Consiglio Nazionale delle Ricerche

k –NEIGHLEV: a Practical Realization of Neighborhood-Based Topology Control in Ad Hoc Networks

D.M. Blough, M. Leoncini, G. Resta, P. Santi

IIT TR-09/2003

Technical report

Agosto 2003

Istituto di Informatica e Telematica

k-NEIGHLEV: a Practical Realization of Neighborhood-Based Topology Control in Ad Hoc Networks

Douglas M. Blough[1]
Georgia Inst. of Tech.
Atlanta, GA – USA

Mauro Leoncini
Univ. Modena e
Reggio Emilia – Italy

Giovanni Resta
IIT – CNR
Pisa – Italy

Paolo Santi[2]
IIT – CNR
Pisa – Italy

***Abstract*— Neighborhood-based topology control has been proven to be very effective in reducing energy consumption and increasing network capacity in ad hoc networks. In this paper, we present a practical realization of this approach that does not rely on distance estimation. Instead, the protocols presented in this paper leverage a feature typical of current wireless cards, namely that discrete power levels can be used for transmission. Ours are the first discrete-power-level protocols that do not require changing the power level on a per-packet basis. We demonstrate, through simulation, that the excellent performance of neighborhood-based topology control is maintained in this more practical setting. We also show that significant energy savings can be obtained if the power levels are optimized for topology control, rather than chosen in an ad hoc manner. Finally, we extend our approach to provide a neighborhood-based topology control protocol that is suitable for mobile networks.**

***Index Terms*—system design, simulations, graph theory.**

I. INTRODUCTION

The topology control problem in wireless ad hoc networks is to choose the transmit power of each node in such a way that energy consumption is reduced and some property of the communication graph (typically, connectivity) is maintained. Besides reducing energy consumption, topology control increases the capacity of the network, due to reduced contention to access the wireless channel. In fact, in [6] it has been shown that it is more convenient, from the network capacity point of view, to send packets along several short hops rather than using long hops. Given the limited availability of both energy and capacity in ad hoc networks, topology control is thus considered as a major building block of forthcoming wireless networks.

Ideally, a topology control protocol should be asynchronous, fully distributed, and localized (i.e., nodes should base their decisions only on information provided by their neighbors). Furthermore, it should rely on information that does not require additional hardware on the nodes, e.g. to determine directional or location information. A final requirement of a good topology control protocol is that it generates a connected and relatively sparse communication graph. These latter features, besides reducing the expected contentions at the MAC layer, ease the task of finding routes between nodes.

Although several recent papers have addressed the problem of topology control in ad hoc networks [1], [2], [4], [7], [8], [9], [10], [11], [13], [14], [16], none of them meet all of the requirements described above.

In [2], we proved that the neighborhood-based approach to topology control has a number of desirable properties, including excellent energy efficiency relative to the best previously-known protocol. The approach consists of maintaining a symmetric version of the k-neighbors communication graph, for some value of k that guarantees connectivity with high probability. However, the k-NEIGH protocol of [2] is based on distance estimation capabilities, which require additional hardware on the nodes, thus not fulfilling one of the requirements for "ideal" topology control.

In this paper, we present a practical realization of neighborhood-based topology control, which leverages the fact that current wireless transceiver technology enables the choice of discrete transmit power levels, as in case of both IEEE 802.11 cards (see, for example, the CISCO Aironet cards [5]) and sensor nodes (e.g., the Medusa II node described in [12]). This fact is often not considered in current approaches to topology control, in which it is implicitly assumed that any value of the transmit power level can be chosen by the nodes. This observation applies, for instance, to the protocols presented in [2], [4], [13], [14]. On the contrary, in this paper we as-

[1] The work of this author was supported in part by the National Science Foundation under Grant ECS–0225417.

[2] The work of this author was funded in part by CNR-NATO under grant 215.34.

sume that nodes can only choose between a finite number of different power levels. In this respect, our approach is similar to that of [8], [11]. A qualitative performance comparison of our protocols and those of [8], [11] is reported in Section VII.

Contrary to the CLUSTERPOW protocol of [8], in this paper we assume that power level of a node is set periodically, and that in the interval between two checks the node uses the same power level on every packet. This choice it motivated by the fact that, as observed in [8], changing the transmit power level incurs a considerable cost with the current transceiver technology, both in terms of latency (on the order of 100ms) and energy consumption. Further, it is reported in [8] that frequent changes in the power level are likely to crash the wireless card. For these reasons, we believe that periodic topology control is preferable to per-packet topology control with current transceiver technology.

II. Preliminaries and Working Assumptions

The protocols presented in this paper are based on the following assumptions:

- nodes can transmit messages at different power levels, denoted $p_0, \ldots, p_{max}$, which are the same for all the nodes;
- the wireless medium is *symmetric*, i.e., if node v can receive a message sent by node u at power p_i, then u is able to receive a message sent by v using the same power p_i.

For the sake of brevity, in the following we will say that a node is *at level i* if its current transmit power is set to p_i. Also, we will let A_i denote the radio coverage area of a given node at level i, $i = 0, \ldots, max$. Note that the assumptions above only guarantee that $A_i \subseteq A_{i+1}$, without imposing any particular shape to the surface covered by a node at level i; in particular, the surface is not necessarily circular, as assumed in many papers.

Let $G=(N, E)$ be the directed graph denoting the communication links in the network, where N is the set of nodes, with $|N| = n$, and $E = \{(u, v)$: v is within u's transmitting range at the current power level$\}$ is the (directed) edge set. Clearly, as the nodes may be at different levels, $(u, v) \in E$ does not imply $(v, u) \in E$.

For every node u in the network, we define the following neighbor sets:

- the *incoming neighbor set*, denoted $N_i(u)$, where $N_i(u)=\{v \in N : (v, u) \in E\}$.
- the *outgoing neighbor set*, denoted $N_o(u)$, where $N_o(u)=\{v \in N : (u, v) \in E\}$.
- the *symmetric neighbor set*, denoted $N_s(u)$, where $N_s(u)= N_i(u) \bigcap N_o(u)=\{v \in N : (v, u) \in E$ and $(u, v) \in E\}$.

Clearly, the neighbor sets of node u change as u's level and the levels of nodes in its vicinity vary. The ultimate goal of k-NeighLev is to cause $N_s(u)$ to contain k (or slightly more than k) nodes, where k is an appropriately chosen parameter[1]. Motivations for our interest in the number of *symmetric* neighbors of a node can be found in [2].

Note that, when a node u changes its level, only the set $N_o(u)$ can vary; i.e., a node has only partial control on its set of symmetric neighbors. Furthermore, the only neighbor set that a node can directly measure is $N_i(u)$, which is not impacted by an increase in u's power level. Thus, to increase the sizes of $N_i(u)$ and $N_s(u)$, some nodes in the vicinity of u must increase their transmit powers. This fact points out a flaw in some existing neighborhood-based protocols, which do not use explicit control messages.

