Localized Protocols for Ad Hoc Clustering and Backbone Formation: A Performance Comparison

Stefano Basagni, *Member*, *IEEE*, Michele Mastrogiovanni, Alessandro Panconesi, and Chiara Petrioli, *Member*, *IEEE*

Abstract—This paper concerns the comparative performance evaluation of protocols for clustering and backbone formation in ad hoc networks characterized by a large number of resource-constrained nodes. Our aim is twofold: We provide the first simulation-based detailed investigation of techniques for clustering and backbone formation that are among the most representative of this area of ad hoc research. Second, we delve into the nature of the selected protocols to assess the effects of the "degree of localization" on their operations, i.e., how being able to execute the protocol based only on local information affects the overall protocol performance. Extensive ns2-based simulation results show that highly localized protocols are rewarded with good performance with respect to all metrics of interest which include protocol duration, energy consumption, message overhead, route length, and backbone size.

Index Terms—Ad hoc networks, wireless sensor networks, clustering, backbone formation.

1 Introduction

This paper provides a thorough simulation-based comparison of some of the most representative solutions for clustering and backbone formation. Such solutions leverage the scalability limits of traditional routing for ad hoc and wireless sensor networks (WSNs) by selecting a fraction of the network nodes, the so-called *backbone*, that is responsible for performing and managing multipoint communication (routing, multicast, broadcast).

A typical approach to backbone formation is to partition the network into *clusters* made up of a *clusterhead* and some *ordinary nodes*. The clusterheads are then joined to form the connected backbone. Backbone formation corresponds to computing a *connected dominating set* (CDS) of the nodes of the *network topology graph*, which is the graph $G = (V, E)$ whose nodes are the network devices and where there is an edge between two nodes if and only if they are *neighbors*, i.e., they are in each other's communication range. A CDS is a connected subgraph of G spanned by the nodes of $V' \subseteq V$ such that every node in the network is distant at most one hop from a node in V'. A good backbone should first and foremost be small; additionally, it should have other characteristics such as robustness to node failures (i.e., hard to disconnect) and low stretch, i.e., routes in the backbone should not be much longer than the shortest routes over the network topology graph.

- *S. Basagni is with the Electrical and Computer Engineering Department, 312 DANA, Northeastern University, 360 Huntington Ave., Boston, MA 02115. E-mail: basagni@ece.neu.edu.*
- *M. Mastrogiovanni, A. Panconesi, and C. Petrioli are with Dipartimento di Informatica, Universitá di Roma "La Sapienza," Via Salaria 113, Rome, Italy. E-mail: {mastrogiovanni, ale, petrioli}@di.uniroma1.it.*

Manuscript received 22 Feb. 2005; revised 21 May 2005; accepted 9 June 2005; published online 24 Feb. 2006.
Recommended for acceptance by I. Stojmenovic, S. Olariu, and D. Simplot-Ryl.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0176-0205.

The "CDS approach" has received a lot of attention and a plethora of different algorithms for computing CDSs have appeared in the literature (a survey of these algorithms is provided in the Appendix which can be found on the CS Digital Library at http://www.computer.org/tpds/archives.htm). Surprisingly, there seems to be no systematic experimental evaluation of the many different approaches. This paper fills this gap by providing a thorough, ns2-based comparative analysis of some of the most interesting solutions that have been proposed in the literature. In our simulations, we take into account characteristics of the network physical and MAC layers, and we make realistic implementation decisions at the network layer.

We consider three major classes of protocols that we distinguish based on their *degree of localization*. The degree of localization of a protocol is a measure of how much the decision about a node being part of the backbone or not depends on nodes that are possibly several hops away from it. Protocols like the one presented by Wu and Li in [11] and enhanced in [4] (referred to in the sequel as WuLi after the name of their inventors) and the one presented in [1] and further developed in [10] that compute *multipoint relay* (MPR) sets rely only on very local information. To decide whether to become part of the backbone or not, a node only has to know its 2-hop neighborhood. Protocols in this group are not only very localized, but computationally very simple, an attractive feature that, in the resource constrained environment of ad hoc and sensor networks, is bound to translate into very efficient protocols. As we shall see, an optimized version of WuLi based on [8] performs particularly well. Algorithms in this class basically first compute a large CDS and then try to prune away nodes by means of local optimizations.

