G' having a maximum node degree of six and, for each pair of vertices u, v, having $\text{hops}_{G'}(u,v) = O(\text{hops}_G(u,v) + \log \Delta)$, where Δ is the maximum node degree in G and $\text{hops}_G(u,v)$ denotes the shortest path length from u to v in G. This result is near-optimal: (i) there is a connected UDG G in which no connected subgraph has degree less than five, and (ii) for any graph G, any bounded-degree subgraph G' must have $\text{hops}_{G'}(u,v) = \Omega(\text{hops}_G(u,v) + \log \Delta)$ for some u, v. Our distributed algorithm scales, preserves link symmetry, does not need node synchronization, and requires only $O(n)$ messages. We perform extensive simulations that quantify the performance of our algorithm in realistic scenarios.**

I. Introduction

In this paper, we design topology control algorithms that simultaneously facilitate low degree (each node communicates directly with only a few other nodes) and short paths. Our motivation for low logical degree stems primarily from networks where nodes are equipped with directional antennas or MIMO (as we elaborate later). Short paths are important for a number of reasons, including (i) maintaining low levels of packet loss – in wireless multi-hop networks links are error-prone and the probability of packet loss increases rapidly with path length; and (ii) better coping with mobility – longer paths are more likely to be disrupted due to motion.

To motivate the goals, consider a mobile ad hoc network in which nodes are equipped with directional antennas. In such a network, nodes need to keep track of their neighbors (to beamform in their direction). However, due to the reduction in the angular coverage with directional antennas, it is possible for neighbors to move out of coverage frequently. A node may simply resort to omnidirectional communications; however, omnidirectional transmissions impose severe constraints on the achievable spatial reuse. Furthermore, it has been shown that mixing omnidirectional and directional communications can lead to link asymmetry which causes problems both in media access control and in routing [13], [8]. Alternative to allowing hybrid communications in the network, each node may periodically send directional control messages to its neighbors, to allow these neighbor nodes to *track* its motion. However, in dense deployments where nodes have too many neighbors, the overhead for such control messages may be prohibitive. Instead of maintaining links with *all* of its neighbors, topology control may be invoked such that a node maintains links only with a sub-set of its neighbors. Along with maintaining connectivity with this restrictive sub-set, it would also be important to ensure that the route-lengths are not increased tremondously. Note that a similar situation arises if nodes are equipped with multi-input multi-output (or MIMO) or smart antennas. For efficiently using MIMO, a node needs to exchange channel state information (or CSI) with each of its neighbors [30]. The overhead burden is likely to be excessive if a node has a large number of neighbors.

We model the wireless network as an *arbitrary* unit disk graph (UDG) $G(V, E)$, wherein a link exists between two nodes *iff* the two nodes are within a unit distance of each other. While it is well known that the transmission range of a node can be affected by wireless channel impairments such as multipath fading and shadowing, for the simplicity of understanding, as in existing topology control papers to-day [17][29][6], we assume that each node has a unit transmission range. With the IEEE 802.11 wireless cards, this range is generally around 250 meters; however, for the purposes of modeling, it can be scaled down to a unit distance. Disk models have also been used with directional transmissions wherein a sequence of disjoint transmissions (called circular transmissions) cover a radial footprint [13]; disk shaped footprints have also been considered with MIMO [14].

The most closely comparable previous efforts for topology control are (i) algorithms that provide subgraphs with bounded hop count but no degree bound[1] [1], [11] and (ii)

This work is supported in part by, the U. S. Army Research Office under the Multi-University Research Initiative (MURI) grant W911NF-04-1-0224 and the NSF CAREER Grant 0237920.

[1]In [1],a degree bound is provided on a backbone constructed on G;

algorithms that provide a degree bound but do not provide any bounds on the hop count of paths [15], [21]. To the best of our knowledge, there is no work to date that *jointly* addresses both these objectives.

In this paper, given the unit-disc graph (UDG) G representing all feasible links, we find a sparse subgraph $G'(V, E_{G'})$ having maximum degree 6 and, for each pair of vertices u, v, having $\text{hops}_{G'}(u, v)$=$O(\text{hops}_G(u, v)+\log \Delta)$, where Δ is the maximum degree of G and $\text{hops}_G(u, v)$ denotes the shortest path length from u to v in G. We call the latter property *bounded hop stretch*. Any graph satisfying this property is a *hop-spanner*.

This result is near-optimal: (i) there is a connected UDG G in which *no* connected subgraph has degree less than 5, and (ii) for any graph G, any bounded-degree subgraph G' must have $\text{hops}_{G'}(u, v)$=$\Omega(\text{hops}_G(u, v)+\log \Delta)$ for some u, v. We prove the above claims later.

In a nutshell, our algorithm first constructs a backbone spanning a selected subset of nodes. We show that this backbone has a maximum degree of 6 and is both a Euclidean 9-spanner and an $O(1)$ hop-spanner. The algorithm then connects the remaining nodes locally to the backbone. The resulting graph is G'=$(V, E_{G'})$, whose properties are described above. We first design a centralized version (which we call LDS for "Low Degree Spanner") of our approach, which facilitates an understanding of why the properties that we claim, hold. We then present a distributed variant of the algorithm (D-LDS for Distributed-LDS) that is amenable to practical implementation. The key features of D-LDS are summarized below:

- **Guaranteed low node degree.** With D-LDS nodes have a maximum degree of 6.
- **Low-bounded hop stretch.** With D-LDS, all node pairs are connected by paths that conform to a bounded hop stretch, as previously specified.
- **Low Euclidean distance stretch.** In terms of the Euclidean distance covered, the path between any two nodes in G' is only longer by a constant factor plus an additive $\log \Delta$ term.
- **Scalability.** Nodes use only local information and do not require synchronization. Thus, the algorithm can be used in *large* ad hoc networks.
- **Low communication cost.** D-LDS quickly converges to a connected sub-graph; this process takes $O(n)$ messages in the worst case.

In addition to showing worst-case bounds for arbitrary unit disk graphs, we perform extensive simulations to understand the behavior of D-LDS with a realistic CSMA (carrier sense multiple access) based MAC protocol for facilitating message exchanges. We observe that, on average, the node degree is is only about 3 even with fairly dense scenarios. The average hop stretch is also extremely small.

The rest of the paper is organized as follows: In Section II we present our model and introduce relevant notation. We present our centralized algorithm and prove its associated propeties in Section III. Our distributed algorithm is described in Section IV. Details of our simulation experiments and results form Section V. We discuss previous related work in Section VI and we conclude in Section VII.

