

Self Organization and Self Maintenance of Mobile Ad Hoc Networks through Dynamic Topology Control

Douglas M. Blough¹, Giovanni Resta², Paolo Santi², and Mauro Leoncini³

¹ School of ECE, Georgia Tech, Atlanta, GA, USA

² Istituto di Informatica e Telematica del CNR, Pisa, Italy

³ Università di Modena e Reggio Emilia, Italy

Abstract. One way in which wireless nodes can organize themselves into an ad hoc network is to execute a topology control protocol, which is designed to build a network satisfying specific properties. A number of basic topology control protocols exist and have been extensively analyzed. Unfortunately, most of these protocols are designed primarily for static networks and the protocol designers simply advise that the protocols should be repeated periodically to deal with failures, mobility, and other sources of dynamism. However, continuously maintaining a network topology with basic connectivity properties is a fundamental requirement for overall network dependability. Current approaches consider failures only as an afterthought or take a static fault tolerance approach, which results in extremely high energy usage and low throughput. In addition, most of the existing topology control protocols assume that transmission power is a continuous variable and, therefore, nodes can choose an arbitrary power value between some minimum and maximum powers. However, wireless network interfaces with dynamic transmission power control permit the power to be set to one of a discrete number of possible values. This simple restriction complicates the design of the topology control protocol substantially. In this paper, we present a set of topology control protocols, which work with discrete power levels and for which we specify a version that deals specifically with dynamic networks that experience failures, mobility, and other dynamic conditions. Our protocols are also novel in the sense that they are the first to consider explicit coordination between neighboring nodes, which results in more efficient power settings. In this chapter, we present the design of these topology control protocols, and we report on extensive simulations to evaluate them and compare their performance against existing protocols. The results demonstrate that our protocols produce very similar topologies as the best protocols that assume power is a continuous variable, while having very low communication cost and seamlessly handling failures and mobility.

Keywords: Wireless multihop networks, topology control, dynamic networks, fault tolerance.

1 Introduction

The topology control problem in wireless ad hoc networks is to choose the transmission power of each node in such a way that energy consumption is reduced and some prop-

erty 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 [9] 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 hops⁴. Given the limited availability of both energy and capacity in ad hoc networks, topology control is thus considered a major building block of forthcoming wireless networks.

Ideally, a topology control protocol should be asynchronous, fully distributed, fault-tolerant, 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.

Most existing topology control protocols focus on initial construction of a good topology. These protocols concentrate on static network environments where one-time topology construction is sufficient, but do not explicitly consider how to maintain a good topology as network conditions change. Sources of dynamism in ad hoc networks include mobility, failures, and dynamic joins of nodes. One approach, designed primarily to deal with node failures, is to construct an initial topology that is highly redundant and can therefore tolerate some dynamic changes without impairing basic network properties such as connectivity. However, this approach is not sufficient to deal with highly dynamic environments such as those arising from node mobility. In addition, due to the high level of redundancy in the initial construction, these topologies are inefficient in that they force nodes to use higher transmission powers than necessary at a given time to withstand potential future changes. The higher than necessary transmission powers result in higher energy consumption by nodes, which reduces network lifetime, and increased interference in the network, which degrades network performance. Thus, these static approaches favor short-term dependability at the expense of longer-term network survivability, while at the same time incurring very serious performance costs.

In this chapter, we propose a new approach to topology maintenance, which dynamically adjusts the topology on an as needed basis in response to network changes. Our approach considers failures, and other sources of dynamism, as an inherent feature in the network and explicitly considers how to maintain a good topology while operating the protocol in a stable and efficient manner. Thus, we do not simply propose to reexecute a static protocol periodically, which would result in a very high overhead. Rather, we maintain local network conditions within a certain range and take explicit local steps to maintain those conditions only when they fall out of the specified range. This allows us to maintain global network properties in the presence of failures, while executing maintenance operations locally and only when sufficient changes have occurred to warrant topology adjustment. In our protocols, we also assume the use of discrete transmission power levels, an assumption that holds true in all existing net-

⁴ This does not necessarily hold true in worst-case distributions of nodes and for a particular choice of the contention measure, see [5].

work interface cards with dynamic transmission power adjustment (a basic requirement for topology control). We also consider explicit coordination between nodes to optimize the topology, rather than having each node optimize its own conditions. In Section 8, we do a thorough evaluation of our protocols, which validates their essential features, namely that they produce good quality topologies with low energy cost and interference while seamlessly and efficiently handling highly dynamic network conditions.

2 Related work

The topology control problem [22] has been deeply investigated in the literature in recent years, including theoretical studies aimed at characterizing optimal topologies according to some performance metric (see, e.g., [19, 21, 25]), and more practical approaches presenting distributed, localized topology control protocols, sometimes with proven performance bounds with respect to optimal [2, 3, 8, 10, 15, 18, 20, 27, 28]. Included in this work is our original k -Neighbors approach [2, 3], upon which this current work builds. Some papers [4, 5] also addressed the topology control problem with the goal of reducing interference, instead of energy consumption as traditionally done in the topology control literature. In this section, we discuss the topology control approaches most relevant to this chapter, namely those explicitly designed to address fault-tolerance and/or node mobility.

Fault-tolerant topology control has been addressed in some recent papers. Typically, fault-tolerance is achieved by requiring some level of redundancy in the constructed topology, e.g., k -connectivity (for some $k > 1$) of the communication graph instead of simple connectivity. For instance, in [1], the authors generalize the CBTC protocol of [27] to construct k -connected topologies in a three-dimensional setting. A distributed algorithm based on localized construction of k -spanning sub-graphs is presented in [16], while [17] mainly focuses on characterizing the critical transmission range for k -connectivity. Other studies essentially extend topology optimization problems to the case of k -connectivity, e.g. [6], which deals with heterogeneous networks, and [25], which focuses on the optimal k -connected topologies for all-to-one and one-to-all communications. However, all of these approaches use static redundancy, which produces denser topologies with higher transmission powers, and correspondingly more interference. Thus, the topologies generated by these protocols suffer both from high energy usage and low throughput, since many studies have shown that wireless multi-hop network performance is interference limited. Despite the higher overheads associated with static redundancy protocols, none of them are guaranteed to maintain connectivity for mobile networks. Thus, the benefits gained from the high overheads are not at all clear. Our alternative approach, proposed herein, dynamically adjusts the topology on an as needed basis to maintain certain properties while failures and node mobility are occurring.