We then consider protocols that use a different approach: They first compute a small dominating set, and then connect it up by inserting new nodes. For the first step, one can use either the *Distributed Clustering Algorithm*

(DCA) [2] or the randomized implementation of the well-known greedy heuristic for set cover [6] (*Local Randomized Greedy* or LRG). Both algorithms compute provably small dominating sets not much larger than the optimum, computing which is NP-hard and, hence, (presumably) infeasible in principle. These algorithms are very fast, but not as local as the ones described earlier because a node might have to wait for information coming from parts of the network outside of its immediate 2-hop neighborhood before it can make a decision on its own. Once a dominating set C is computed, in order to obtain a connected backbone, it is sufficient to join the nodes in C that are two and three hops away [3]. This is very simple and local, but may result in large backbones. A way out is the sophisticated cycle-breaking technique of [5] which prunes redundant nodes away from the backbone, while maintaining connectivity. When paired with DCA and LRG to produce their sparsified versions DCA-S and LRG-S, this approach produces backbones that are small and yet robust, all at a very reasonable cost. When backbone size is at a premium, this might well be the best solution; for most scenarios the very local approach described earlier, which is more economical, gives the protocol of choice.

Finally, we considered the WAF protocol (so named here after the initials of the authors' last names) presented in [9]. This protocol uses the approach of computing a small dominating set and connecting it up. The second step is done in the best possible way, i.e., by means of a spanning tree. While this approach computes the smallest CDSs among all protocols we tested, it is infeasible because of its huge time and communication costs. This is not surprising since WAF is based on two processes that are inherently sequential: Leader election and spanning tree computation. In order to make decisions, nodes have to wait for information coming from far away in the network, i.e., this approach has an extremely low degree of localization. This protocol, however, retains its utility as a benchmark, for it produces the smallest backbones. We compared it with the two best solutions emerging from the two groups above to assess the best trade-offs between backbone quality and computation cost.

The rest of the paper is organized as follows: Section 2 describes the selected protocols together with their extensions. We describe our simulation environment in detail and we evaluate and discuss the performance of the protocols with respect to the selected metrics of interest in Section 3. Section 4 concludes the paper.

2 Protocols Description and Implementation

Our comparative analysis focuses on the following protocols: WuLi, MPR (with their extensions), DCA and DCA-S, LRG and LRG-S, and, finally, WAF. As mentioned, these are all distributed and localized protocols for computing a backbone which is nothing but a connected dominating set (CDS). The algorithms are quite different, especially concerning the degree of localization they exhibit. In the following description, and in the performance evaluation section that follows, this important aspect of the considered protocols has been the leading point of view for their presentation and comparison.

WuLi [11] is a very simple distributed procedure consisting of a few local rules, the execution of which creates the desired CDS. The simplest rule is the following: If a node v has two neighbors that are not neighbors themselves, then v enters a set C. It is straightforward that this rule eventually generates a set C which is a CDS (provided that the network is not a clique, a case that can be dealt with separately). The algorithm is extremely simple and efficient but, not surprisingly, tends to create very large CDSs. Therefore, the authors discuss other local rules whose aim is to prune away unnecessary nodes.

The original protocol presents two pruning rules. Rule 1: For every pair of nodes u and v in C, the one with the smaller ID, say v, can be removed from C if v and all its neighbors are covered by u (i.e., they are u's neighbors). Rule 2: Assume nodes u, v, and w are in C, and assume that v's ID is the smallest of the three. Assume also that u and w are neighbors of v and are in each other's transmission range. If each neighbor of v is covered by u or w, then v can be removed from C.

The version of the WuLi protocol that we consider in this paper is the one presented in [4] where Rule 1 and Rule 2 are generalized into the following Rule k: Assume that node $u \in C$ and that a subset S of $\leq k$ neighbors of node u is such that

1. the subgraph spanned by S is connected,
2. S is contained in C,
3. each node in S has an ID larger than u, and
4. each neighbor of u is covered by the nodes in S.

Then, u can be removed from C.

Being the most general, this last rule prunes away the largest number of nodes. In the following, we will denote with WuLi(i) this version of the WuLi protocol where $k = i$.

We implemented WuLi in the following way: Every node exchanges its neighbor list with its own neighbors. Based on this simple and local exchange of information, a node can decide whether it is in C or not (high degree of localization) and communicates this information to its neighbors. Once it is aware of which of its neighbors belong to C, a node u in C locally executes Rule k to decide whether it will be included in the final backbone or not.