II. System model and notation

We model the wireless network as a graph $G(V, E)$, where the set V of vertices represents the nodes and the edges in E represent the communication links.

We assume that all nodes use the same, maximum transmission power. We use the unit disk graph (UDG) model [9]. We scale the (maximum) transmission radius of nodes to 1 unit; thus, the UDG model implies that a node pair u, v in the network can have direct communication iff their Euclidean distance is not larger than unity. When u and v have a direct link, u is said to be a *neighbor* of v and vice versa.

In our distributed scheme we assume that nodes are able to infer the direction of the sender of the messages they receive (as used in [29], [17]). This can be accomplished via the deployment of antenna arrays and using estimation techniques for computing the Angle-of-Arrival (AOA) [25]. Alternatively, nodes might be equipped with global positioning system (GPS) units to facilitate this requirement[2].

In our network model all nodes have distinct IDs (e.g. MAC address). In our distributed scheme we assume that nodes have information regarding their two-hop neighborhood [7], [28]. Two-hop neighborhood of node u is defined as the set of nodes N_1 that are neighbors of u, plus those that are neighbors of the nodes in N_1. This information can simply be collected via the use of HELLO messages [4].

We assume the communication channel is symmetric and obstacle-free, and that signal degradation occurs only due to path loss. We do not assume global clock synchronization.

In the following, we define the two data structures that are used by our algorithms.

Maximal Independent Set: Given a graph $G = (V, E)$, a subset $M \in V$ is an *independent set* if each edge in E is incident on at most one vertex in M. A maximal independent set (MIS) is an independent set such that for all vertices $v \in (V - M)$, the set $M \cup \{v\}$ is not independent; in other words, every vertex that is not in M is adjacent to some vertex in M. Note that the "Maximal Independent Set" problem is not the same as the "Maximum Independent Set" problem (the problem of computing the independent set of maximum size), which is NP-complete [26].

however, this bound is not imposed on the final topology.

[2]GPS has been used recently in [15], [16], [29] and in other topology control studies.

Our algorithm requires the computation of a maximal independent set, which can be constructed in *O(n)* time for UDGs, where *n*=|V| [22]. Distributed algorithms for finding a maximal independent set are presented in [22], [27].

Balanced Binary Tree: A non-empty binary tree to is a *balanced* binary tree, if both its left and right subtrees (T_L and T_R) are balanced binary trees and $|height(T_L) - height(T_R)| \leq 1$. In a balanced binary tree consisting of N nodes, the length of the longest path from the root to a leaf node is $O(\log N)$. An empty tree is a balanced binary tree.

Finally, we introduce the following notation:
Notation ($\angle(evf)$): If e and f are two edges incident on vertex v, we denote the angle between e and f at v as $\angle(evf)$.

III. The Low-Degree Spanner (LDS)

In this section we describe our centralized algorithm LDS. First, we show that the achieved bounds on the degree and hop stretch (as stated earlier) are near-optimal.

Lemma 1. For a subgraph G' of a unit disk graph G, connectivity cannot be guaranteed if the maximum node degree in G' is less than 5.

Proof. With the UDG model, a node can have at most five "independent"[3] neighbors [18]. Consider the star topology, where the central node u has 5 independent neighbors, each of which is exactly at unit distance from u. If we remove the edge between u and *any* neighbor, i.e., make the degree of u less than five, the topology becomes disconnected. □

Lemma 2. For a unit disk graph G, any bounded-degree subgraph G' must have $\text{hops}_{G'}(u,v) = \Omega(\text{hops}_G(u,v) + \log\Delta)$ for some u, v, where $\text{hops}_G(u,v)$ represents the hop count of the shortest path between nodes u, v in G.

Proof. For a unit disk graph G, consider any degree-Δ node u and its neighbor set ν. In any subgraph G' with maximum degree δ, let k be the maximum hop-distance from u to a node in ν. As there are at most δ^{k+1} nodes within k hops of u, $\Delta{=}|\nu|{\leq}\delta^{k+1}$, which implies $k{\geq}\log_\delta(\Delta) - 1$. □

A. Algorithm Description

LDS consists of three phases: organizing the nodes in *G=(V,E)* into distinct sets, constructing a backbone that connects all sets, and assembling the remaining nodes (that are not on the backbone) in each set into balanced binary trees which are then linked to the backbone.

1) Phase 1: Creating Groupings on G: A maximal subset M of V is found such that every node in the set $(V-M)$ lies within a distance $1/2$ from some node in M, but no two nodes in M lie within distance $1/2$ from each other. Note that this construction ensures that *all* nodes that share a neighbor in M are within a distance of 1 from each other (i.e., form a clique). LDS determines M by the maximal independent set algorithm proposed in [26]. Nodes in M are called *dominators*; nodes in the set $(V - M)$ are called *member* nodes. Each member node will arbitrarily choose a unique node in M that is within a distance of 1/2 from itself, as its *dominator*. The set of member nodes that choose a particular dominator w are said to belong to w's group, *group(w)*. Any pair of nodes within a group has an edge in $G = (V, E)$, and the groups are *disjoint*.

2) Phase 2: Construction of the Higher-Tier backbone: In this phase LDS constructs a backbone that interconnects separate groups. Two groups are said to be *connected* if there is at least one edge between them (between any two members that belong to the distinct groups) in $G = (V, E)$. If there is only one such edge, it is designated to be a *gateway link*. If there are multiple edges, *one* of them is arbitrarily chosen to be a *gateway link*. Nodes that are the end points of gateway links are marked as *backbone nodes*. In addition, LDS also marks all dominators as backbone nodes. At the end of the process, in any group with more than one node, if the dominator is the only backbone node selected, an arbitrary additional node s in the group is chosen and marked as a backbone node (to aid the proof of Lemma 4).

LDS constructs a backbone $H{=}(V_H, E_H)$, where V_H consists of the designated backbone nodes and the set E_H is formed as described next.

LDS considers all edges (from G) between vertices in V_H, in the order of nondecreasing length. At each step, the considered edge $e = (u, v)$ is added to E_H unless there is an edge f in E_H that is incident on u or v, making an angle of less than 52° with e. (If f was inserted before e, this implies that $d(f) \leq d(e)$, where $d(e)$ represents the Euclidean length[4] of the edge e.) The algorithm terminates when all edges between the nodes in V_H have been thus considered. The resulting graph $H(V_H, E_H)$ is the desired backbone that connects all distinct groups.