Relatively few papers have been explicitly concerned not only with construction of the network topology, but also with its maintenance in presence of dynamic network conditions due to, e.g., node mobility, failures, and new nodes joining the network. In [27], the authors describe a procedure to reconfigure the network topology built by CBTC in presence of node join/leaves. However, no evaluation of the procedure's over-

head nor its capability to maintain a good topology is carried out. Most of the dynamic events discussed in [27] require the protocol to be completely reexecuted by at least one node, which incurs substantial cost in environments with moderate to high dynamism. In [26], the authors present a distributed algorithm for building a k -connected topology in a three-dimensional network, and describe a procedure for updating the topology in presence of dynamic network conditions. However, again there is no evaluation of the performance of the protocol under dynamic conditions.

To the best of our knowledge, the only papers that explicitly deal with topology control in presence of node mobility are [18] and [21]. In [18], the authors introduce the notion of *contention index*, and show through simulation that capacity of a mobile network shows a high degree of correlation with the contention index independently of node speed. Then, they present a localized, distributed protocol called MobileGrid aimed at keeping the contention index of each node close to the optimal value. However, while properties of the contention index have been evaluated in a mobile setting, the overall MobileGrid protocol has been evaluated only in stationary networks. In [21], the authors present two simple neighborhood-based protocols and evaluate their performance in both static and mobile settings. However, the paper only evaluates the throughput and average delay experienced when a set of random flows are created in the network, with and without topology control. Although throughput and delay of a set of random flows are an indirect indication of the quality of the underlying network topology, an explicit evaluation of important network properties such as connectivity, average node degree, and energy cost is missing in [21]. Thus, the one presented in this chapter is, to the best of our knowledge, the first distributed topology control approach whose performance (expressed in terms of connectivity, average degree, and energy cost of the constructed topologies) is extensively evaluated in both stationary and mobile networks. In addition, the protocols of [18, 21] both suffer from a technical flaw, which is described in detail in the next section.

In addition to dealing with dynamic networks, the topology control protocols we present herein are based on selection of a discrete and finite set of transmission power levels. The idea of using level-based power changes was introduced in [20], and further developed in [12], neither of which considers dynamic networks. The protocols proposed in [12, 20] change the transmission power on a per-packet basis: the network nodes exchange messages at different power levels in order to build the routing tables (one for each level); the information contained in these tables is then used by the nodes to set the appropriate transmission power when sending messages. Since the topology of the network is not changed by these protocols, we call this approach *power control*, instead of topology control. Nevertheless, the assumption that transmission power can only be set to certain predetermined values, which our protocols, as well as those of [12, 20], adopt is coherent with all existing wireless networking cards that have power control capability. Thus, this feature is essential to a practical topology control approach.

A final novel aspect of the work described in this chapter is an “unselfish” version of our topology control protocol, in which nodes try to coordinate their power increases in order to “minimize” the overall local power consumption. To our knowledge, this is the first protocol to consider explicit coordination between nodes. In summary, our protocols are the first to be evaluated thoroughly in dynamic settings, the first to con-

sider discrete power levels without the use of per packet power control, and the first to consider explicit coordination between nodes.

3 Preliminaries and Working Assumptions

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

- nodes can transmit messages at different power levels, denoted p_0, \dots, p_{max} , which are the same for all nodes,
- message loss is handled at the MAC layer, e.g. through a retransmission mechanism, and
- 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 .

The second assumption is very similar to the assumption of an abstract MAC layer, recently proposed by Kuhn, Lynch, and Newport [13]. The third assumption, namely that the wireless medium is symmetric, is not essential. In fact, this assumption is not used in the dynamic version of the protocol presented in Section 7. In this section, we adopt this assumption in order to simplify the presentation of the static version of the protocol. However, the static version could easily be augmented to exchange neighbor lists (as is done in the dynamic protocol version) instead of simple power levels (see static protocol reported in Figure 1). With this augmentation, the symmetric medium assumption can be removed. This change, while complicating the protocol specification, would not increase the *number* of messages exchanged but would incur an increase in the size of each message.

For the sake of brevity, in the following we will say that a node is *at level i* if its current transmission power is set to p_i . Also, we will let A_i denote the radio coverage area of a given node at level i , $i = 0, \dots, max$. 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 is 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 $E = \{(u, v) : v \text{ is within } u\text{'s transmission range at the current power level}\}$ 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 $N_s(u) = N_i(u) \cap N_o(u) = \{v \in N : (v, u) \in E \text{ and } (u, v) \in E\}$.

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 our topology control protocols is to cause $N_s(u)$ to contain k (or slightly more than k) nodes, where k is an appropriately chosen

parameter⁵. 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 of 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 transmission 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 [18]. 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 transmission 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 use explicit control messages is the LINT/LILT protocol of Ramanathan and Rosales-Hain [21]. However, in [21], 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 neighbor-based protocols make use of explicit control messages.

4 Basic Protocol for Static Networks

Our neighborhood-based topology control with power levels (NTC-PL) protocol implements the following idea. By circulating short control messages, nodes can 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 6, 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

⁵ This requirement will be loosened in the mobile version of the protocol.

are used to trigger some of the receivers to increase their transmission 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 transmission power level is reached. The routines that are executed upon the reception of control messages are described next.

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 ⁶. 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 step by step power increase procedure described above.

A pseudo-code description of the NTC-PL protocol is shown in Figure 1. In order to improve readability, we drop the 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 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 always the minimum necessary for it to have k symmet-

⁶ 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 $v \notin N_i$
 - $l_v = \lambda$;
 - $N_i = N_i \cup \{v\}$;
 - if $l \geq l_v$ then $N_s = N_s \cup \{v\}$;
- Upon receiving a help message (v, λ) :
 - if $v \in N_i$ and $v \notin N_s$ stepwise-increase($l + 1, l_v$);
 - if $v \notin N_i$
 - $l_v = \lambda$;
 - $N_i = N_i \cup \{v\}$;
 - if $l < l_v$ stepwise-increase($l + 1, l_v$);
 - else $N_s = N_s \cup \{v\}$;
- Procedure stepwise-increase (i, f):
 - for $h = i, \dots, f$, do:
 - set $l = h$;
 - send beacon (u, l) ;
 - $N_s = N_s \cup \{z\}$, for any $z \in N_i$ s.t. $l = l_z$;

Fig. 1. Algorithm NTC-PL performed by node u .