Consider, for example, the MobileGrid protocol of Liu and Li [10]. MobileGrid is based on a parameter called the *contention index* (*CI*). The goal of MobileGrid is to achieve an "optimal" value of *CI* at each node. For any given node u, *CI* is defined as the number of nodes within u's transmission range (including u). However, *CI* is estimated as the number of nodes whose messages can be overheard by u. Using our terminology, *CI* at node u is defined in terms of $N_o(u)$, but it is estimated in terms of $N_i(u)$. If the estimated *CI* is too low at node u, the protocol prescribes that u's transmit power be increased. This may increase the number of nodes within u's transmission range, but it definitely does not increase the estimated value of *CI* and might actually decrease it due to other nodes' responses to u's increase. Since the estimated *CI* does not increase, u will increase its power level again at the next period and it is possible that this repeats until u reaches the maximum power.

Another neighborhood-based protocol which does not require explicit control messages is the LINT/LILT protocol of Ramanathan and Rosales-Hain [13]. However, in [13], the authors assume that a symmetric set of neighboring nodes is available as a result of the underlying routing protocol, which is left unspecified. In a certain sense, the problem incurred by MobileGrid is thus overlooked.

In order to avoid the problem mentioned above, our k-neighbors protocols make use of explicit control messages.

III. Stationary Networks

The k-NeighLev protocol implements the following idea. By circulating short control messages, nodes can

[1] This requirement will be slightly weakened in the mobile version of the protocol.

let neighbors know their current power level. Based upon this information, and knowing its own level, a node can determine its symmetric neighborhood. If the number of symmetric neighbors is too low, a node can then send one or more control messages (of a different type) and trigger a power level increase in nearby nodes that are potential neighbors. This process continues until there are at least k symmetric neighbors or the node reaches the maximum power setting. In Section V, we will discuss how to set the value of the fundamental parameter k.

The protocol uses two types of control messages: *beacon* and *help* messages. Both types of messages contain the sender's ID and current power level. Beacon messages are used to inform current (outgoing) neighbors of the power level of the sender, so that their symmetric neighbor sets can be properly updated. On the contrary, help messages are used to trigger some of the receivers to increase their transmit power level, so that the symmetric neighbor count of the help sender is (possibly) increased.

Initially, all nodes set their powers to level 0, and send a beacon message. After node u has sent this initial message, it waits for a certain stabilization time T_0, during which it only performs interrupt handling routines in response to the messages received by other nodes. The main goal of these routines, which are described in detail below, is to update u's symmetric neighbor set. After time T_0, node u checks whether it has at least k symmetric neighbors. If so, it becomes *inactive*, and from this point on it participates in the protocol by simply responding (if necessary) to the control messages sent by other nodes. Otherwise, it remains *active*, and it enters the *Increase Symmetric Neighbors* (ISN) phase. During the ISN phase, node u sends help messages at increasing power levels, with the purpose of increasing the size of its symmetric neighbor set. This process is repeated until $|N_s(u)| \geq k$, or the maximum transmit power level is reached.

The routines that are executed upon the reception of control messages are as follows.

When node u receives a beacon message (v, l_v), it first checks whether $v \in N_i(u)$. If so, u has already received a control message from v, and the current beacon is simply ignored. Otherwise, u stores in a local variable $l_v(u)$ the level l_v, which represents the minimum power level needed for u to reach node v[2]. Furthermore, node u includes v in its list of incoming neighbors and, if $l_u \geq l_v$ (here, l_u denotes u's current power level), also in the list of symmetric neighbors.

When node u receives a help message (v, l_v), it checks whether this is the first control message received by v. If so, it sets the $l_v(u)$ variable and the set of incoming and symmetric neighbors as described above. Furthermore, node u compares its power level to l_v and, if $l_u < l_v$, it increases its power level to l_v, so that v's symmetric neighbor set will eventually be increased in size. As a side effect, node v is included in $N_s(u)$. In increasing its power from level l_u to l_v, node u sends a sequence of beacons, one at each power level from $l_u + 1$ to l_v. By doing so, we guarantee that when the variable $l_u(y)$ is set at node y, it actually stores the minimum power required for y to reach node u.

If the help message (v, l_v) is not the first control message from v that is received by u, then $v \in N_i(u)$, and node u knows the minimum power level needed to reach v (which is stored in the variable $l_v(u)$). Thus, node u simply checks whether $v \in N_s(u)$; if so, u is already a symmetric neighbor of v, and the help request from v is ignored. Otherwise, $l_u < l_v$, and the power level of u is increased to $l_v(u)$ (which is the minimum level needed to render u and v symmetric neighbors), using the same one by one power increase procedure as described above.

A pseudo-code description of the protocol k-NEIGHLEV is shown in Figure 1. In order to improve readability, we drop the "argument" u (which is clearly redundant at node u) from the variables $l_x(u)$, $N_s(u)$, and $N_i(u)$. Finally, we recall that when a node is at level i, all the messages are sent at power p_i.

It is easy to see that the protocol terminates in finite time. Moreover, the following theorem, whose detailed proof can be found in [3], shows that there exist values of the waiting times T_l such that the protocol correctly determines a symmetric communication graph, which has the following property: the power setting of a node is greater than the minimum necessary for it to have k symmetric neighbors only when one of its in-neighbors requires help in achieving its own symmetric neighbor requirement.

Theorem 1: The k-NEIGHLEV protocol satisfies the following properties:

(a) the total number of control messages exchanged is $O(n \cdot max)$;

moreover, there exists values T_l, $l = 0, \ldots, max$, of the waiting times such that, with high probability:

(b) at the end of the protocol execution, node $u \in N_s(v)$ if and only if node $v \in N_s(u)$;

(c) node u sends the help message at level i only if the number of nodes in A_{i-1} is smaller than k, $i < max$.

Proof: (Sketch) By simple code inspection, it is easy to see that a node (either active or inactive) sends at most one beacon and one help message per level. Hence, the total number of control messages sent is at most $2n(max + 1)$, which proves (a).

If the waiting times T_l are properly set (see [3] for de-

[2]Here, the symmetry assumption of the wireless medium is used.