3) Phase 3: Connecting the remaining nodes to the Higher-tier backbone: In this phase LDS finalizes the construction of the connected topology $G' = (V, E_{G'})$. The process executes in two steps. First, in every group that contains member nodes that do not lie on the backbone, LDS constructs a rooted, balanced binary tree $T(w)$ to connect these nodes. This construction is possible, as every group is a clique (as discussed earlier).

Second, LDS links these trees to the backbone as follows. Initially, $E_{G'}$ contains the "backbone edges" E_H as computed above. For each dominator w, if tree $T(w)$ exists (i.e., is not empty), then LDS removes an arbitrarily selected backbone edge (w, v) from $E_{G'}$ (Lemma 4 shows that this

[3]Two neighbors v,z of node u are *independent*, if they *cannot* be connected by a direct edge.

[4]Throughout the paper we use the notation *d(e)* for the Euclidean length of an edge *e* and *D(P)* for the Euclidean length of a path *P*.

edge has to exist); and replaces it with the two edges (w, r) and (r, v), where r is the root of $T(w)$. The edges belonging to $T(w)$ are also added to $E_{G'}$. Note that this process preserves the degrees of w and v.

B. Analysis of the algorithm

In the following, we validate the properties of LDS.

Lemma 3. Let e=(u, v),f=(u, w) be two edges incident on node u, and let g=(v, w) be the edge across the angle $\angle(euf)$. If $d(f)$<$d(e)$ and $\angle(euf)\leq 52°$, then $d(g)$<$d(e)$.

Proof. Assume $d(g)>d(e)$. Then $\angle(euf)$ is the largest angle in $\triangle(uvw)$, which implies $\angle(euf)\geq 60°$. However $\angle(euf)\leq 52°$, which contradicts the assumption; therefore $d(g)<d(e)$. □

Lemma 4. In $H(V_H, E_H)$, each dominator u has an edge with at least another backbone node (to be removed in Phase 3), if the number of nodes in *group(u)* is greater than 1. In particular, edge $(u, v)\in E_{G'}$, where v is the backbone node closest to u.

Proof. Note that at the beginning of Phase 3, $E_{G'}\supseteq E_H$; thus, it is sufficient to show $(u, v)\in E_H$. We prove the lemma by contradiction. First, note that *group(u)* contains at least two nodes in V_H by construction. (Phase 2 ensures that in each group, if the number of member nodes - excluding the dominator- is greater than zero, the dominator is *not* the only backbone node in this group.) Assume that $v \in V_H$ is the closest backbone node to dominator u[5], and that edge $(u, v) \notin E_H$. Then, by construction, there exists either an edge (u, w) or an edge (v, z) in E_H, that *blocks*[6] (u, v) (w,$z \in V_H$). The blocking edge cannot be of the form (u, w), because this would imply $d(u, w) \leq d(u, v)$ which contradicts the assumption. Therefore, it must be that $(v, z) \in E_H$, blocks (u, v). Then $d(v, z)\leq d(u, v)$ and $\angle(v, z)v(v, u) < 52°$. By Lemma 3, $d(u, z)<d(u, v)$, which contradicts the assumption. □

Theorem 1. The maximum node degree in G' is 6.

Proof. It suffices to show that $H(V_H, E_H)$ has a maximum degree of 6; this is because Phase 3 of the algorithm ensures that in each group:

- *(i)* member nodes that are not backbone nodes are assembled into balanced binary trees, such that the tree has a maximum node degree of 3,
- *(ii)* the degree of the root increases by 2 in connecting to the backbone, but becomes at most 4,
- *(iii)* the degree of any node in H is preserved in constructing G' via appending the balanced binary trees.

Then, this theorem holds due to the following Lemma. □

[5]In case of ties, the backbone node with smaller ID is deemed closer.

[6]We say that an edge e is "blocked" by an adjacent edge f, *iff* $d(f) \leq d(e)$ and $\angle(evf) < 52°$.

Lemma 5. The degree of any node in H is at most 6.

Proof. Assume $\exists$ a node u in V_H with degree $deg(u)>6$. In this case, two of the edges that are incident on u must create an angle of at most $(360/7)<52°$. However, by construction, no two edges in H make an angle of less than $52°$. □

Theorem 2. The backbone H=(V_H, E_H) is a (Euclidean) 9-spanner; for each pair of backbone nodes u, v with $(u, v) \in E$, $D_H(u, v) \leq 9 \cdot d(u, v)$, where $D_H(u, v)$ is the Euclidean length of the path between u and v in H.

Proof. The proof is by induction on the edges between the backbone nodes in G, in the order they are considered in constructing E_H. Consider one such edge e=(u, v). If $(u, v)\in E_H$, then the theorem holds. If not, then $\exists$ an edge f=(u, w) $\in E_H$ such that $d(u, w)\leq d(u, v)$ and $\angle(euf) < 52°$. By Lemma 3 $d(v, w)<d(u, v)$ and thus, (v, w) was considered by LDS prior to (u, v).

Consider path $P(v, u)$ formed by $P'(v, w)$ followed by edge (w, u). For induction, we postulate that, $\exists$ a path $P'(v, w)$ in E_H such that $d(P')\leq 9{\cdot}d(v, w)$. We use this to show $d_H(u, v)\leq 9{\cdot}d(u, v)$. Then, $d(P)\leq 9{\cdot}d(v, w)+d(w, u)$. With this, it is enough to show:

$$d(u, w) + 9 \cdot d(v, w) \leq 9 \cdot d(u, v). \tag{1}$$

Or, equivalently,
$$\frac{d(u, w)}{d(u, v) - d(v, w)} \leq 9. \tag{2}$$

Let w' be a point on (u, v) s.t. $d(v, w')$=$d(v, w)$, so the triangle $\triangle(vww')$ is isosceles with equal angles at w and w'. Then $d(u, w')$=$d(u, v)$-$d(v, w)$, and inequality(2) becomes:

$$\frac{d(u, w)}{d(u, w')} \leq 9. \tag{3}$$

For any fixed $d(u, w)$ and a fixed angle $\angle(euf)$ at u, $d(u, w')$ is minimized when $d(u, v)$ is minimized. The smallest value $d(u, v)$ can take is $d(u, w)$, since inequality $d(u, w) \leq d(u, v)$ has to be satisfied. Then the value of the left hand side of inequality(3) is maximum when the denominator $d(u, w')$ is minimized, i.e., when $d(u, w)$=$d(u, v)$. The triangle $\triangle(uvw)$ is isosceles with equal angles at v and w. Let z be the midpoint of (v, w). By considering the right triangle $\triangle(uvz)$ we compute $d(v, z)$ and thereby $d(v, w)$ as:

$$d(v, w) = 2 \cdot d(v, z) = 2 \cdot d(u, v) \cdot \sin(\frac{\angle(euf)}{2}). \tag{4}$$

$$\begin{aligned}\text{Thus, } d(u, w) + 9 \cdot d(v, w) &= d(u, v) + 9 \cdot d(v, w) \\ &\leq d(u, v) + 18 \cdot d(u, v) \cdot \sin(\frac{\angle(euf)}{2}) \\ &\leq d(u, v) \cdot \left(1 + 18 \cdot \sin(\frac{\angle(euf)}{2})\right) \\ &\leq 9 \cdot d(u, v) \end{aligned} \tag{5}$$

The last step in inequality(5) is due to $\angle(euf)<52^\circ$. □

Remark 1. The factor 9 can be improved to 8 by considering angles of $51.5 > 360/7$ degrees above instead of 52°.

Lemma 6. The number of groups connected to any given group g is $O(1)$.

Proof. In Phase 1, dominators are chosen such that they are at least $1/2$ units apart. Draw disks of radius $1/4$ centered at each dominator whose groups are connected to g. These disks are disjoint and all lie within a circle of radius 2 centered at g's dominator. The number of such disks is $O(1)$. □

Corollary 1. The number of backbone nodes in each group is $O(1)$. Therefore, the number of backbone nodes that are reachable from any backbone node is $O(1)$.

Theorem 3. If any two nodes u, v in $G(V, E)$ are connected by a path P in G, then in $G'(V, E_{G'})$, a path P' such that $\text{hops}_{G'}(P') = O\left(\text{hops}_G(P) + \log \Delta\right)$ connects them. (Recall that Δ is the maximum node degree in $G(V, E)$.)

Proof. Note that each group forms a clique in G; then, each group has at most $\Delta+1$ nodes. Consider the path P from u to v in G. Let $g_1, g_2, g_3, \ldots, g_{k+1}$ be the sequence of groups that P traverses. P has at least k edges: for each $i = 1..k$, there is an edge from g_i to g_{i+1} in G. By construction, there is such an edge also in E_H for each $i = 1..k$. Let a_i and b_i be backbone nodes in g_i, such that a_i connects g_i to g_{i-1} and b_i connects g_i to g_{i+1}. This implies $d(b_{i-1}, a_i) \leq 1$, $d(a_i, b_i) \leq 1$ and $d(b_i, a_{i+1}) \leq 1$ in G. By Theorem 2 there is a path p_i from b_i to a_{i+1} and a path q_i from a_i to b_i in the backbone graph $H(V_H, E_H)$, such that $D(p_i)$ and $D(q_i)$ are each ≤ 9. Construct P' from u to v in G' as follows. From u, traverse the edges in the balanced binary tree of u's group (g_1, w.l.o.g.) to get to the root. From the root get to a_1. Then traverse paths $p_1, q_2, p_2, q_3, \ldots, p_k, q_{k+1}$ to get from a_1 to b_{k+1}. From b_{k+1} get to the root of the balanced binary tree in v's group (g_{k+1}); then traverse the tree to get from b_{k+1} to v. Traversal in each binary tree requires at most $\log_2 \Delta$ edges. As each of the paths p_i and q_i has Euclidean length at most 9, they contain nodes from $O(1)$ groups (as they pass through $O(1)$ groups). By Corollary 1 each such path has $O(1)$ edges (connecting at most all the backbone nodes in all visited groups). The number of these paths is $2k$. Therefore the hop count of path P' is $O(k + \log \Delta)$. □

Corollary 2. If two nodes u, v in $G(V, E)$ are connected by a path P in G, then in $G'(V, E_{G'})$ they are connected by a path P' such that $D_{G'}(P')=O\left(D_G(P) + \log \Delta\right)$.

Proof. The proof follows from Theorem 3 and from fact that each hop is at most of unit length. □

Lemma 7. The time complexity of LDS is $O(n \log n)$.

Proof. A maximal independent set can be constructed in $O(n)$ time [22]. By Corollary 1, the number of edges on the backbone is $O(n)$ (the number of groups being $O(n)$ in the worst case). Sorting these edges takes $O(n \log n)$ time. Constructing each binary balanced tree takes $O(\Delta \log \Delta)$ time (requires sorting the nodes in a group w.r.t. their ID's). Therefore, the time-complexity of LDS is $O(n \log n)$. □

IV. Distributed Low-Degree Spanner (D-LDS)

Next, we present our distributed algorithm Distributed LDS (D-LDS). With D-LDS, nodes make independent decisions on the links that they maintain, based only on local information with regards to their two-hop neighborhoods. D-LDS works on arbitrary networks, does not need synchronization among nodes, is scalable, and converges quickly to a connected topology with the desired properties.

D-LDS consists of four phases which are described next.

1) Phase 1: Finding the dominators and forming groups: D-LDS identifies independent dominators to form separate groups. For the dominator selection, we use the distributed algorithm for finding a Maximal Independent Set (MIS) from [27]. In brief the algorithm is as follows (further details may be found in [27]): Initially every node is colored white. A node selected as per some tie-breaking criterion (e.g. node that has the lowest ID in its 1-hop neighborhood) becomes black and declares itself a *dominator*. It broadcasts a message within its 1-hop neighborhood, announcing its transition. All of its 1-hop neighbors that receive the message become gray (if they were white). Those that have turned gray declare themselves to be *dominated* by this dominator. The algorithm terminates when there are no more white nodes left. Both time and message complexity of this construction are $O(n)$. This algorithm ensures that each node has one dominator, since a node changes color only once. Upon termination of the MIS construction, each dominator and its associated gray nodes constitute a *group*.

In order to guarantee connectivity in the later steps of D-LDS, all nodes in the same group must be able to reach each other. Hence, in constructing the MIS and in identifying the groups, nodes will transmit with a power such that the range is half the maximum possible range. As with many topology control methods in literature (designed to achieve energy efficiency), we assume that nodes are capable of adjusting their power levels[7].