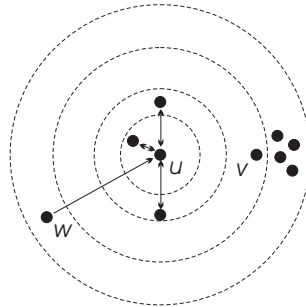


Fig. 2. Example in which the “unselfish” behavior of node u generates a more energy efficient local solution.

ric neighbors, except when one of its in-neighbors requires help in achieving its own symmetric neighbor requirement.

Theorem 1 *The NTC-PL protocol satisfies the following properties:*

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

moreover, there exist values T_l , $l = 0, \dots, \max$, of the waiting times such that:

(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: By 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 sufficiently large, 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. ■

5 Protocol Variation with Unselfish Behavior

The NTC-PL protocol 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 NTC-PL protocol causes *all* nodes that receive it to become symmetric neighbors of the sender, if they are not already. 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, NTC-PL 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 NTC-PL is illustrated in Figure 2. Suppose $k = 4$ and the transmission powers of nodes u , v , and w are set to 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 transmission power settings: 1mW, 5mW, 20mW, 30mW, 50mW, and 100mW (these are the power levels used in the Cisco Aironet card [7]). 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, from a total energy standpoint, 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 NTC-PL inefficiency described above, motivates the design of an “unselfish” variation of the basic protocol, which we call NTC-PLU.

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 NTC-PLU 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 transmission power later, in response to another help message). Unfortunately, predicting transmission power increases is impossible in practice, and the optimizations performed by NTC-PLU can be regarded only as *heuristics*.

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

Before ending this section, we remark that the optimizations based on the well-known triangular inequality described in [2] can be applied to the final communication graphs produced by both NTC-PL and NTC-PLU. 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.

6 Setting the Value of k

The desired number of symmetric neighbors k is clearly a fundamental parameter of our protocols: 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, the problem of determining the ideal number of neighbors has already been studied in [2]. 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 it was assumed that each node could set its transmission 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] (which in turn depends on a fundamental theorem in [29]) 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 NTC-PL 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.⁷*

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

⁷ W.h.p. means with probability converging to 1 as the number n of network nodes goes to infinity.

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 the fact that, as proven in Theorem 2 of [2], $k \in \Theta(\log n)$ implies that graph G_k^- , which is a sub-graph of L_k^- , is connected w.h.p. ■

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

Note that the result stated in Theorem 2 holds under the assumption of perfectly circular coverage region, which is hardly met in practical scenarios. Yet, recent works [14, 24] support the conjecture that the same asymptotic result on the value of k holds also in a cost-based connection model, which is shown in [24] to closely resemble log-normal shadowing propagation (i.e., irregular coverage regions).

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 NTC-PL and NTC-PLU 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 NTC-PL protocol are connected. Note that, in general, the graphs generated by NTC-PL and NTC-PLU are different, so different values of k could be used. We have verified through our experiments that the graphs generated by NTC-PLU are relatively less connected than those generated by NTC-PL with the same value of k . However, with the value of k chosen (which guarantees at least 98% of connectivity with NTC-PL), also the graphs generated by NTC-PLU show good connectivity on the average. For this reason, in the simulations reported in Section 8, we have used the same value of k in both protocols.

Table 1. 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

The ideal values of k for different values of n are reported in Table 1. 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. 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 NTC-PL and NTC-PLU with respect to the minimum value of k required.

7 Protocol Variation for Dynamic Networks

In this section, we present a protocol variation for dynamic networks, which handles mobility, failures, and dynamic joining of nodes. The principal complicating factor in dealing with dynamic networks 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, $N_s(u)$ is not accurate. One must also be careful not to adjust power levels too quickly when topology changes occur, lest the protocol exhibit unstable behavior.

Based on the discussion above, our version of NTC-PL for dynamic networks 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} . These basic ideas are sufficient to deal with all sources of dynamism, which include mobility, node failures, and node joins. Note also that these mechanisms no longer rely on the assumption of a symmetric wireless medium and, hence, we remove that assumption for this section.

The details of our procedure for estimating N_s are given in Figure 3. When a node is first powered up, it initiates this procedure, which is described next. 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 .

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 4 shows the procedure for increasing $|N_s|$, while Figure 5 shows how a decrease in $|N_s|$ is achieved.

There are two main differences in how $|N_s|$ is increased in the dynamic case (Figure 4) compared to the static version of NTC-PL. 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 in the dynamic case, whereas in the static 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.

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  $\text{Timer}_v = 0$  then set  $\text{Timer}_v$  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  $\text{Timer}_v$  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. 3. Procedure for estimating N_s performed by node u

increase_neighbors()

```

while  $(|N_s| < k_{\text{low}})$  and  $(l < \text{max})$  do
  count  $\leftarrow 0$ 
  while  $(|N_s| < k_{\text{low}})$  and  $(\text{count} < l)$  do
    send help message  $(u, l, N_i)$ 
    wait  $T_l$ 
    count  $\leftarrow \text{count} + 1$ 
  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. 4. Procedure for increasing N_s performed by node u

This is the reason that help requesters send help messages multiple times at the same power level.

While re-sending help messages at the same power level might seem inefficient, the simulation results of Section 8.3 demonstrate that the message overhead of the dynamic protocol is extremely low. This is due to the use of low and high water marks on k , which are quite effective at limiting the frequency of protocol execution, making minor inefficiencies during protocol execution much less important. The primary motivation behind repetitive help messages at the same power level is to avoid the following scenario, which can occur in networks with mobility. 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 NTC-PL 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 4, while using more control messages and time, produces more graceful changes in power levels and avoids unnecessary large increases in power by many nodes.

The need to decrease the number of neighbors (Figure 5) arises with mobility and/or dynamic node joins, because the basic NTC-PL protocol ensures that nodes' power levels are set as small as possible. For example, 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

```

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  $l \leftarrow l - 1$ 
  if  $|N_s| < k_{\text{low}}$  then  $l \leftarrow l + 1$ 

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. 5. Procedure for decreasing N_s performed by node u

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} . The main considerations are as follows. k_{low} should be set high enough to maintain the desired connectivity property. For a given k_{low} , k_{high} determines the width of the allowable k range, which influences protocol stability and message overhead. If the k range is quite wide, the protocol will not be triggered often and it will have low overhead and exhibit stable behavior. On the other hand, a wide k range, will produce a higher average k , which means higher node degrees and energy costs. Thus, k_{high} should be set to the smallest value that maintains stable protocol behavior and reasonable overhead. In Section 8.3, we carefully evaluate the choices of these parameters through simulation.