- Main:
  - set $l = 0$, $N_i = N_s = \{\}$;
  - send beacon (u, l);
  - set $h = 0$; /* Remark: remember the level before sleeping */
  - wait (for a stabilization time) T_0;
  - repeat
    - if $|N_s| \geq k$ exit; /* ... and starts operating */
    - set $l = h + 1$; /* Go up one level (if no power increase has been forced by interrupt handling routines) */
    - send help message (u, l);
    - set $h = l$; /* Remark: (again) remember the level before waiting */
    - wait T_l;
  - until $l = max$
  - start node operations;
- Upon receiving a beacon message (v, λ):
  - if $v \notin N_i$
    - $l_v = \lambda$;
    - $N_i = N_i \bigcup \{v\}$;
    - if $l \geq l_v$ then $N_s = N_s \bigcup \{v\}$;
- Upon receiving a help message (v, λ):
  - if $v \in N_i$ and $v \notin N_s$ stepwise-increase($l + 1, l_v$);
  - if $v \notin N_i$
    - $l_v = \lambda$;
    - $N_i = N_i \bigcup \{v\}$;
    - if $l < l_v$ stepwise-increase($l + 1, l_v$); else $N_s = N_s \bigcup \{v\}$;
- Procedure stepwise-increase (i, f);
  - for $h = i, \ldots, f$, do:
    - set $l = h$;
    - send beacon (u, l);
    - $N_s = N_s \bigcup \{z\}$, for any $z \in N_i$ s.t. $l = l_z$;

Fig. 1. Algorithm k-NEIGHLEV performed by node u.

tails on how these times must be set), all the messages triggered by a help request sent by node u are received by u before the node checks its neighbor count again. This implies that: (1) if node v becomes symmetric neighbor of u in response to the help message, then u will include v in its symmetric neighbors set after the stabilization time, and (b) is proved; (2) denoting with n_{i-1} the number of nodes in the coverage area A_{i-1} centered at u, node u will have symmetric neighbors count at least n_{i-1} after sending the help message at power $i - 1$ and waiting for the stabilization time; thus, node u sends the help message at power i only if $n_{i-1} < k$, and (c) is proved. ■

IV. OPTIMIZATIONS: THE k-NEIGHLEVU PROTOCOL

The k-NEIGHLEV presented in the previous Section leaves room for some optimization.

A first simple optimization is the following. Suppose that there is a node u having fewer than k neighbors in its A_{max} vicinity, and such that the surface $A_{max} - A_j$ centered at u is empty, for some $j < max$. This circumstance can be easily detected: all that is required is one additional variable $b(u)$ storing the last level at which u has added to its symmetric neighbor set. When node u eventually sets its level to max, and verifies that $|N_s(u)|$ is still less than k, it can safely backtrack to power level $b(u)$.

A second and more serious opportunity for optimization is motivated by the observation that a help message in the k-NEIGHLEV protocol causes *all* the nodes that receive it to become symmetric neighbors of the sender, if they are not already so. This mechanism might be quite inefficient, forcing unnecessary power increases in the vicinity of the help sender. For example, suppose the surface A_{i-1} of node u contains $k - 1$ nodes, and that the surface $A_i - A_{i-1}$ contains $c > 1$ potential symmetric neighbors. In this case, k-NEIGHLEV would force all the nodes in $A_i - A_{i-1}$ to increase their power levels, increasing u's symmetric neighbor set size to $k + c - 1$. On the other hand, a single power increase among the nodes in $A_i - A_{i-1}$ would have been sufficient for u to meet its requirement on $N_s(u)$.

Another potential inefficiency of k-NEIGHLEV is illustrated in Figure 2. Suppose $k = 4$ and nodes u, v, and w are at levels 2, 1, and 4, respectively. Assume also that $|N_s(v)| \geq 4$ and $|N_s(w)| \geq 4$. Finally, suppose that the levels correspond to the following transmit power settings: 1mW, 5mW, 20mW, 30mW, 50mW, and 100mW (these are the power levels used in the Cisco Aironet card [5]). Now, node u has at least two choices for reaching the desired number of neighbors:

- "selfish" behavior: since $|N_s(u)| < 4$, send a help request at level 2, thus forcing node v to increase its power level;
- "unselfish" behavior: use the information stored in $N_i(u)$, which lists w, and increase the level to $l_w(u)$.

In case of selfish behavior, which corresponds to the basic protocol implementation, the overall power increase in the vicinity of u is 10mW+25mW, due to node u stepping up one level and v two levels. In case of unselfish behavior, the increase is 30mW, due to u stepping up two levels. Hence, unselfishness is preferable in this case. Note that the opposite conclusion would be drawn if the node powers in Figure 2 were all scaled up by one level. In that case, the power increases would change to 50mW (20mW+30mW) for the selfish approach and 70mW for the unselfish one.

This example, with its opposite conclusions depending on the node power levels, along with the k-NEIGHLEV

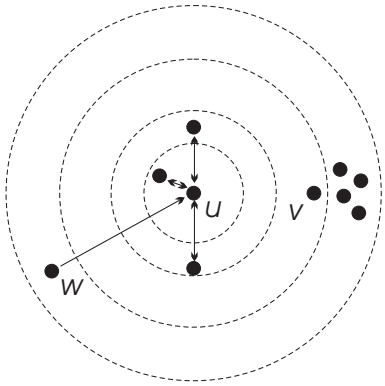

Fig. 2. Example in which the "unselfish" behavior of node u generates a more energy efficient local solution.

inefficiency described above, motivates the design of an "unselfish" variation of the basic protocol, which we call k-NEIGHLEVU, where the U stands for unselfish.

Suppose node u has ended its $(i-1)$th round and still has fewer than k symmetric neighbors. Its behavior is now modified according to the following rules.

- Instead of sending a help control message at level i, which would trigger blind power increases, node u sends an *enquiry* control message, carrying the same data as the help request.
- In response to an enquiry, a node at level less than i does not immediately step up; rather, it sends a *reply* control message at (temporary) level i, whose purpose is to let u know that it is a potential helper. The reply message contains the sender's ID and current power level. By gathering this information from all the potential helpers, node u is able to identify the locally "optimal" solution in its vicinity.
- Node u schedules one of several possible actions, whose aim is to satisfy the constraint on the symmetric neighbor set (or to get closer to it) at the minimum energy cost: (i) simply increase u's current power, if there are enough elements in $N_i(u) - N_s(u)$ to reach the threshold k; (ii) send a generalized help (i.e., the old-style help request); (iii) send a *selective* help, asking a subset of the nodes in A_i to increase their power levels.

Note that in k-NEIGHLEVU some nodes perform temporary power increases, thus partially impairing our periodic approach to topology control. However, these changes in the power level occur only during the network setup phase, and not during the network operational time, as is the case with per-packet topology control.

Selective help requests call for a decision in order to choose the target nodes. This can be done by again using energy considerations, and ties can be broken randomly. In any case, we remark that, because of full asynchrony and in absence of a global coordination, a solution which is locally optimal at a certain time might become sub-optimal later (e.g., because a certain node in the u's vicinity would have increased its transmit power later, in response to another help message). Unfortunately, predicting transmit power increases is impossible in practice, and the optimizations performed by k-NEIGHLEVU can be regarded only as *heuristics*.