2) Phase 2: Identification of the gateway nodes: Each dominator w maintains a list, $S(w)$, of the backbone nodes in *group(w)*; $S(w)$ initially includes $\{w\}$. By exchanging small messages, a dominator obtains information as

[7]If directional antennas are used, we assume that the broadcasts are achieved using circular transmissions as in [13]; the power level is adjusted such that the broadcast reaches only those neighbors that are within half of the node's directional range.

to whether any of its *members* have neighbors from other groups. If there is a node (u) in *group(w)* such that u has a neighbor v from another group (say *group(z)*), then *group(z)* is "reachable" from *group(w)*. Let $Q(w)$ denote the set of groups that are reachable from *group(w)*; this set is initially empty. At each instance when w detects a reachable group (e.g. *group(z)*), it checks whether *group(w)* is already connected to this group. If it is, then *group(z)*$\in Q(w)$. Otherwise w or z nominates gateway nodes for this pair of groups. Towards this, w checks whether its ID is greater than the ID of z. If so, it declares u and v to be the gateway nodes between *group(w)* and *group(z)* (if not, z is responsible for gateway selection). Next, w adds u to $S(w)$ and *group(z)* to $Q(w)$; it also sends a message to z to trigger the addition of *group(w)* to $Q(z)$ and v to $S(z)$.

The ID comparisons in the above discussion ensure that: *i)* for every pair of groups, exactly two gateways are identified, and *ii)* gateway identification is performed by only one of the dominators. With this approach, no synchronization is necessary; equally importantly, there are no conflicts with regards to the selection of the gateway nodes.

Before proceeding to the Phase 3, w checks if it is the *only* nominated backbone node for its group (i.e., whether $S(w) = \{w\}$). If this is the case and there are additional nodes in *group(w)*, w *arbitrarily* chooses a node s from among these nodes and marks it as a backbone node; it includes s in $S(w)$. This check ensures that, if (*group(w)*-$S(w)$) contains nodes that will be connected to the backbone in Phase 4 (to be described), then there is at least one backbone link such that both end nodes are in *group(w)*.

3) Phase 3: Construction of the Backbone: The backbone is constructed to span all dominators and gateway nodes (i.e. all *backbone nodes*); each backbone node participates in this construction in a completely local, decentralized manner. The decisions regarding each communication link is made on the basis of the computed distance to the one hop backbone neighbors. We define the distance estimation function that facilitates this computation:

Definition (Distance Function): The distance function $\delta : P \rightarrow D$ maps power values to distance values as follows: $\hat{d}(u,v)=\delta(P_r(u,v))$, where $P_r(u,v)$ is the received power of a packet from u to v or vice versa (as the channel is assumed symmetric). We assume that the δ function strictly decreases with received power; this is typically true for channels wherein only path loss is experienced. Thus, given two neighbors v and z of u, δ executed at u determines v to be *closer* than z, if the power levels of packets[8] received from v are greater than the power levels of packets received from z. Ties are broken based on ID numbers; the node with the higher ID is deemed closer.

Each backbone node u maintains a list, $A(u)$, of its 1-hop neighbors. u considers each node v in $A(u)$ and examines whether the link (u,v) is "feasible".

Definition (Feasibility of a Link): Let Y denote the set of nodes that are closer to node u than v. The nodes in Y reside *inside* a circle centered at u with radius=$\hat{d}(u,v)$. Similarly, let Z denote the set of nodes that are closer to v than u. (All links are unprocessed and unmarked, initially.)

1. A link (u,v) is *feasible*, if **both** of the following two conditions are satisfied:
   (i) Every link (u,y_i) ($y_i \in Y$) that would make an angle of less than 52° with link (u,v) must be marked as "unfeasible".
   (ii) Every link (v,z_i) ($z_i \in Z$) that would make an angle of less than 52° with link (v,u) must be marked as "unfeasible".

   A communication link that is *feasible* is created.
2. On a similar note, for a link (u,v) to be *unfeasible*, it is necessary and sufficient that **either** of the following two conditions is satisfied:
   (i) $\exists$ a link (u,y_i) ($y_i \in Y$), making an angle of less than 52° with (u,v) ((u,y_i) has been deemed *feasible* before).
   (ii) $\exists$ a link (v,z_i) ($z_i \in Z$), making an angle of less than 52° with (u,v) ((v,z_i) has been deemed *feasible* before).

   If link (u,v) is *unfeasible*, both endpoints keep this information and (u,v) is not created.

During this construction phase, each backbone node broadcasts a message (to its 2-hop neighborhood) upon deciding the status of each of its incident links. The output graph at the end of this phase is the backbone.

4) Phase 4: Finalizing the construction of the connected topology: In each group, the nodes that are not backbone nodes (if any) will form a balanced binary tree. Let the set of such nodes in *group(w)* be $R(w)$. The tree construction is triggered by the dominator (w) of the group and is performed concurrently at every node in $R(w)$. A simple scheme for building a balanced binary tree proceeds as every node in $R(w)$ performs the following: *(i)* Sorts the nodes in $R(w)$ as per their IDs, in increasing order. (The sorted array is unique and is *the same* at all nodes in $R(w)$, due to 2-hop neighborhood information at all nodes.) *(ii)* Checks its index (w.l.o.g. k) in the sorted order, and connects to nodes at indices $2k$ and $2k+1$ (if $2k$ and $2k+1$ do not exceed the number of elements in $R(w)$). We note that no message exchanges are necessary for this construction, since the IDs are known and the tree is unique for a given set $R(w)$.

Let r be the root of the tree constructed in *group(w)*. Next, r will attach itself to the backbone, as w=*dominator(r)* removes *any* link (w,j) along the backbone and forms links

[8]To prevent the distance information from becoming stale, periodic updates will be necessary. We assume that the HELLO messages facilitate this process.

(w, r) and (r, j). This procedure requires two unicast messages: from w to j for the removal of the link (w, j), and from w to r, to trigger the formation of link (r, j). Phase 4 terminates after this construction is complete for all groups.

Theorem 4. Nodes in the topology constructed by D-LDS have a maximum degree of 6.

Proof. The proof follows from that of Theorem 1 as D-LDS emulates LDS in all phases. □

Theorem 5. Let $G'(V, E_{G'})$ be the topology constructed by D-LDS. For each pair of nodes u, v that had a path connecting them in G, $\exists$ a path connecting them in G', s.t. $\text{hops}_{G'}(u, v) \leq O(\text{hops}_G(u, v) + \log \Delta)$.