From an (ns2) implementation point of view, this protocol uses three messages. The first is for generating the rich CDS, and the following two are used for making the neighbors aware of its membership in C and its final decision. All messages are broadcast by each node to all of its neighbors. Each transmission has an associated timer. If the timer expires and node u has not received the corresponding message from all its neighbors, u sends a unicast message to all those neighbors that did not reply, asking for retransmission. The first message is resent in broadcast (again), while the last two, more critical, are retransmitted using a unicast packet. Time-outs have a fixed length (0.7s for most of the implementations), which is increased with a random jitter (uniformly chosen in $[0, 0.3]$s) to decrease the likelihood of collisions. (These values have been chosen also for the implementation of the other protocols.)

Stojmenovic et al. [8] propose an effective strategy for improving the performance of the WuLi protocol. The idea is that there is no need for a node in C to exchange its

membership information with its neighbors. Each node can locally determine whether it belongs to C or not. If this is the case, Rule k can be applied to the entire two-hop "neighbor subgraph" independently of whether the neighbors belong to C or not (as nodes do not communicate such decisions, there is no way for a node to know which of its neighbors are in C). This idea spares a node of exchanging a message (the second message in the implementation of WuLi) without affecting the "pruning power" of Rule k. Moreover, the nodes are now sorted according to their degree rather than their IDs, now used only for breaking possible ties. Using degrees affects the sparsification efficacy of Rule k positively. The "Stojmenovic" implementation of the WuLi algorithm uses only two messages. The first allows the nodes to collect information about their two-hop neighborhood and the second is used by a node for informing its neighbors about its final decision.

A high degree of localization is also shown by the simple and elegant protocol proposed in [1]: The *multipoint relays*-based construction of a CDS, termed MPR. The basic idea is to have each node building a local CDS of the subgraph induced by its two-hop neighbors. Nodes in the local CDS form the sets of multipoint relays. Finding the multipoint relays is quite fast, given the evident high degree of localization of the corresponding protocol. There is an initial exchange of messages via which a node is made aware of its two-hop neighborhood. After this phase, a node u locally selects a set C_u of its neighbors as its multipoint relays by using the following simple greedy algorithm. Step 1: A neighbor v of u is inserted in C_u if there is a two-hop neighbor of u covered only by v. Step 2: A neighbor v of u is inserted in C_u if v covers the largest number of uncovered two-hop neighbors of u. This last step is iterated till all two-hop neighbors are covered.

It is clear that node u only has to wait for the information about its two-hop neighbors. The set C of all C_us is a dominating set of the entire network: A node belongs to C if it belongs to one of the C_us. As for the WuLi protocol, C can be pretty rich. Furthermore, it is not necessarily connected. Therefore, the authors define a couple of rules to ensure connectivity and to prune away redundant nodes. Rule a stipulates that a node u enters the final backbone B if it has the smallest ID among all its (one-hop) neighbors. Rule b states that v enters B also if it is a multipoint relay of its neighbor with the smallest ID. The node terminates the protocol by communicating its final decision to all its neighbors. The resulting set B is proven to be a CDS, as required. We implemented this protocol by using three messages: The first is for exchanging the list of neighbors (broadcast), the second is used by a node u to communicate to a neighbor v for which u is the neighbor with the smallest ID that $v \in C_u$ (unicast), and the final one is used by a node to make every neighbor aware of its final decision (broadcast).

Wu [10] has noticed that some nodes selected by Rule a are not essential for a CDS. Moreover, the basic greedy algorithm does not take advantage of Rule b. Hence, in [10], two enhancements to the MPR original scheme are proposed for obtaining a smaller CDS. The first enhancement concerns Rule a, which now states that a node u decides to stay in C if it has the smallest ID among all its neighbors *and* it has two unconnected neighbors. This extended Rule a and the original Rule b generate a CDS for all possible networks except those whose topology graph is a clique (a case, again, that can be easily dealt with separately). The second enhancement pertains to the formation of C_u which now includes all of u's neighbors for which u is *not* the neighbor with the smallest ID (called *free neighbors*). If the selection of the free neighbors still leaves some of the nodes in the two-hop neighborhood uncovered, additional nodes are added to C_u according to the MPR greedy rule. We observe that MPR-E (the extended version of MPR as proposed by Wu) generates a smaller CDS (and, hence, a better backbone) than the original MPR without imposing any additional communication overhead.