With respect to k-NEIGHLEV, k-NEIGHLEVU allows a finer control of the symmetric neighbor set, so a better energy efficiency is expected. On the other hand, k-NEIGHLEVU in general exchanges more control messages as compared to k-NEIGHLEV, due to up to three phases of interaction (enquiry–reply–help) between nodes. Thus, only simulation can help us to understand the relative overall performance of the two protocols.

Before ending this section, we remark that the optimizations described in [2] can be applied to the final communication graphs produced by both k-NEIGHLEV and k-NEIGHLEVU. In order to apply these optimizations, which are aimed at identifying edges in the communication graph that can be pruned without impairing connectivity and symmetry, it is sufficient that every node, at the end of the protocol execution, sends a message containing its list of symmetric neighbors.

V. SETTING THE VALUE OF k

The desired number of symmetric neighbors k is clearly a fundamental parameter of our protocol: small values of k are likely to induce disconnected communication graphs, while large values force the majority of the nodes to end protocol execution at larger than necessary levels. In this section, we characterize the "ideal" value of k both analytically and through simulation.

Note that, in [2], we already studied the problem of determining the ideal number of neighbors. However, in [2] the focus was on the number of nodes a node could reach, rather than on symmetric neighbors. Moreover, the protocol in [2] was distance based, and we assumed that each node could set its transmitting range to any value between 0 and the maximum range. As a consequence, it was possible to set the nodes' ranges so that each node had exactly k (outgoing) neighbors. Here we are interested in symmetric neighbors and have the availability of only a small number of power settings, which makes it infeasible to obtain exactly k neighbors in all cases. Nonetheless, the results in [2] can be used to prove the following theorem, which holds under the assumption that the radio coverage area is circular.

Theorem 2: Let n nodes be placed uniformly at random in $[0,1]^2$ and assume that maximum power is sufficient for

each node to reach at least k other nodes. Let L_k be the actual communication graph generated by k-NEIGHLEV with parameter k, and let L_k^- be the graph obtained by L_k by removing the asymmetric links. If $k \in \Theta(\log n)$, then L_k^- is connected w.h.p.

Proof: By property (c) of Theorem 1 and the above assumption on maximum power, every node u at the end of the protocol execution has a power level sufficient to reach at least its k closest neighbors. This means that L_k is a super-graph of the k-closest neighbors graph G_k, and also that L_k^- is a super-graph of the symmetric sub-graph G_k^- of G_k. Hence, the proof follows immediately by Theorem 2 of [2]. ■

It can be seen that the same result of Theorem 2 holds also for the communication graph generated by k-NEIGHLEVU.

Note that the result stated in Theorem 2 is weaker than than that stated in Theorem 2 of [2], since the reverse implication ($k \in \Theta(\log n)$ neighbors are *necessary* for connectivity) might not hold. In fact, graph L_k^- in general contains more edges than G_k^-.

The characterization of the ideal value of k given in Theorem 2 is of theoretical interest, but it cannot be used in practice. Thus, we have evaluated the value of k to be used in the k-NEIGHLEV and k-NEIGHLEVU protocols by simulation. For different values of n, we have performed 1000 experiments with increasing values of k, recording the percentage of connected graphs generated at the end of the protocol execution. The ideal value of k, which will be used in the subsequent set of simulations aimed at evaluating the performance of our protocols, is the minimum value such that at least 98% of the graphs generated by the k-NEIGHLEV protocol are connected. Note that, in general, the graphs generated by k-NEIGHLEV and k-NEIGHLEVU are different, so different values of k could be used. We have verified through our experiments that the graphs generated by k-NEIGHLEVU are relatively less connected than that generated by k-NEIGHLEV with the same value of k. However, with the value of k chosen (which guarantees at least 98% of connectivity with k-NEIGHLEV), also the graphs generated by k-NEIGHLEVU show good connectivity on the average: at least 95% of the nodes are in the largest connected component, and in about 90% of the cases the graph is fully connected. For this reason, in the simulations reported in Section VI we have used the same value of k in both protocols.

The ideal values of k for different values of n are reported in Table I. The value of $k = 4$ provides at least 98% connectivity for values of n in the range 150–500, while higher values of k are needed for smaller networks.

TABLE I
IDEAL VALUE OF k FOR DIFFERENT VALUES OF n.

n	k	n	k
50	6	300	4
100	5	350	4
150	4	400	4
200	4	450	4
250	4	500	4

Note that these values are considerably smaller than those needed by the k-NEIGH protocol of [2]. As discussed above, this is due to the fact that, on the average, several symmetric edges are added by k-NEIGHLEV and k-NEIGHLEVU with respect to the minimum value of k required.

Before ending this section, we remark that setting the appropriate value of k in our protocols is not a problem in practice. In fact, the ideal value of k *does not depend* on the node density, but only on the number n of nodes in the network; furthermore, as seen from Table I, the dependence of k on n is very loose, and a relatively inaccurate knowledge of the actual number of nodes in the network is sufficient to generate a good estimate of k.

VI. SIMULATIONS

In this section, we report the result of simulations we have performed to evaluate the performances of our protocols. Besides the k-NEIGHLEV and k-NEIGHLEVU, we have implemented the CBTC protocol of [16], which has been adapted to take into account the transmit power level actually available; i.e., the transmit power level of any node at the end of CBTC execution is rounded up to the next power level available. For details on CBTC, the reader is referred to [16].

For the three topology control protocols considered, we implemented both the basic version (called phase 1 in the following), and the optimization that can be carried out on the communication graph generated after phase 1 (see [2], [16] for details on the optimization phase). The optimization phase of the various protocols is called phase 2 in the following.