Proof. The proof follows from that of Theorem 3. □

Corollary 3. For each pair of nodes u, v that had a path connecting them in G, there is a path P' connecting them in G', such that $D_{G'}(P') = O(D_G(P) + \log \Delta)$, where $D_G(P)$ denotes the Euclidian length of path P.

Proof. The proof follows from Theorem 4 and Corollary 2. □

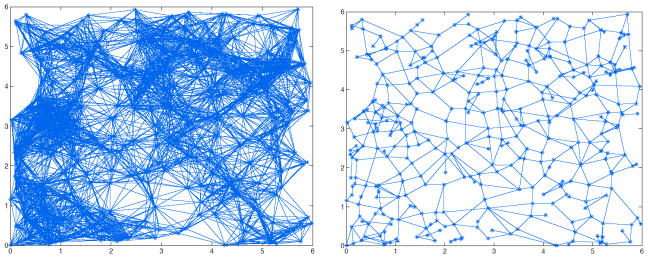
Theorem 6. For any input graph G, D-LDS constructs the final topology G' using $O(n)$ messages (in the worst case).

Proof. Two-hop neighborhood information can be provided at all nodes with a total message complexity of $O(n)$ ([7]). Message complexity of finding an MIS is $O(n)$ ([27]). In the worst case, the number of backbone links is $O(n)$ (Lemma 7). As each backbone node sends only one message upon deciding the status of a link, the total number of messages generated is $O(n)$. Finally, the balanced binary tree construction does not require message exchanges (two-hop neighborhood information is already exchanged). For appending the trees to the backbone, $O(n)$ messages are necessary in the worst case (2 messages are exchanged per group; number of groups is $O(n)$ in the worst case). Therefore the overall message complexity of D-LDS is $O(n)$. □

V. Simulation Experiments

We study the performance of our distributed algorithm via extensive simulations. We have implemented D-LDS in a C++ simulator. For accessing the channel, all the nodes use CSMA which has been popularly considered for broadcasting packets in ad hoc networks. Our program takes as input a unit disk graph, where a unit is the range achieved with the maximum transmission power; it outputs a hop-spanner having degree at most six. Figure 1 illustrates the input and the output graphs: Figure 1(a) is a UDG constructed by randomly placing 360 nodes in a 6x6 unit area, and Figure 1(b) is the topology output by D-LDS.

Metrics of Interest: The metrics of our interest are as follows:



(a) The Unit Disk Graph (b) The Graph Generated by D-LDS

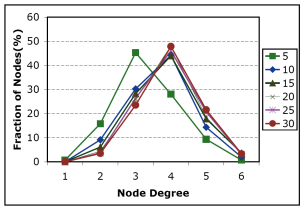
Fig. 1. Visualization of the Unit Disk Graph and the Graph Generated by D-LDS. (360 nodes are generated in a 6 by 6 topology.)

1) **Node degree distribution.** For every degree possible, we measure the fraction (in percentage) of the nodes in the output graph that have this degree.
2) **Average hop stretch.** For every edge in the UDG, we measure the shortest path in terms of hops between its nodes in the graph constructed by D-LDS. The length of this shortest path is defined to be the hop stretch for the particular UDG edge. The average is simply computed over all UDG edges.
3) **95 percentile hop stretch.** We define it as the 95% tail[9] of the set of the hop stretch values of all the edges in the considered UDG.
4) **Number of messages.** The number of messages required by D-LDS to construct the final topology. Note that the construction of the MIS and the acquisition of the two-hop neighborhood are well studied problems [22],[7]. Therefore, we do not include the messages required during these phases in our assessments.

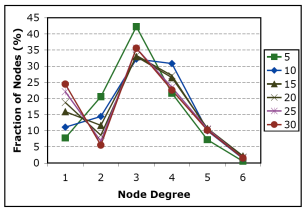
Simulation Setup. Our objective is to show that the performance of D-LDS scales when moderate to large networks are considered. The simulations are performed with a multiplicity of topologies that are placed in an area of interest; the area is varied from 4x4 to 10x10 units. We also consider different node densities; we vary the number of nodes from 5 to 30 per square unit. Every unit is 250m, which is the coverage range for a wireless card complying with the IEEE 802.11 standard. For every topology we generate nodes randomly, using a uniform distribution. We remark that while our simulations are performed using the specific considered distribution, the bounds that were derived previously hold for arbitrary unit disk graphs.

We first consider the node degree distribution. We perform experiments in a 6x6 unit area *and* with different node densities. Figure 2(a) shows the degree distribution for the backbone nodes, since the balanced binary tree nodes have well defined degrees (half of them have degree 3 and the other half -the leaves- have degree 1.) As shown in Figure

[9] x% tail of a set of values, is the value that is bigger than x% of the values in that set.



(a) Node Degree Distribution for the Backbone



(b) Node Degree Distribution for Global Topology

Fig. 2. Distribution of the Node Degree for the Topology Generated by D-LDS

2(a), almost half of the nodes have degrees of at most 4, for node densities of 10 and above. (When the node density is 5, the UDG is quite sparse; this explains the lower node degrees as compared to the other densities.) It is also important to note that not more then 4% of the nodes in the network reach the worst case bound of 6. To have a complete picture of the node degree distribution of the final topology, we also plot the results including the tree nodes. As expected, a significant number of nodes have degrees less than 3.

The promising results on the node degree acquire more importance given that (as we plot in Figure 3(a)) the graph constructed by D-LDS, for all considered networks of high density, is a hop-spanner with the worst case 95 percentile stretch being 12. In the same figure we also plotted the average path stretch factor; this is close to 6 for all considered network sizes. In addition, we measured the average and the 95 percentile hop stretch as a function of node density while the area size is fixed to 6x6 units (Figure 3(b)). We see that the path stretch factor is even lower for more moderate node densities that are more likely in ad hoc network deployments.

Another important metric of performance for a topology control algorithm is the number of messages that are exchanged prior to convergence to the global topology with the desired properties. We showed in Section IV that the number of messages required by D-LDS is $O(n)$ i.e., = cn, where c is a constant. The simulations, however, can provide an estimate of the hidden constant c. Towards this, we generate enough nodes such that in every square unit approximately 30 nodes are present; this is considered a very high density for an ad hoc network. Given this density, the network construction is varied from a 4x4 unit area with 480 nodes (a moderately sized ad hoc network), to a 10x10 unit area with 3000 nodes (a large ad hoc network). We measured the total number of messages required by D-LDS to construct the final topology, and divided it by the number of nodes. The results are depicted in Figure 3(c) and they show that the measured estimate of the constant stabilizes to between 18 and 19 and does not change even when the number of nodes is as high as 3000.