Similar to the dynamic version of our protocol, the LINT protocol of [21] tries to maintain the symmetric neighbor set size between low and high water marks. However, LINT suffers from the same problem pointed out earlier with the MobileGrid protocol of [18]. In LINT, nodes simply increase their transmission range when the number of neighbors is too low. The problem is that increasing a node's transmission range is not guaranteed to increase its neighbor set size and might even lower it. LINT does not do the type of local coordination that is part of our protocols and is necessary to ensure that actions taken by nodes have the desired effect (increasing or decreasing the neighbor set size). Thus, *our dynamic protocol is the first that can guarantee a lower bound on the symmetric neighbor set size of a node in a dynamic environment*. Furthermore, it includes the ability to decrease neighbor set sizes and to ensure that larger-than-necessary transmission power adjustments do not occur, two features that are not necessary in the static version of the protocol.

8 Simulations

In this section, we report the result of simulations we have performed to evaluate the performances of our protocols, both on static networks and on dynamic networks.

The performances of the various protocols are compared with respect to the following metrics:

- total *energy cost*, defined as the sum of the power levels of all nodes: at the end of protocol execution for the static protocols and as a function of time for the dynamic protocol
- average *logical* and *physical* node degrees. The logical degree of node u is its degree in the communication graph, while the physical degree is the number of nodes in the radio coverage area of u . Due to the removal of asymmetric links and to optimizations, the physical degree is usually larger than the logical degree.
- the average number of messages per node : sent during phase 1⁸ for the static protocols and as a function of time for the dynamic protocol
- for the dynamic protocol only, the average percentage of nodes that are in the largest connected component (LCC) of the communication graph

⁸ We recall that phase 2 requires one further message per node sent in both protocols.

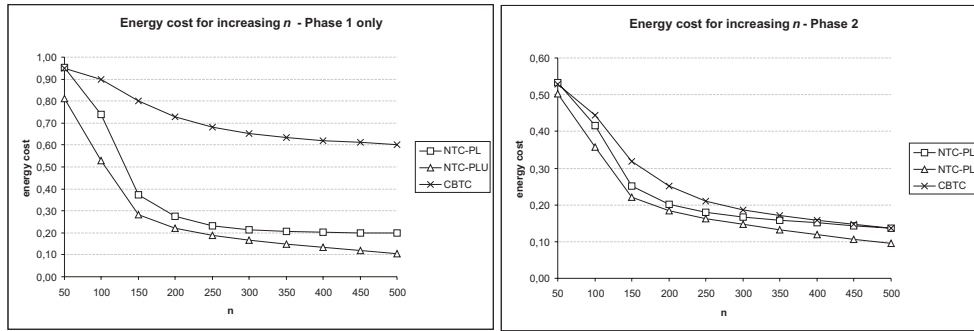


Fig. 6. Energy cost of the NTC-PL, NTC-PLU and CBTC protocols as the network size increases, before (left) and after (right) optimization. The energy cost is normalized with respect to the case of no topology control, where all the nodes transmit at maximum power.

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 [9]. The average LCC size is used to evaluate network connectivity in the dynamic case. Note that no protocol can guarantee the network is fully connected at all times in that case. Thus, we strive to maintain as many nodes as possible in the largest connected component.

8.1 Simulation results for static networks: minimum density

Besides the NTC-PL and NTC-PLU protocols, we have evaluated the performance of the CBTC protocol of [27], which is the best known static topology control protocol. We have adapted CBTC to take into account the transmission power level actually available; i.e., the transmission power level of any node at the end of CBTC execution is rounded up to the next power level available.

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, 27] for details on the optimization phase). The optimization phase of the various protocols is called phase 2 in the following.

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 transmission power levels specified in the data sheets of the Cisco Aironet 350 card [7], namely 1mW, 5 mW, 20mW, 30mW, 50mW, and 100mW. As reported in the data sheets, the transmission 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 transmission range at the other power levels, which are 173m, 134m, 109m, 55m and 24m, respectively. This setting of the transmission range resembles a simple free space wireless channel model.

We have considered a simulation area of 1 square kilometer. According to the data reported in [23], 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.

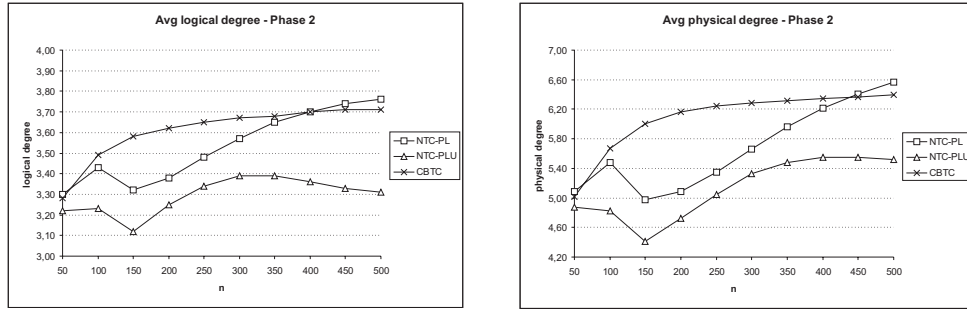


Fig. 7. Average logical (left) and physical (right) node degree after the optimization phase.

The results of this set of simulations are reported in Figures 6–8, and are averaged over 1000 runs. From the figures, it is seen that:

- both NTC-PL and NTC-PLU 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% (NTC-PL) and 77% (NTC-PLU).
- 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 NTC-PL 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 NTC-PLU 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 NTC-PL, and as high as 91% for NTC-PLU.
- concerning the logical and physical node degree of the communication graph, NTC-PLU 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 NTC-PLU is used is as much as 26% smaller than that generated by CBTC, and as much as 12% smaller than that generated by NTC-PL. With respect to the case of no topology control, NTC-PLU reduces the average physical node degree by about 75%.
- NTC-PLU always perform better than NTC-PL, with respect to both energy cost and node degree. In terms of communication overhead, NTC-PLU exchanges more

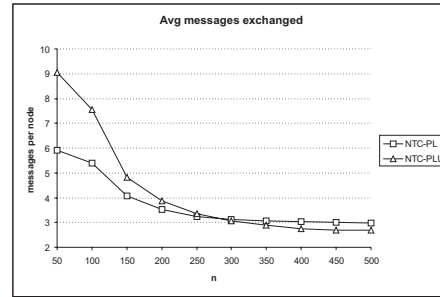


Fig. 8. Average number of message per node sent during phase 1 of the NTC-PL and NTC-PLU protocols.

messages than NTC-PL when the network size is small. However, when the network size increases, the situation is reversed: for $n \geq 300$, NTC-PLU generates fewer messages than NTC-PL. Although, when being executed at a particular node, NTC-PLU generates more messages at a given power level than NTC-PL, NTC-PLU will terminate earlier than NTC-PL in their searches for the proper power level in some cases. For example, a node executing NTC-PLU will terminate its protocol execution when it knows that increasing its own power to a certain level is sufficient to achieve the proper symmetric neighborhood size and that doing so is more efficient than stepping through more power levels and sending help messages at each of those levels. For larger networks, the benefits of this early termination appear to outweigh the higher per level communication costs. Thus, when the network size is large, NTC-PLU performs better than NTC-PL in all respects.