We finally describe those protocols that show a lower degree of localization, namely, DCA, LRG, and WAF and we introduce the extensions to DCA and LRG, DCA-S and LRG-S.

The basic approach to building a CDS followed by these three protocols is 1) compute a (hopefully) small dominating set and 2) connect it.

For step 1) there are two strategies. One is to give a distributed implementation of the best algorithm (in terms of size) for dominating set computation. This is the well-known greedy heuristic for set cover. This greedy strategy repeatedly selects the node that dominates the highest number of nondominated nodes, puts it into the solution, and proceeds. It is well-known that this simple heuristic computes a dominating set whose size is $O(\log n)$ times the optimum value. This is the best possible approximation that can be obtained in polynomial time, unless $P = NP$.

A distributed implementation of this heuristic has been given in [7]. Note, however, that this implementation exhibits a very low degree of localization since it is essentially a straightforward distributed implementation of the sequential process described above. A "more parallel," hence, much faster, implementation of the greedy strategy is a nontrivial task and it is based on the use of randomness. In this paper, we implement the randomized greedy heuristic introduced in [6]: *Local Randomized Greedy*, or LRG. The algorithm proceeds in rounds. The computation and communication in each round is the following.

1. Each node computes the highest span in its two-hop neighborhood. A node span is defined as the number of its neighbors currently uncovered. This step requires a node to transmit two messages: One for communicating its span and the second for broadcasting the bigger span in its neighborhood.
2. A node u is a candidate to the CDS C if its span $d(u)$ is $\geq d(v)$ for each v that is in u's two-hop neighborhood. If u is a candidate, it broadcasts a message to all its neighbors.
3. Based on this last message, every uncovered node v computes its support, which is the number of candidates that cover v (it may include itself). This number is sent to all neighboring candidates with another message.
4. Finally, each candidate u decides whether to enter C or not with probability $1/med(u)$, where $med(u)$ is the average support of all its uncovered neighbors.

The decision is broadcast to all u's neighbors. A node which is still uncovered, or that has neighbors that are still uncovered, starts the process again in the next round.

The authors prove that LRG produces a CDS whose size is within $O(\log n)$ from the optimum in poly-logarithmic time with high probability. The time bound is tight, since a network is provided in the paper where LRG takes at least a poly-logarithmic number of rounds to terminate (with high probability). Not surprisingly, this network is organized in layers. This organization induces the chain of dependency that limits LRG's degree of localization. From an implementation perspective, the fact that LRG requires quite a number of messages per round leads to a high message and energy overhead even if the algorithm takes only few rounds to complete successfully.

A different approach to step 1 is to compute a maximal independent set of the network nodes. An *independent* set is a collection of nodes such that no two of them are neighbors. If an independent set is maximal, then it is also a dominating set. This is the approach taken by both the DCA and WAF protocols. Note that a maximal independent set, in general, tends to be large. In this paper, we use the model of *Unit Disk Graphs* (UDGs), commonly used for ad hoc networks. Two nodes are neighbors if and only if their Euclidean distance is smaller than or equal to their (common) transmission range. For this kind of graph, the size of a maximal independent set is at most five times that of a smallest dominating set, and it is therefore a good approximation of the optimum.

In the DCA protocol [2], all nodes are initially UNDECIDED and have an associated weight. Node weights induce a total ordering of the nodes (ties are broken by using IDs). During the execution of the algorithm, each node will decide to be IN or to be OUT (of the independent set). A node decides when all its neighbors with larger weight have decided. When the time comes, a node decides to be OUT if one of its neighbors is IN. Otherwise, it decides to be IN. (This simple scheme can be somewhat optimized by letting nodes to be OUT as soon as a neighbor is IN). The IN nodes form a maximal independent set, i.e., the minimal dominating set required for clustering. The execution time depends on possible chains of dependency between the nodes, which is why DCA exhibits a degree of localization lower than that of WuLi and MPR. The worst case arises when the network topology is a chain of nodes whose weights are sorted in decreasing order: The node with the smallest weight has to wait for all other nodes in the chain.