Convergence of D-LDS: Next we examine the convergence properties of D-LDS.

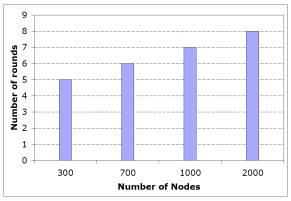


Fig. 4. Time Convergence for D-LDS.

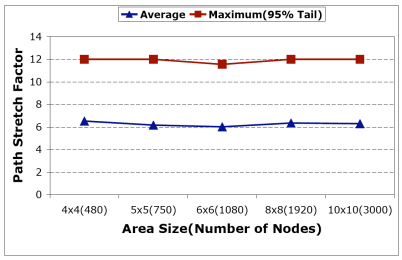
Convergence to a connected sub-graph: The unreliable wireless channel that is considered can result in packet losses at the link layer. This can, in turn, delay the establishment of edges with D-LDS. We are interested in investigating how the connectivity of the network is affected by this delay in edge establishment. In other words, we study the convergence of D-LDS in terms of forming the final graph. To understand this, we perform the following experiment. 1000 nodes are placed in a 6 by 6 unit area and we trace the connectivity of the backbone at different stages of D-LDS execution; at each stage, note that only a percentage of the final set of edges is established. We investigate the impact on the backbone connectivity only; the binary balanced trees are constructed by nodes that are within the one hop reach of each other, and hence, do not affect the connectivity of the other nodes beyond their group.

The results of the experiment are shown in Table I. Note that D-LDS can provide high connectivity even when just a subset of edges are established. Full connectivity is practically provided even if only 70% of the edges are established.

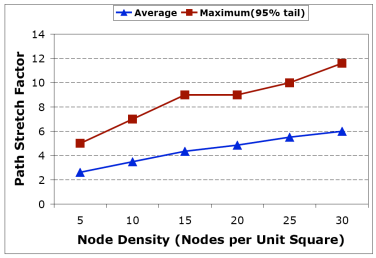
Convergence in time: Finally, we examine the time it takes for the backbone to converge (after the groups are identified). This provides us with an idea of the parallelism posible with D-LDS. We define convergence time in terms of "the number of rounds", where a round corresponds to the the duration of transmission of a single message. Since with D-LDS, edges can be inserted in parallel, we expect the time taken for convergence to be much smaller than the worst case $O(n)$ bound. To corroborate this expectation, we construct topologies of varying density in a 6 x 6 unit area. We depict the results in Figure 4. We observe that, even with a large number of nodes (as high as 2000), only 8 rounds are needed for convergence. This demonstrates the high degree of parallelism that is possible with D-LDS.

Established Edges (%)	Connectivity (%)
50	62.9
60	82.2
70	96
80	99

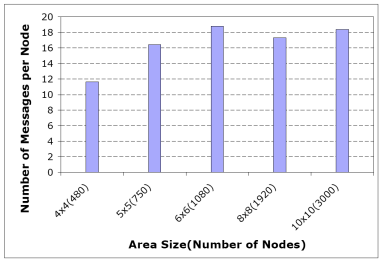
TABLE I
CONNECTIVITY AT DIFFERENT STAGES OF THE D-LDS EXECUTION.



(a) Hop Stretch for High Density(30 nodes/ square unit)



(b) Hop Stretch for Various Densities



(c) Total Number of Messages Divided by the Number of Nodes

Fig. 3. Hop Stretch and Average Message Complexity of D-LDS for Various Topologies.

	Max. Node Degree	Short Links (Low Transm. Power)	Hop-Stretch	Message Complexity (Comm. Cost)
[4]	Not bounded	Yes	Not bounded	2n
[11]	Not bounded	Yes	Constant C, $C > 0$	O(n)
[15]	6	Yes	Not bounded	No claims
[16]	Not bounded	Yes	Not bounded	O(nlogn)
[21]	6	Yes	Not bounded	13n
[24]	*(i)* 12 ; *(ii)* 9	Yes. (Longer links in *(ii)*)	Not bounded	*(i)* 24n ; *(ii)* 3n
[17]	8	Yes	Not bounded	13n
[29]	6	Yes	Not bounded	No claims
D-LDS	**6**	Yes	$\boldsymbol{O(\text{hops}_{UDG}(u,v) + \log(\Delta))}$	O(n)

TABLE II

COMPARISON OF TOPOLOGY CONTROL ALGORITHMS IN TERMS OF KEY FEATURES OF D-LDS.

VI. RELATED WORK

In this section we present an overview of the previous topology control approaches.

Euclidean Spanners: Most previous work on the construction of sparse *and/or* low-weight spanners target reducing energy consumption. A greedy centralized algorithm is proposed in [12] for constructing a sparse, low-weighted spanner G' of an input graph G. Despite the attractive properties of G', the algorithm is inapplicable to ad hoc networks since it assumes that G is a complete graph (i.e., all pairs of nodes have direct links in between), and it requires global topology information. There is recent work that offers distributed solutions to bounding the Euclidean length of paths. In [16], the authors present a distributed algorithm that constructs a 2.5-spanner of a UDG. However, the proposed methods do not guarantee a bound on the node degree.

Topology Control Algorithms that Bound the Node Degree: There has been prior work on constructing bounded-degree subgraphs [15][19] [29][4][21]. The best theoretical bound on node degree so far is 6 and is achievable by the methods proposed in [15], [21], [29]. Computational geometric constructions such as Delaunay Triangulation (DT) [16], [11], [1] and the Relative Neighborhood Graph (RNG)[21] have been considered for *localized* topology control. The properties of these structures allow imposing bounds on energy-consumption or maximum node degree [10]. The proposed algorithms however, do not address hop stretch in terms of hop-count.