8.2 Simulation results for static networks: 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 herein, are practically identical to those obtained in the minimum density scenario. In other words, for a given value of n , the performance of NTC-PL, NTC-PLU 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 NTC-PL and NTC-PLU, 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.

8.3 Simulation results for dynamic networks

The simulation set-up for dynamic networks is as follows. We focus on mobility as the source of dynamism, because this produces a much more dynamic situation than would typically occur with node failures and joins. The mobility model we consider is the widely-used random waypoint model [11]. We use the same size deployment region (1 square km) and the same transmission power levels as in the static network simulations. The number of nodes is 100. The pause time for the random waypoint model is set to zero, to produce the most dynamic possible network. We consider both a low speed scenario and a high speed scenario. Node velocity is set to 1 m/sec in the low speed case and 15 m/sec in the high speed case. Nodes send beacon messages once every second in the dynamic protocol.

One of the main issues to evaluate is how to set the lower and upper thresholds (k_{low} and k_{high}) on neighborhood size. We carry out two sets of experiments to evaluate this. In the first set of experiments, we gradually increase both k_{low} and the width of the range. We refer to these as the $k=x-2x$ experiments, because we set k_{low} to 5, 6, 7, and 8, and set k_{high} to twice k_{low} in each case. In the second set of experiments, we fix k_{low} to 5 and we narrow the range by gradually reducing k_{high} . We refer to these as the $k=5-x$ experiments.

Figures 9 and 10 show the results for the $k=x-2x$ experiments, at low speeds and high speeds, respectively. We first describe the low speed results (Figure 9). With each of the different ranges of k values, the protocol is able to maintain more than 90% of the nodes in the network within the largest connected component (LCC). The LCC size varies from about 92% for $k=5-10$ up to about 96% for $k=8-16$. The protocol achieves this while still producing substantial energy savings. The energy cost varies from about 50% of the maximum power energy cost for $k=5-10$, up to around 68% of the maximum for $k=8-16$. It can also be seen that having a range of neighborhood sizes within which no protocol execution is triggered is very effective at keeping the overhead of the protocol low. Only around 1.2 messages per second per node are sent by the protocol for each of the neighborhood ranges. In terms of the density of the topologies that are produced, we see that the logical degrees fall in the middle of the neighborhood range, tending slightly toward the lower threshold. There is a steady increase in logical degree as the lower threshold of the neighborhood range is increased. Physical degree follows the same trends, but is slightly higher. Overall, the protocol with a neighborhood range of $k=5-10$ performs quite well, achieving better than 90% of nodes in the LCC with energy cost about 50% of the maximum power setting, a logical degree of 7, and a physical degree of 9, while only exchanging 1.2 messages per second per node.

In looking at the high speed results (Figure 10), we see that the protocol performance is very similar to the low speed results in terms of energy cost, logical degree, and physical degree. The main differences are that there is more variation in the LCC size and the message cost is higher. The variation in LCC size can be attributed to multiple neighborhood changes occurring before the protocol can finish adjusting to the first change. This situation can temporarily cause some local connectivity losses that are repaired with a short delay. These variations are most noticeable for the narrowest neighborhood range ($k=5-10$), and are hardly noticeable for the widest range ($k=8-16$). Since, at a higher speed, more neighborhood changes occur per unit of time, we would

expect message cost to increase. Nevertheless, the cost is still quite low, ranging from 1.5 messages per second up to 1.7 messages per second. Although LCC size is slightly degraded compared to the low speed case, the narrowest range is still able to achieve around 90% of the nodes in the LCC and the wider ranges are still above 90%.

Figures 11 and 12 show the results for the $k=5-x$ experiments, at low speeds and high speeds, respectively. Again, we focus on the low speed results (Figure 11) first. As the neighborhood range becomes narrower, we expect that the protocol will be triggered more often and so the message cost will increase. Also, since the lower threshold of the range remains fixed, as the range becomes narrower, we should see a reduction in logical and physical degree. The results do indeed illustrate these general trends. It is interesting to see that when we narrow the range from $k=5-10$ to $k=5-9$, we achieve almost the same LCC size, so there is little impact on the quality of the topology in terms of connectivity. There are also noticeable benefits in terms of degree: average logical degree and average physical degree are both reduced by 0.5, which is about a 7% reduction. There is a smaller benefit in terms of energy cost (about 4%). These benefits come at a relatively small increase in message cost, from about 1.25 messages per second to about 1.4 messages per second (an 11% increase). This indicates that some narrowing of the neighborhood range is beneficial. However, when the range is narrowed further (the $k=5-8$ and $k=5-7$ cases), the results degrade substantially. The protocol takes a long time to converge to a stable LCC size and the message cost is substantially increased, particularly during the transient phase while the LCC size is converging. While the $k=5-8$ results do eventually reach a good state, the transient period is quite long. We believe this is due to the random waypoint mobility model, which has a steady state node distribution that concentrates most of the nodes in the center of the region. The protocol appears to work well with a narrow neighborhood range once this concentration effect has occurred but poorly prior to that. Since this is just an artifact of the mobility model, we do not recommend using the narrower neighborhood ranges in general settings.

For the high speed case (Figure 12), there is a noticeable drop-off in LCC size when going from $k=5-10$ to $k=5-9$. However, if slightly lower than 90% LCC size can be tolerated, the $k=5-9$ range has similar benefits in terms of degrees and energy to what it achieved in the low speed case. In contrast to the low speed case, we do not see the long convergence times for the $k=5-8$ and $k=5-7$ neighborhood ranges. We attribute this to the faster convergence of the random waypoint model to its steady state node distribution due to the higher speed of the nodes. Both of the narrower neighborhood ranges perform fairly poorly here in terms of LCC size and so are probably not suitable regardless of their better convergence behavior.