The performance of the various protocols will be compared with respect to the following metrics:

- total *energy cost*, defined as the sum of the power levels of all the nodes at the end of the protocol execution;
- average *logical* and *physical* node degree. The logical degree of node u is its degree in the final communication graph, while the physical degree is the number of nodes in the radio coverage area of u at the

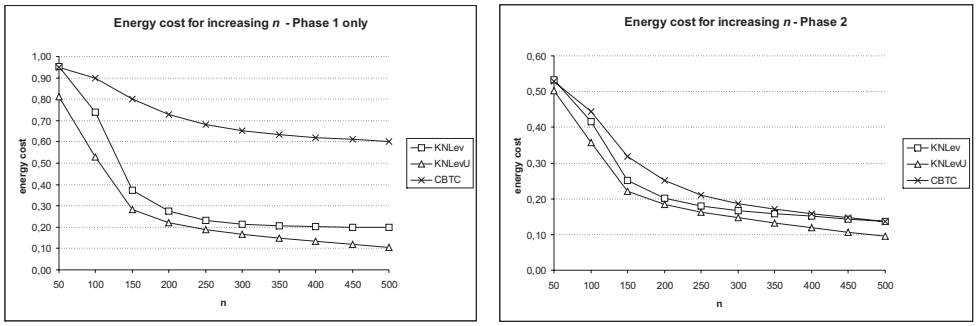

Fig. 3. Energy cost of the k-NEIGHLEV, k-NEIGHLEVU and CBTC protocols as the network size increases, before (left) and after (right) optimization. The energy cost is normalized with respect to the case of no topology control, where all the nodes transmit at maximum power.

end of the protocol execution. Due to the removal of asymmetric links and to optimizations, the physical is usually larger than the logical degree.
– in case of the k-NEIGHLEV and k-NEIGHLEVU protocols, we have also evaluated the average number of message per node sent during phase 1.[3]

The energy cost gives an idea of the energy efficiency of the topology generated by the protocol, while the node degree (especially the physical degree) gives a measure of the expected number of collisions at the MAC layer, and thus, of the expected impact on network capacity [6].

A. Simulation results for minimum density

In the first set of simulations, we have considered networks of increasing size, while maintaining the node density at the minimum level required to guarantee connectivity w.h.p. when all nodes transmit at maximum power.

We have considered the transmit power levels indicated in the data sheets of the Cisco Aironet 350 card [5], namely 1mW, 5 mW, 20mW, 30mW, 50mW, and 100mW. As reported in the data sheets, the transmit range at maximum power is about 244 meters. According to this data, and assuming a distance-power gradient of $\alpha = 2$, we have determined the transmit range at the other power levels, which are 173m, 134m, 109m, 55m and 24m, respectively.

We have considered a simulation area of 1 square kilometer. According to the data reported in [15], the minimum number of nodes to be deployed in the simulation area in order to generate a communication graph which is connected w.h.p. when all the nodes transmit at maximum power is about 100. We have then increased the number of nodes in steps of 50, up to 500, scaling the simulation area in such a way that the node density remains the minimum necessary for connectivity at maximum power. We have also considered smaller networks, composed of 50 nodes. The values of n, and the corresponding side s of the simulation area are reported in Table II.

TABLE II
MINIMUM DENSITY SCENARIOS CONSIDERED IN THE SIMULATIONS. THE MAXIMUM TRANSMITTING RANGE IS $244m$. THE AREA SIDE s IS EXPRESSED IN Km.

n	s	n	s
50	0.72	300	1.74
100	1	350	1.88
150	1.22	400	1.95
200	1.44	450	2.07
250	1.63	500	2.16

The results of this set of simulations are reported in Figures 3–5, and are averaged over 1000 runs. From the figures, it is seen that:

– both k-NEIGHLEV and k-NEIGHLEVU clearly outperform CBTC in terms of energy cost when optimizations are not implemented. The relative savings achieved by our protocols increase with the network size, and they can be as high as 67% (k-NEIGHLEV) and 77% (k-NEIGHLEVU).
– when optimizations are implemented, our protocols still perform better than CBTC in terms of energy cost. However, in this case the relative gain in performance is less significant. The relative improvement of k-NEIGHLEV with respect to CBTC can be as high as 21% (when $n = 150$), but tends to be less significant as n increases. On the contrary, the improvement achieved by k-NEIGHLEVU with respect to CBTC tends to increase with n, and can be as high as 30% when $n = 500$. The energy savings achieved by our protocols with respect to the case of no topology control increase with n, and can be as high as 86% for k-NEIGHLEV, and as high as 91% for k-NEIGHLEVU.

[3]We recall that phase 2 requires one further message per node sent in both protocols.

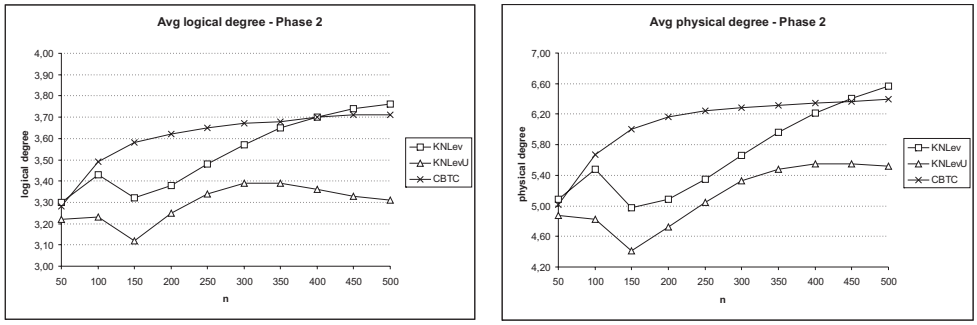

Fig. 4. Average logical (left) and physical (right) node degree after the optimization phase.

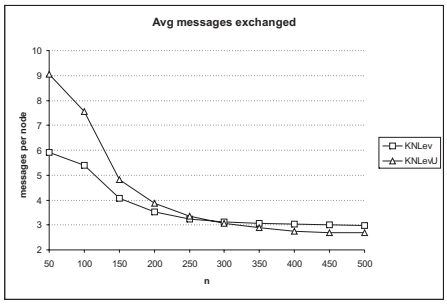

Fig. 5. Average number of message per node sent during phase 1 of the k-NEIGHLEV and k-NEIGHLEVU protocols.

– concerning the logical and physical node degree of the communication graph, k-NEIGHLEVU performs clearly better than the other protocols, especially in terms of physical degree (which is the one that determines the expected impact on network capacity). The average physical degree when k-NEIGHLEVU is used is as much as 26% smaller than that generated by CBTC, and as much as 12% smaller than that generated by k-NEIGHLEV. With respect to the case of no topology control, k-NEIGHLEVU reduces the average physical node degree by about 75%.
– k-NEIGHLEVU always perform better than k-NEIGHLEV, with respect to both energy cost and node degree. In terms of messages per node sent, k-NEIGHLEVU exchanges relatively more messages than k-NEIGHLEV when the network size is small. However, when the network size increases, the situation is reversed: for $n \geq 300$, k-NEIGHLEVU generates fewer messages than k-NEIGHLEV. Thus, when the network size is large, k-NEIGHLEVU performs better than k-NEIGHLEV in all respects.

B. Simulation results for increasing density

In the second set of experiments, we have evaluated the effect of node density on the performance of the various protocols. Starting with the minimum density scenario for $n = 100$, we have increased the number of nodes up to $n = 400$ while leaving the side s of the simulation area unchanged (1 kilometer in all cases).