Algorithms that Construct Euclidean Spanners with Bounded Node Degree: There are prior efforts on designing Euclidean spanners to achieve sparseness, low weight and bounded degree [3], [2], [10], [23]. However, the proposed algorithms were not designed for UDGs; furthermore, it is assumed that global topology information is available at the nodes. Such limitations render these algorithms inappropriate for ad hoc networks. More recently, there have been efforts to control the energy consumption in a given wireless network. The centralized algorithm proposed in [5] constructs a t-spanner of the *complete* input graph, with $t \approx 10.02$; however, the degree bound achieved is 27. The centralized algorithm proposed in [20] achieves a degree bound of $19 + \lceil \frac{2\pi}{\alpha} \rceil$, where $\alpha \in (0, \pi/2)$. A distributed version of this algorithm is presented in [28]; the imposed bounds are the same. In [19], the properties of the proximity structures, the Yao Graph (YG) and the Gabriel Graph (GG), are exploited to provide a power-efficient spanner with bounded degree; however the bounds achieved for the in-degree and the out-degree are 63 and 7, respectively. The algorithm proposed in [24] constructs a Euclidean spanner; however, the tightest degree bound achieved is strictly higher than 8. This degree bound has been improved to 8 by Li et. al in [17]; the proposed algorithm constructs energy-efficient topologies for both unicast and broadcast communications.

Topology Control for Bounded Hop Stretch: In [1], Alzoubi et.al. propose a localized algorithm that builds a sub-

graph with a hop stretch of 3, but no degree bound is imposed. (The algorithm provides a degree bound on a backbone constructed on the input graph G; however, the bounds are very large: 295 for the *dominator*s and 7384 for the *connector*s, and are *not* imposed on the final topology.). In [11], Gao et al. propose a distributed algorithm that constructs a spanner in terms of both Euclidean and topological distances. In [6], Burkhart et al. propose an algorithm which, *given* a parameter t, can construct a minimum-interference Euclidean t-spanner of the UDG. (The authors indicate that their results are extendable to hop spanners with slight modifications.). However both [6] and [11] do not study bounding the node degree.

In summary, despite the existence of previous work that optimize the Euclidean path stretch and the node degree, there are no algorithms, to the best of our knowledge, that impose bounds on **both** the node degree and the *hop stretch*. Furthermore, our work distinguishes itself from the previous efforts by achieving the most attractive bounds to date on these two metrics. In Table VI, we list the distributed, locally-maintained topology control algorithms to provide a compact view of what each scheme achieves; the table also shows that D-LDS compares favorably with these solutions.

VII. Conclusions

In this paper, we propose algorithms (a centralized and a distributed version) for the construction of a bounded-degree spanner with a low hop stretch for ad hoc networks. These features are highly desirable especially with specialized physical layer capabilities such as directional antennas or MIMO. Our approach leads to the construction of a spanner with a maximum node degree of 6. In addition, our approach offers bounded path stretch in terms of **hop count**. If the original path length in terms of hop count is P, the new path length is $O(\text{hops}(P) + \log \Delta)$ (where, Δ is the maximum degree of a node in the input graph) in the worst case. We extensively simulate our distributed algorithm D-LDS. Our simulations demonstrate that D-LDS constructs topologies with extremely low average path stretch in typical ad hoc network deployments. The induced node degree is also typically much smaller than the theoretical bound 6 in typical deployments.

References

[1] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. In *IEEE TPDS*, May 2003.

[2] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *ACM STOC'95*.

[3] S. Arya and M. H. M. Smid. Efficient construction of a bounded degree spanner with low weight. In *ESA '94: Proceedings of the Second Annual European Symposium on Algorithms*.

[4] D. M. Blough, M. Leoncini, G. Resta, and P. Santi. The k-neigh protocol for symmetric topology control in ad hoc networks. In *ACM MobiHoc'03*.

[5] P. Bose, J. Gudmundsson, and M. H. M. Smid. Constructing plane spanners of bounded degree and low weight. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*.

[6] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does topology control reduce interference? In *ACM MobiHoc'04*.

[7] G. Calinescu. Computing 2-hop neighborhoods in ad hoc wireless networks. In *ADHOC-NOW'03*.

[8] R. Roy Choudhury and N. H. Vaidya. Impact of directional antennas on ad hoc routing. In *IFIP PWC'03*.

[9] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86, 1990.

[10] D. Eppstein. Spanning trees and spanners, 1996. Technical Report.

[11] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *ACM MobiHoc'01*.

[12] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5), 2002.

[13] T. Korakis, G. Jakllari, and L. Tassiulas. A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks. In *ACM MobiHoc'03*.

[14] K.Sundaresan, R. Sivakumar, and M. A. Ingram. A fair medium access control protocol for ad-hoc networks with MIMO links. In *IEEE INFOCOM'04*.

[15] N. Li, J. C. Hou, and L. Sha. Design and analysis of an mst-based topology control algorithm. In *IEEE INFOCOM'03*.

[16] X. Li, G. Calinescu, and P. Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *IEEE INFOCOM'02*.

[17] X. Li, W. Song, and W. Wang. A unified energy-efficient topology for unicast and broadcast. In *ACM MOBICOM*, 2005.

[18] X. Li, I. Stojmenovic, and Y. Wang. Partial delaunay triangulation and degree limited localized bluetooth multihop scatternet formation. In *IEEE TPDS*, volume 15, 2004.

[19] X. Li, P. Wan, Y. Wang, and O. Frieder. Sparse power efficient topology for wireless networks. In *HICSS'02*.

[20] X. Li and Y. Wang. Efficient construction of low weight bounded degree planar spanner. In *COCOON'03*.

[21] X. Li, Y. Wang, P. Wan, and O. Frieder. Localized low weight graph and its applications in wireless ad hoc networks. In *IEEE INFOCOM'04*.

[22] M. Min, F. Wang, D. Du, and P. M. Pardalos. A reliable virtual backbone scheme in mobile ad-hoc networks. In *IEEE MASS'04*.

[23] J. S. Salowe. Euclidean spanner graphs with degree four. *Discrete Applied Mathematics*, 54(1), 1994.

[24] W. Song, Y. Wang, and X. Li. Localized algorithms for energy efficient topology in wireless ad hoc networks. In *ACM MobiHoc'04*.

[25] M. Takai, J. Martin, R. Bagrodia, and A. Ren. Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In *ACM MobiHoc'02*.

[26] R.L.Rivest T.H.Cormen, C.E.Leiserson and C.Stein. In *Introduction to Algorithms*, 2001. The MIT Press.

[27] P. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *IEEE INFOCOM'02*.

[28] Y. Wang and X. Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. In *DIALM-POMC'03*.

[29] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for wireless multihop ad hoc networks. In *IEEE INFOCOM'01*.

[30] M. Zorzi, J. Zeidler, A. Anderson, A.L. Swindlehurst, M. Jensen, S.V. Krishnamurthy, B. Rao, and J. Proakis. Cross-layer issues in MAC protocol design for MIMO ad hoc networks. In *IEEE Wireless Communications Magazine (to appear)*.