9 Discussion

In this chapter, we have presented topology control protocols that use a discrete number of transmission power levels, as opposed to assuming that the power level can be set to an arbitrary value in a given range. The protocols implement a neighborhood based approach to topology control in which a node uses its number k of nearest neighbors to route in/out traffic. We have shown by means of extensive simulations that the protocols are effective in reducing the energy cost, comparing favorably with the well known

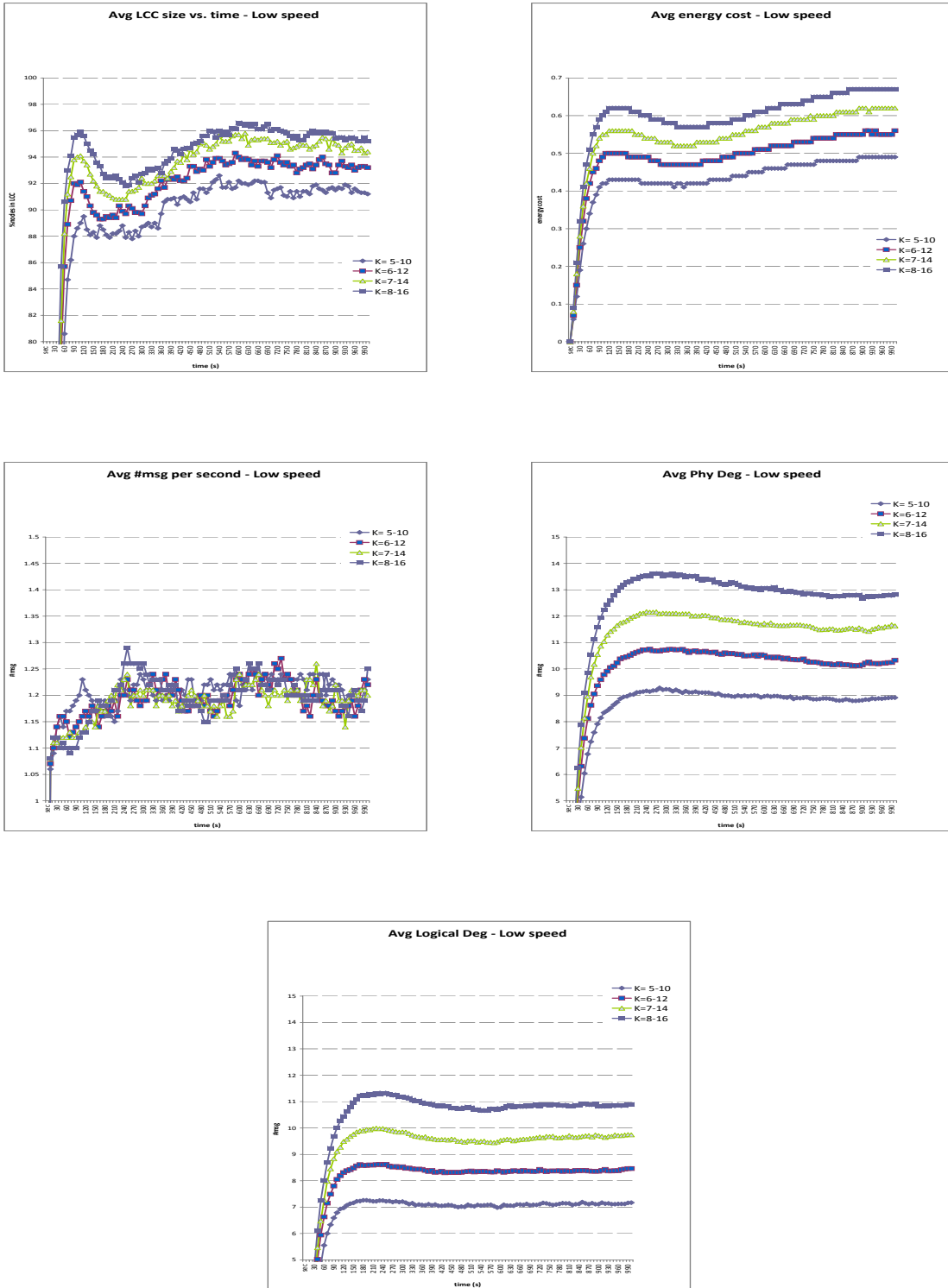


Fig. 9. Dynamic protocol performance: low speed nodes, different neighborhood size ranges