The results of this set of simulations, which are not reported for lack of space, are practically identical to those obtained in the minimum density scenario. In other words, for a given value of n, the performance of k-NEIGHLEV, k-NEIGHLEVU and CBTC does not change with the side of the deployment region, i.e., with the node density. This is due to the fact that the protocols considered rely on relative, rather than absolute, location information. In case of k-NEIGHLEV and k-NEIGHLEVU, the information considered is relative distance, while in case of CBTC it is relative angular displacement.

We believe this result is quite interesting, since it shows that *it is only the size n of the network that determines the performances achieved by the various protocols*, in terms of both energy savings and increase in network capacity.

C. Optimizing the power levels

The results reported in the previous sections use power levels typical of the CISCO Aironet 350 card, which are not necessarily optimal for our purposes. To motivate this statement, consider the scenario in which $n = 100$ nodes are placed in a simulation area of side $s = 1Km$, and the transmitting ranges of 244m, 173m, 134m, 109m, 55m and 24m are used. Disregarding the border effect, the expected number of neighbors of a node at level l can be easily calculated as $E[N_l] = 99 \cdot \pi r_l^2$, where r_l is the transmitting range corresponding to power level l, expressed in kilometers. The expected number of neighbors at every level is reported in Table III.

As seen from the Table, the values of $E[N_l]$ using the CISCO specification are very badly distributed. In particular, it is expected that few nodes will take advantage of the levels 0 and 1, which potentially provide the better savings.

In order to circumvent this problem, we have designed the power levels according to the following criterion. The

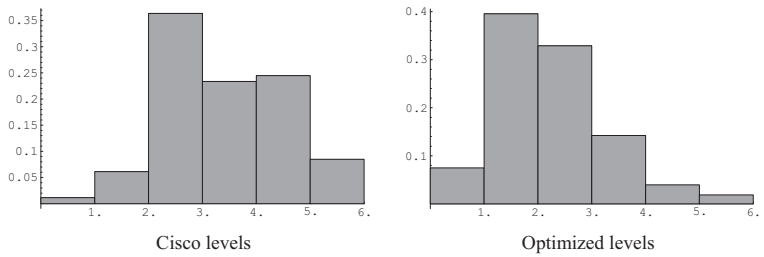

Fig. 6. Empirical distribution of the node power levels using the Cisco (left) and optimized (right) power levels.

TABLE III

EXPECTED NUMBER OF NEIGHBORS AT EVERY POWER LEVEL.

	Cisco		Optimized	
l	r_l	$E[N_l]$	r_l	$E[N_l]$
0	0.024	0.18	0.056	1
1	0.055	0.94	0.113	4
2	0.109	3.69	0.150	7
3	0.134	5.58	0.179	10
4	0.173	9.3	0.204	13
5	0.244	18.5	0.244	18.5

maximum power level is left unchanged, since this value is needed for connectivity purposes. The minimum level is determined in such a way that $E[N_0] = 1$. The other power levels are set so that the remaining neighbors are evenly distributed among them. The optimized choice of power levels is also reported in Table III. As seen in Figure 6, which reports the empirical distribution of the node power levels obtained from 100 runs of the simulator with $n = 100$, with the optimized power levels, more nodes use smaller power levels. Thus, an overall reduction in energy consumption is expected.

To validate this, we have re-executed our simulation experiments for $n = 100$, and evaluated the resulting performance of the various protocols. Furthermore, we have performed the optimization process for two other values of n, namely $n = 50$ (small networks) and $n = 400$ (large networks), and repeated the experiment. The results of our simulations are summarized in Table IV. For every protocol and performance metric, the table reports the percentage improvement with respect to the case of the Cisco levels. The energy cost and node degrees reported in the table refer to the communication graphs obtained after phase 2.

As seen from the table, optimizing power levels has beneficial effects on the performance of k-NEIGHLEV and k-NEIGHLEVU for $n = 50, 100$: the energy cost is reduced by more than 10%, the physical degree is reduced by as much as 7.3%, and the number of messages exchanged is considerably reduced. k-NEIGHLEV experienced more benefit from the optimization of the power levels, while CBTC got only a moderate reduction in the energy cost.

The situation is quite different for large networks ($n = 400$). In this case, the performance of k-NEIGHLEV and CBTC are considerably reduced with respect to the case of Cisco levels: the energy cost is worse by about 13%. However, the messages exchanged by k-NEIGHLEV decrease by about 35%. Interestingly, k-NEIGHLEVU still benefits from the optimized power levels: at the expense of an increase in the number of messages exchanged, both the energy cost and the node degree are reduced by more than 10%.

We believe that the poor performance of k-NEIGHLEV and CBTC with optimized power levels in large networks is due to the fact that, when n is large, a different choice of the power levels should be used. In particular, setting the minimum level in such a way that $E[N_0] = 1$ is too pessimistic when n is large, and a smaller power for the minimum level 0 should be used. We leave a closer investigation of this point as further research.

VII. DISCUSSION

The simulation results reported in the previous section have shown that our protocols, and in particular k-NEIGHLEVU, compare favorably with CBTC, both in terms of energy efficiency and expected benefits on network capacity. Further, our protocols do not require specialized hardware on the nodes, but they simply leverage features available on standard wireless cards. This is a major advantage over CBTC, which relies on directional information provided by directional antennas, or other expensive devices. In terms of message overhead, our protocols exchange few control messages (less than 6 messages per node when $n \geq 150$), thus showing good scalability performance. On the other hand, CBTC produces a connected communication graph whenever possible, while our protocols provide connectivity w.h.p.

It is also interesting to compare qualitatively the performance of our protocols to that of other level-based topology control protocols, such as COMPOW [11] and CLUSTERPOW [8].

TABLE IV

PERCENTAGE IMPROVEMENT WITH OPTIMIZED POWER LEVELS.

	k-NEIGHLEV				k-NEIGHLEVU				CBTC		
n	Cost	LogD	PhD	Mess	Cost	LogD	PhD	Mess	Cost	LogD	PhD
50	14.2%	1.5%	4.5%	16.6%	11.5%	0.9%	3.9%	19.4%	10.9%	0	0.2%
100	12.9%	2.3%	7.3%	24.3%	5.6%	0.3%	2.5%	28.9%	8.7%	0.3%	0
400	-13.6%	-0.3%	-2.7%	34.7%	11.0%	10.1%	12.8%	-32.2%	-13.0%	0	0

The goal of COMPOW is to determine in a fully distributed fashion a minimum *common* value of the transmit power level such that the resulting communication graph is connected. It is known that such level corresponds to the longest edge of the MST built on the nodes [15]. Thus, an analysis of the expected COMPOW performance can be done by evaluating the expected length of this edge. This has been done in [15], from which it is seen that, when $n = 100$, the expected length of the longest MST edge is about $160m$ (when nodes are distributed in a square with side $s = 1Km$). Assuming the CISCO power levels, this means that, on the average, the nodes with COMPOW will converge to the common power level corresponding to the transmitting range of 173m. Comparing the resulting energy cost with that of k-NEIGHLEVU under the same conditions, we have that our protocol is about 30% more energy efficient than COMPOW.