Once the dominating set (or independent set) is computed, the task is to connect it to form a backbone (Step 2). The approaches used by DCA and WAF are of opposite nature. Let $G = (V, E)$ be our network and let $D \subseteq V$ be the independent set computed in Step 1 by DCA. To connect D consider the following *auxiliary graph* H with node set D. Two vertices u and v in D are connected by an edge if their distance in G is at most 3. It can be proven that H is connected [3]. Now, every edge uv of H corresponds to a path p_{uv} of length at most 3 in G, i.e., a path with at most two vertices. Let $P(uv)$ denote the vertices of p_{uv}. To obtain a CDS C from D, simply add all such vertices to C, i.e., $C = D \cup (\bigcup_{uv \in H} P(uv))$. In a synchronous, distributed setting computing C in this fashion takes constant-time, and even in an asynchronous environment, we expect the computation and communication cost to be low. Nodes in D simply need to gather two-hop neighbors information. Specifically, they have to be made aware of which neighbors of their neighbors are in D or are dominated by a node in D. Based on this information every node in D selects gateways to interconnect with other nodes in D which are at most three hops away. The selected gateways are part of the backbone. This rule is also used to connect the nodes in the dominating set generated by LRG into a CDS.

The second approach to connect D is much more economical in terms of size, but it is computationally expensive. This is the approach followed by WAF [9]. The idea is to connect D by computing a spanning tree T of H and to augment D with those vertices that correspond to edges of T. That is, $C = D \cup (\bigcup_{uv \in T} P(uv))$. This protocol is implemented by first electing a leader r among the nodes, which is going to be the root of T. Then, a tree is constructed via an r-started flooding of a tree construction message m. While m travels through the network, each node selects as its parent in the tree the node from which it received m first. Each node also computes its rank, defined as the pair (level, node ID). When such a process is completed, the root enters the maximum independent set and starts a color-marking process of the nodes, which proceeds layer by layer, to construct the maximal independent set based on the tree. Nodes in the maximal independent set are then connected through the dominating tree T.

Distributed leader election is *enormously* expensive in practice and exhibits a very low degree of localization. Computing a spanning tree or electing a leader are highly sequential tasks that require coordination and, hence, message exchanges, between far away nodes in the network. The solution adopted in WAF for leader election, for example, requires nodes to be progressively joined to form a connected *fragment* with a leader. At the beginning of the algorithm, each node is an isolated fragment of which it is the leader. As leader election progresses, adjacent fragments merge into one, and the leader of the bigger fragment is selected to lead the newly formed one. Every time two fragments merge into one, the information on the new fragment size and the ID of its leader needs to be propagated to the fragment members and to the nodes in adjacent fragments. For this reason, although these algorithms are quite good in terms of CDS size, our simulation confirmed the expectation that they are extremely expensive in terms of time and communication cost.

The last approach to Step 2 that we present here tries to find a compromise between the described extremes. The algorithm described in [5] finds this compromise by deviating significantly from previous approaches. The basic idea is the following: In order to connect D, find a subgraph S of H that is connected and sparse. By "sparse," we mean that $|S| = O(|D|)$. To clarify, a spanning tree connects D up with $|D| - 1$ edges, but we would be satisfied to do the same with, say, $5|D|$ edges. To create S, the authors make use of an old lemma of the great late mathematician Paul Erdös. Roughly, this lemma says that if a graph does not have small cycles then it must have few edges. Therefore, to sparsify H, it suffices to destroy all small cycles contained in

it. The lemma by Erdös ensures that by destroying all cycles of length $O(\log |H|)$, the remaining graph will have $O(|H|)$ many edges (a cycle is destroyed if at least one of its edges is deleted). The crux is how to destroy all small cycles while maintaining connectivity. In a distributed fashion, this can be done by the following, amazingly simple procedure. Let us call a cycle *short* if it has length $c \log n$, for some fixed c (in practice, $c = 2$), and assume that each edge has a unique identifier. Then, the algorithm is simply this: The lowest ID edge of every short cycle is deleted. Clearly, this destroys all short cycles. The perhaps surprising thing is that it leaves a connected subgraph S of size $O(|D|)$ that connects all vertices of D. The resulting CDS is then $C = D \cup (\bigcup_{uv \in S} P(uv))$. This sparsification procedure can be implemented in a completely asynchronous fashion. The communication cost is proportional to the maximum length of the cycles to be broken, since each edge needs to know if it is the smallest ID edge in a small cycle. We termed the DCA protocol enhanced with this "sparsificator" the DCA-S protocol. More specifically, we call DCA-S(i) the protocol obtained by applying the sparsificator to the DCA backbone, where i is the maximum length of the cycles to be broken. LRG-S and LRG-S(i) are obtained from LRG in a similar way.