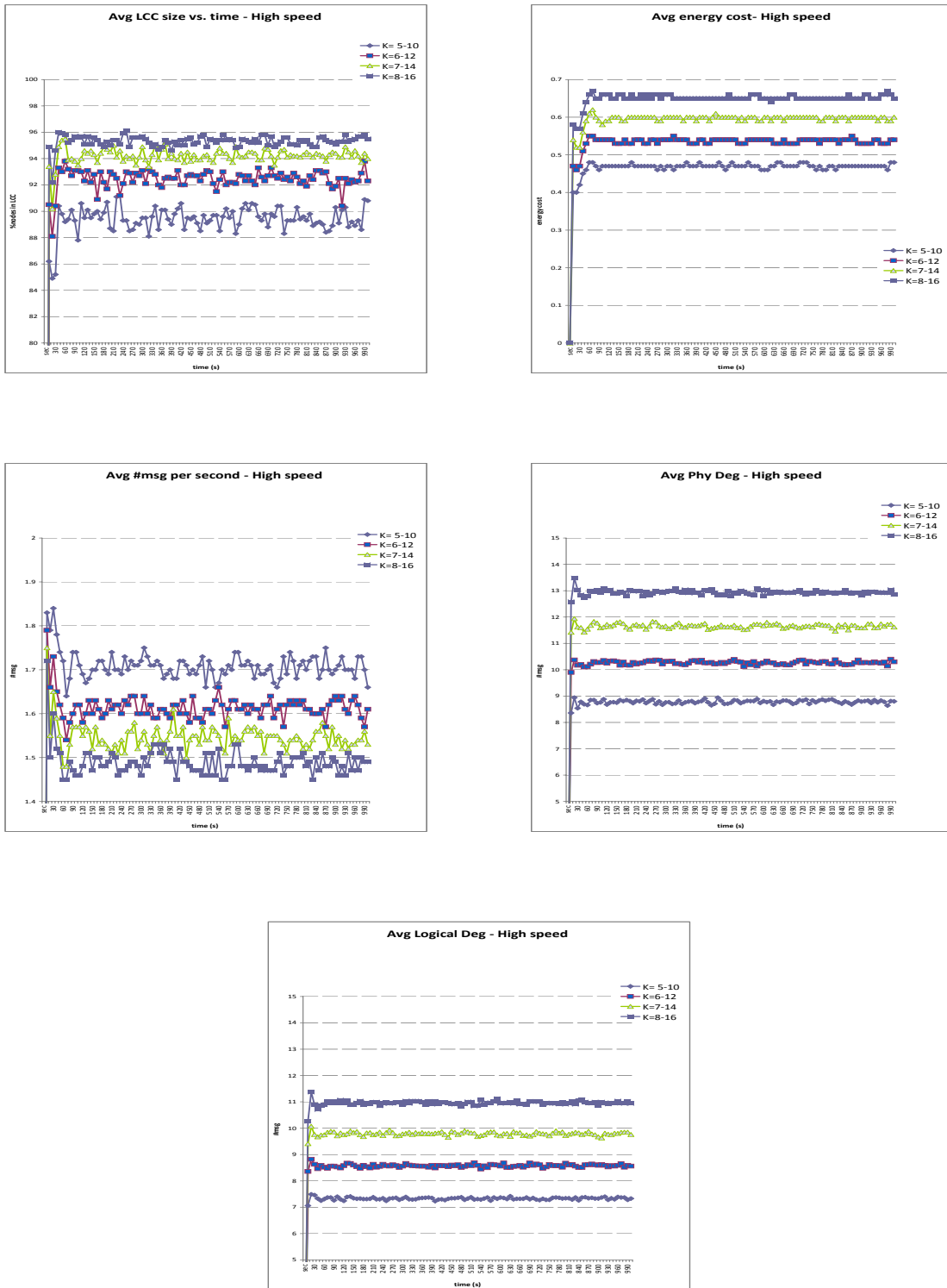


Fig. 10. Dynamic protocol performance: high speed nodes, different neighborhood size ranges

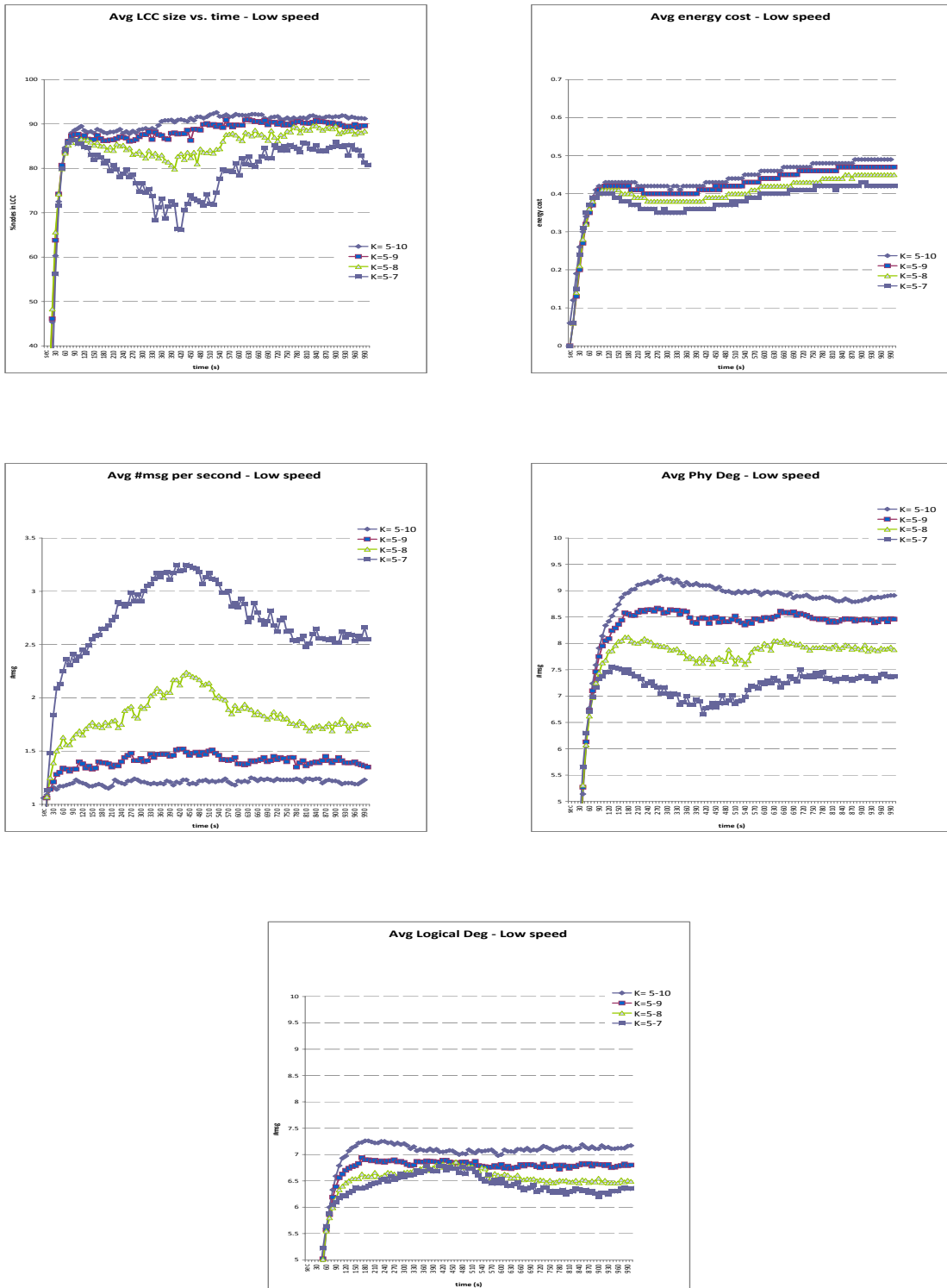


Fig. 11. Dynamic protocol performance: low speed nodes, different neighborhood size upper thresholds



Fig. 12. Dynamic protocol performance: high speed nodes, different neighborhood size upper thresholds

CBTC protocol of Wattenhofer, et al. [27]. We have also proposed and thoroughly evaluated a variation of our base protocol, which is designed for dynamic networks that experience failures, mobility, and other dynamic conditions. Our results show that this dynamic protocol can effectively establish and maintain topologies with low degrees and good connectivity properties in highly dynamic environments, while incurring very low message overheads.