As observed in [8], COMPOW performs poorly when the nodes are not uniformly distributed. To circumvent this problem, the authors of [8] presented CLUSTERPOW. In CLUSTERPOW, every node u in the network maintains several routing tables, one for each power level. The routing table for level i, RT_i, is updated by a routing daemon (there is one daemon for every power level), and contains all the nodes that are reachable by node u using transmit power level at most i. CLUSTERPOW thus induces a clustering on the network nodes in a natural way: for every node u, several clusters are defined, with the cluster at level i formed by those nodes that u can reach using power level at most i. When node u needs to send a message to v, it sends the message with power level j, where j is the minimum level such that $v \in RT_j$.

A comparison of the CLUSTERPOW performance with that of our protocols is not immediate. A first observation is that CLUSTERPOW, contrary to our protocols, requires global knowledge, which incurs a considerable message overhead. Furthermore, CLUSTERPOW changes the transmit power on a per-packet basis. This means the its performance heavily depends on the traffic pattern, as illustrated by the following example.

Suppose node u wants to send a message to v, and that 50mW is the minimum power needed for u to reach v. This means that, along every path connecting u and v, there exists at least one link which requires at least that power. Let us denote with $P = u, w_1, \ldots, w_k, v$ one of such paths, and let w_i, w_{i+1} be the last hop in P that requires at least power 50mW. Using CLUSTERPOW, all the nodes preceding w_i in the path will use power 50mW, while those following w_i will use lower powers. Note that, in general, node w_i could have been reached using smaller power than 50mW. Thus, CLUSTERPOW might be inefficient in some situations, forcing many of the nodes to transmit at higher powers than necessary. Clearly, the occurrence of such a scenario depends on the network topology and on the traffic pattern, so predicting the overall CLUSTERPOW performance is not easy. The authors of [8] proposed a different version of CLUSTERPOW to address this problem, which unfortunately cannot be easily implemented in practice.

To some extent, our protocols solve the problem with CLUSTERPOW outlined above. Considering the transmission of a message from u to v along P, only nodes w_i and w_{i+1} (and possibly other nodes, if there exist other 50mW links in P) will use power 50mW to transmit, while the other nodes will use in general lower powers. On the other hand, our protocols might be inefficient when the communication patterns are localized: if u has power level 100mW and needs to send a message to a node that is within its 1mW transmitting range, the communication is highly inefficient.

The discussion above motivates our feeling that, once the technological problems encountered in the implementation of per-packet power control will be tackled, a combination of periodic (to adjust the maximum transmit power) and per-packet (to send messages in the vicinity of a node) topology control will be the best choice.

VIII. DEALING WITH MOBILITY

The principal complicating factor in dealing with mobility for neighborhood-based protocols is the inherently transient nature of the neighbor set of a node. Due to this, we can not hope to calculate N_s exactly but only to estimate it. Consider, for example, when a node in $N_s(u)$ moves out of range of u. There is an unavoidable delay before this event is detected and, during this time, $N_s(u)$

Main:

$l = 0$; $N_i \leftarrow \emptyset$; $N_s \leftarrow \emptyset$
every T seconds do
  send beacon message (u, l, N_i)

Upon receiving an ordinary (non-beacon) message from node v:

$N_i \leftarrow N_i \cup \{v\}$
if Timer$_v = 0$ then set Timer$_v$ to expire in T seconds

Upon receiving a beacon message (v, l_v, $N_i(v)$):

$N_i \leftarrow N_i \cup \{v\}$
if $u \in N_i(v)$ then $N_s \leftarrow N_s \cup \{v\}$
if $|N_s| > k_{\text{high}}$ then call decrease_neighbors()
set Timer$_v$ to expire in T seconds

Upon expiration of Timer$_v$:

$N_i \leftarrow N_i - \{v\}$; $N_s \leftarrow N_s - \{v\}$
if $|N_s| < k_{\text{low}}$ then call increase_neighbors()

Fig. 7. Procedure for estimating N_s performed by node u

is not accurate. One must also be careful not to adjust power levels too quickly when topology changes occur, lest the protocol lead to unstable behavior.

Based on the discussions above, our version of k-NEIGHLEV for mobility is based on the following two key ideas. First, in order to estimate N_s, nodes periodically send beacon messages containing their estimated N_i sets at their current power levels. If node u hears a beacon from node v and $u \in N_i(v)$, then u and v are symmetric neighbors. Second, instead of trying to maintain $|N_s|$ at a value of exactly k, we set low and high water marks on $|N_s|$, denoted by k_{low} and k_{high}, respectively. A node initiates steps to increase its neighbor set size only when its estimated $|N_s|$ falls below k_{low} and tries to decrease its neighbor set size only when the estimated $|N_s|$ exceeds k_{high}.

The details of our procedure for estimating N_s are given in Figure 7. When a node is first powered up, it initiates this procedure. Nodes send beacon messages containing their N_i sets every T seconds, where T is a user-specified parameter that provides a trade off between protocol overhead and delay in detecting changes to the neighbor set. Whenever a node u receives a message (beacon or otherwise) from node v, u adds v to its N_i set. When u receives a beacon message from v, u also adds v to its N_s set if u appears in the N_i set of v that is contained in the beacon message. Also when receiving a beacon message from v, u sets a timer to expire in T seconds. If the timer expires before u receives another beacon from v, then v is no longer an in-neighbor (nor a symmetric neighbor) of u.

increase_neighbors()

while ($|N_s| < k_{\text{low}}$) and ($l < max$) do
  count $\leftarrow 0$
  while ($|N_s| < k_{\text{low}}$) and (count $< l$) do
    send help message (u, l, N_i)
    wait T_l
    count $\leftarrow$ count+1
  if ($|N_s| < k_{\text{low}}$) then $l \leftarrow l + 1$

Upon receiving a help message (v, l_v, $N_i(v)$):

$N_i \leftarrow N_i \cup \{v\}$
if ($u \notin N_i(v)$) then
  if ($l < l_v$) then $l \leftarrow l + 1$
  send beacon message (u, l, N_i)
if $u \in N_i(v)$ then $N_s \leftarrow N_s \cup \{v\}$

Fig. 8. Procedure for increasing N_s performed by node u

The remainder of the protocol sets forth the actions to be taken when the size of N_s falls below k_{low} or exceeds k_{high}. Figure 8 shows the procedure for increasing $|N_s|$, while Figure 9 shows how a decrease in $|N_s|$ is achieved.