3 Protocols Comparison

We performed a thorough simulation-based performance evaluation of the protocols described in Section 2. The selected groups of protocols have been implemented in the VINT project network simulator (ns2) and their performance has been compared to analyze the costs associated to clustering and backbone formation, as well as to assess the properties of the resulting backbone. In particular, we have considered the following metrics (all averages):

1. **Protocol duration**, i.e., the time needed by each protocol to complete clusterhead selection and backbone formation.
2. **The overhead per node** (in bytes) associated to the protocol operations. (These are physical layer measurements, which account for collisions and for the corresponding automatic packet retransmissions at the MAC layer.)
3. **Energy consumption per node**. This metric is important for assessing the application of clustering and backbone formation to networks whose nodes are energy constrained. If the energy and the overhead cost of maintaining a backbone is high, then clustering and backbone reorganization may impose a non-negligible burden to the network. On the other hand, clustering reorganization in terms of clusterhead and gateway rotation is needed to prevent nodes with more demanding roles from depleting their energy, possibly impairing network operations.
4. **Backbone size**, i.e., the fraction of the network nodes that are in the backbone (percentage). A smaller backbone is desirable for minimizing the routing overhead. In sensor networks with nodes that can turn off their radio interface (energy saving sleep mode), the backbone size is also a measure of how many nodes need to stay awake for data transport. Usually, the nodes in the backbone stay up or have a higher duty cycle for guaranteeing routes to the sink and, hence, the smaller the backbone, the more nodes can be in energy conserving mode.
5. **Route length** over the subgraph G_b of the network topology graph induced by the backbone construction. (G_b is the graph where there are no links between ordinary nodes). This metric gives a measure of how well a routing protocol can perform over the backbone.
6. **Backbone robustness**, defined as the number of backbone nodes whose removal (because of failure or energy depletion), causes backbone disconnection or the uncovering of ordinary nodes. This is the metric that provides an indirect measure on how long the network will be operational before requiring a backbone recomputation.

3.1 Simulation Environment

Our ns2 implementation is based on the CMU wireless extension, i.e., on the IEEE 802.11 MAC with the DCF whose parameters have been modified to take sensor nodes characteristics into account (e.g., shorter transmission radius and different values for the energy model).

The simulations refer to scenarios in which n static wireless nodes with a maximum transmission radius of 30 meters are randomly and uniformly scattered in a geographic square area of side L. We make the assumption that two nodes are neighbors if and only if their Euclidean distance is ≤ 30m. We call *network topology graphs* the topologies generated by drawing an edge between each pair of devices that are neighbors. Each device has an initial (residual) energy of 1J. The power consumed while transmitting, receiving, and while in asleep mode are equal to 24mW, 14.4mW, and 0.015mW, respectively. These values are from the actual energy model of the sensor prototypes developed within the IST Energy Efficient Sensor Networks (EYES) project. The power consumed while in the idle state has been set equal to the power consumed while asleep. This corresponds to comparing the protocol's performance under an ideal awake/asleep schedule of the nodes that keeps them up only when needed.

In our simulations, the number of nodes n has been assigned the values 50, 100, 150, 200, 250, and 300, while L has been set to 200m. This allows us to test our protocol on increasingly dense networks, from (moderately) sparse networks with average degree equal to 3.5 to dense networks ($n = 300$) where the average degree is around 20.

All the sensors start the protocol at the same time and run the clustering and backbone formation algorithms to form a hierarchical multihop ad hoc network. All our results achieve a 95 percent statistical confidence with 5 percent precision.

3.2 Experiments

According to the experiments that we performed, the rest of this section is organized into three major parts. We started by comparing the performance of the considered variants of the WuLi and MPR protocols. We have investigated: 1) The general Rule k to derive a CDS as proposed in [4]. In