Our protocols rely on appropriate selection of neighborhood size parameters: k in the case of static networks and the range $[k_{\text{low}}, k_{\text{high}}]$ in the dynamic network case. Our simulation results provide a starting point for selection of these parameters in actual networks. We recommend starting with a fairly conservative choice, e.g. $k = 5$ for static networks of medium to large size, $k = 6$ for small static networks, a range of $[5, 10]$ for moderately dynamic networks, and a range of $[6, 12]$ for highly dynamic networks. These parameters can then be gradually adjusted over time to minimize overhead and energy consumption while maintaining desired network properties.

A final issue worth discussion is the discovery that, when nodes coordinate their transmission power decisions, an unselfish topology control protocol performs better than a selfish one. Thus, to optimize the overall topology, it is necessary to ensure that nodes are acting for the common good and not in their own self-interests. Designing schemes which can provide rewards to nodes that faithfully execute an unselfish protocol is an interesting topic for future research.

References

1. M. Bahramgiri, M. Hajiaghayi, and V.S. Mirrokni, "Fault-tolerant and 3-Dimensional Distributed Topology Control Algorithms in Wireless Multi-hop Networks," *Wireless Networks*, Vol. 12, no. 2, pp. 179–188, 2006.
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, "The k -Neighbors Approach to Interference Bounded and Symmetric Topology Control in Ad Hoc Networks," *IEEE Trans. on Mobile Computing*, Vol. 5, no. 9, pp. 1267–1282, 2006.
4. D. M. Blough, M. Leoncini, G. Resta, and P. Santi, "Topology Control with Better Radio Models: Implications for Energy and Multi-Hop Interference," *Performance Evaluation*, Vol. 64, no. 5, pp. 379–398, June 2007.
5. M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger, "Does Topology Control Reduce Interference," *Proc. ACM Mobicom 04*, pp. 9–19, 2004.
6. M. Cardei, S. Yang, and J. Wu, "Fault-Tolerant Topology Control for Heterogeneous Wireless Sensor Networks," *Proc. IEEE Int. Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 1–9, 2007.
7. *Cisco Aironet 350 data sheets*, available at <http://www.cisco.com/en/US/products/hw/wireless>.
8. M. Dyer, J. Beutel, and L. Thiele, "S-XTC: A Signal-Strength Based Topology Control Algorithm for Sensor Networks," *Proc. IEEE Int. Symp. on Autonomous Decentralized Systems (ISADS)*, pp. 508–518, 2007.
9. P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Trans. Information Theory*, Vol. 46, no. 2, pp. 388–404, 2000.

10. Z. Huang, C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Topology Control for Ad Hoc Networks with Directional Antennas," *Proc. IEEE Int. Conference on Computer Communications and Networks*, pp. 16–21, 2002.
11. D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, Kluwer Academic Publishers, pp. 153–181, 1996.
12. V. Kawadia and P.R. Kumar, "Power Control and Clustering in Ad Hoc Networks," in *Proc. IEEE Infocom 03*, 2003.
13. F. Kuhn, N. Lynch, and C. Newport, "The Abstract MAC Layer," *Proceedings of the 23rd International Conference on Distributed Computing*, pp. 48–62, 2009.
14. F. Kuhn, R. Wattenhofer, A. Zollinger, "Ad Hoc Networks beyond Unit Disk Graphs", *Wireless Networks*, Vol. 14, pp. 715–729, 2008.
15. L. Li, J.H. Halpern, P. Bahl, Y. Wang, and R. Wattenhofer, "A Cone-Based Distributed Topology Control Algorithm for Wireless Multi-hop Networks," *IEEE/ACM Trans. on Networking*, Vol. 13, no. 1, 2005.
16. N. Li and J.C. Hou, "FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Sensor Networks," *Proc. ACM Int. Conference on Mobile Computing and Networking (MobiCom)*, pp. 275–286, 2004.
17. X-Y. Li, P-J. Wan, Y. Wang, and C-W. Yi, "Fault-Tolerant Deployment and Topology Control in Wireless Networks," *Proc. ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 117–128, 2003.
18. J. Liu and 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.
19. E. Lloyd, R. Liu, M.V. Marather, R. Ramanathan, and S.S. Ravi, "Algorithmic Aspects of Topology Control Problems for Ad Hoc Networks," *Mobile Networks and Applications*, Vol. 10, n. 1-2, pp. 19-34, 2005.
20. S. Narayanaswamy, V. Kawadia, R.S. Sreenivas, and P.R. Kumar, "Power Control in Ad Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW Protocol," *Proc. European Wireless 2002*, pp. 156–162, 2002.
21. R. Ramanathan and R. Rosales-Hain, "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment," *Proc. IEEE Infocom 2000*, pp. 404–413, 2000.
22. P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks*, John Wiley and Sons, 2005.
23. P. Santi and D.M. Blough, "The Critical Transmitting Range for Connectivity in Sparse Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, Vol. 2, no. 1, pp. 1–15, January-March 2003.
24. C. Scheideler, A. Richa, P. Santi, "An $O(\log n)$ Dominating Set Protocol for Wireless Ad Hoc Networks under the Physical Interference Model", *Proc. ACM MobiHoc*, pp. 91-100, 2008.
25. F. Wang, M.T. Thai, Y. Li, X. Cheng, and D. Du, "Fault-Tolerant Topology Control for All-to-One and One-to-All Communication in Wireless Networks," *IEEE Trans. on Mobile Computing*, Vol. 7, no. 3, pp. 322–331, 2008.
26. Y. Wang, L. Cao, T.A. Dahlberg, F. Li, and X. Shi, "Self-Organizing Fault-Tolerant Topology Control in Large-Scale Three-Dimensional Wireless Networks," *ACM Trans. on Autonomous and Adaptive Systems*, Vol. 4, no. 3, 2009.
27. R. Wattenhofer, L. Li, P. Bahl, and Y. Wang, "Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks," *Proc. IEEE Infocom 2001*, pp. 1388–1397, 2001.
28. R. Wattenhofer and A. Zollinger, "XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks," *Proc. Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2004.

29. F. Xue and P.R. Kumar, "The Number of Neighbors Needed for Connectivity of Wireless Networks," *Wireless Networks*, Vol. 10, pp. 169–181, March 2004.