There are two main differences in how $|N_s|$ is increased with mobility (Figure 8) compared to the stationary version of k-NEIGHLEV. First, a node's N_i set is included with its help message. This is to allow nodes that receive help messages to use the most recent information to determine if the sender is a symmetric neighbor given that the N_s set is only an estimate of the actual symmetric neighbor set. The second, and more important, difference is that nodes which respond to help messages increase their power level by only one setting when mobility is involved, whereas in the stationary case they increase their power to match that of the sender. Since this does not guarantee that responders will be heard by the help requester, its N_s set might not be increased by this response. This is the reason that help requesters send help messages multiple times at the same power level.

While resending help messages at the same power level might seem inefficient, we do this to avoid the following scenario, which could be quite common in mobile networks. A node requires a high power level while communicating in a sparse part of the network and then moves to a denser part where nodes are communicating with much lower power levels. Since the node sends its help message at its current power level, the basic k-NEIGHLEV protocol would potentially cause many nodes in the dense part of the network to switch to very high power levels, which is clearly wasteful of energy. The procedure in Figure 8, while using more control messages and time, produces more graceful changes in power levels and avoids unnecessary large increases in power by many nodes.

decrease_neighbors()

while ($l > 0$) and ($|N_s| > k_{\text{high}}$) do
send a check_reduce message (u, l)
wait T_l
if stop message received then exit
otherwise $l \leftarrow l - 1$

Upon receiving a check_reduce message (v, l_v):

if ($v \in N_s$) and ($|N_s| = k_{\text{low}}$) and ($l \geq l_v$)
then send stop message to v

Fig. 9. Procedure for decreasing N_s performed by node u

The need to decrease the number of neighbors (Figure 9) arises only when there is mobility since the basic k-NEIGHLEV protocol ensures that nodes' power levels are set as small as possible in a certain sense. With mobility, a node could require a high power level in one area but that level could produce far more neighbors than necessary when it moves to a new area. The ability to decrease levels is therefore required. We must be cautious when power levels are decreased, however, lest we leave another node with too few neighbors causing it to initiate a round of help messages and possibly leading to circular behavior. Thus, before we allow a node to reduce its level, we force it to send a "check reduce" message to its current neighbors to make sure that the reduction will not leave any node with too few neighbors. If any node that hears a check reduce message from v has v as a symmetric neighbor, has the minimum number of symmetric neighbors currently, and is in danger of not hearing v if v's power level is reduced, then it sends a stop message to v. If v hears at least one stop message, then it does not reduce its power level.

One remaining question is how to choose k_{low} and k_{high}. A conservative approach would be to set k_{low} to the minimum value necessary to ensure a connected network with high probability. With this setting, if nodes operate at values higher than the low water mark most of the time, energy is wasted. However, setting k_{low} even one below this minimum value could occasionally result in disconnected networks. Hence, the low water mark should be determined based on whether periods of disconnection can be tolerated or not. The high water mark should be set as small as possible such that the protocol is not triggered too often, thereby consuming more energy in executing the protocol than is saved through the topology control that results. Such an evaluation can only be done through simulation and is a topic for future research.

Observe that most of the issues raised when dealing with mobile networks are in common with the scenario of *dynamic* networks, in which nodes enter and leave the network at different times. Thus, the mobile version of our protocol can also be used to deal with dynamic networks.

REFERENCES

[1] M. Bahramgiri, M. Hajiaghayi, V.S. Mirrokni, "Fault-tolerant ad 3-Dimensional Distributed Topology Control Algorithms in Wireless Multi-hop Networks", *Proc. IEEE Int. Conference on Computer Communications and Networks*, pp. 392–397, 2002.

[2] D. M. Blough, M. Leoncini, G. Resta, and P. Santi, "The k-NEIGH Protocol for Symmetric Topology Control in Ad Hoc Networks", *in Proc. ACM MobiHoc 03*, pp. 141–152, June 2003.

[3] D. M. Blough, M. Leoncini, G. Resta, and P. Santi, "k-NEIGHLEV: a Practical Realization of Neighborhood-Based Topology Control in Ad Hoc Networks", *Tech. Rep. IIT-TR-12/03*, Istituto di Informatica e Telematica, Italy, July 2003.

[4] S.A. Borbash, E.H. Jennings, "Distributed Topology Control Algorithm for Multihop Wireless Networks", *Proc. IEEE Int. Joint Conference on Neural Networks*, pp. 355–360, 2002.

[5] *Cisco Aironet 350 data sheets*, available at `http://www.cisco.com/en/US/products/hw/ /wireless`.

[6] P. Gupta, P.R. Kumar, "The Capacity of Wireless Networks", *IEEE Trans. Information Theory*, Vol. 46, n. 2, pp. 388–404, 2000.

[7] Z. Huang, C. Shen, C. Srisathapornphat, C. Jaikaeo, "Topology Control for Ad Hoc Networks with Directional Antennas", *Proc. IEEE Int. Conference on Computer Communications and Networks*, pp. 16–21, 2002.

[8] V. Kawadia, P.R. Kumar, "Power Control and Clustering in Ad Hoc Networks", *in Proc. IEEE Infocom 03*, 2003.

[9] L. Li, J.H. Halpern, P. Bahl, Y. Wang, R. Wattenhofer, "Analysis of a Cone-Based Distributed Topology Control Algorithm for Wireless Multi-hop Networks", *Proc. ACM PODC 2001*, pp. 264–273, 2001.

[10] J. Liu, B. Li, "MobileGrid: Capacity-aware Topology Control in Mobile Ad Hoc Networks", *Proc. IEEE Int. Conference on Computer Communications and Networks*, pp. 570–574, 2002.

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

[12] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava, "Energy-Aware Wireless Microsensor Networks", *IEEE Signal Processing Magazine*, Vol 19, n. 2, pp. 40–50, 2002.

[13] R. Ramanathan, R. Rosales-Hain, "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment", *Proc. IEEE Infocom 2000*, pp. 404–413, 2000.

[14] V. Rodoplu, T.H. Meng, "Minimum Energy Mobile Wireless Networks", *IEEE Journal Selected Areas in Comm.*, Vol. 17, n. 8, pp. 1333–1344, 1999.

[15] P. Santi, D.M. Blough,"The Critical Transmitting Range for Connectivity in Sparse Wireless Ad Hoc Networks", *IEEE Transactions on Mobile Computing*, Vol. 2, n. 1, pp. 1–15, January-March 2003.

[16] R. Wattenhofer, L. Li, P. Bahl, Y. Wang, "Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks", *Proc. IEEE Infocom 2001*, pp. 1388-1397, 2001.