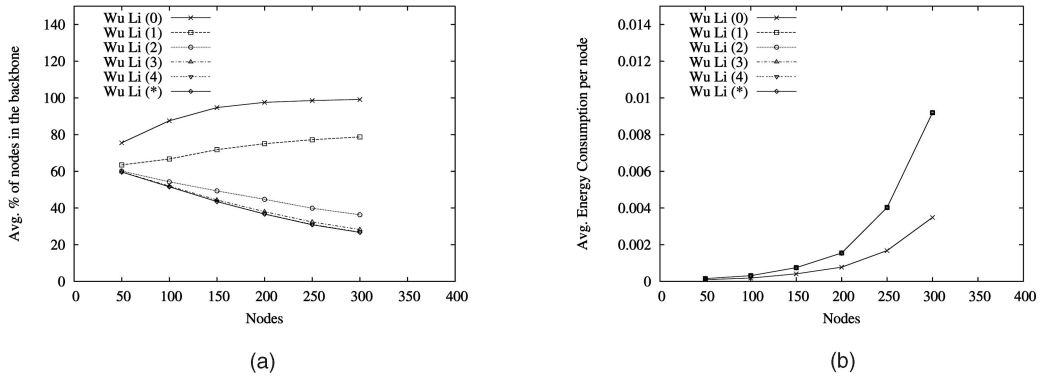


Fig. 1. Average backbone size and energy consumption as functions of k. (a) Backbone size (percent of n). (b) Energy consumption (per node).

particular, we have analyzed the effects of the parameter k. 2) The impact on the backbone metrics of the different rules for computing the nodal weight, namely, when the weights equal the node IDs, or when they are based on the nodal degree as suggested by Stojmenovic in [8]. 3) The pros and cons of the different variants of the WuLi protocol proposed in [11], [4], and [8], as well as of the variants of the MPR protocol presented in [1] and improved in [10].

This first comparative performance evaluation has the threefold aim of identifying the best performing protocols in this class, of showing the rationale behind the different ideas they are based on and behind their different performance, and of assessing the advantages of using protocols with high localization degree.

The second batch of simulations concerns the performance of those protocols with a lower degree of localization, such as DCA [2], LRG [6], and their sparsified variants described above. Our experiments aim at discovering which of the two approaches would result in backbones with a smaller number of nodes and shorter routes and in better performance with respect to all the other relevant metrics. We also investigate the sparsified version of both DCA and LRG, namely, DCA-S and LRG-S, respectively. The aim in this case is assessing the effectiveness of the sparsification rule in producing slimmer connected backbones and to determine the corresponding overhead in terms of energy, transmitted bytes, and time. Finally, we study the performance trade-offs associated to different tunings of key parameters of the sparsification technique. As before, based on this comparison, we identified the best performing algorithm in this second class of protocols according to the selected metrics.

The third and final set of simulations compares the elected "champion protocols" from the first two sets with the WAF backbone formation protocol [9]. WAF, as mentioned, shows a low degree of localization, given that, before deciding whether they will be part of the backbone or not, nodes must wait for the election of a leader and the following tree construction. As mentioned earlier, despite this longer set up time and a higher message exchange, WAF forms extremely slim, tree-like backbones, hence, our interest in comparing this solution with the best ones from the other classes.

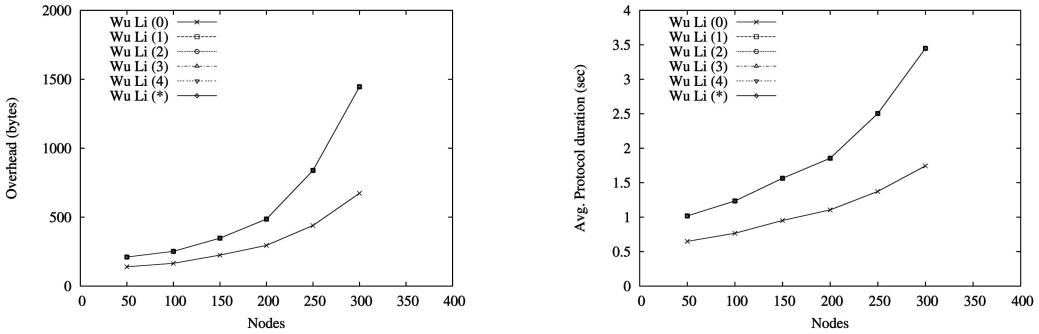
This final comparison allows us to clearly picture the impact of the different degrees of localization showed by the three different classes of protocols we considered. We observe that the higher the degree of localization, the simpler the protocols, which leads to lower complexity (in term of messages/byte exchanged and time) without compromising the capability to generate slim backbones with acceptably short routes. Therefore, a high degree of localization is a key element in the design of effective protocols for ad hoc clustering and backbone formation.

3.2.1 Protocols with a High Degree of Localization: WuLi and MPR

Our first set of experiments aims at evaluating the general Rule k proposed by Dai and Wu in [4]. In particular, we have studied the impact of the parameter k on the performance of the WuLi protocol. Results are depicted in Fig. 1, Fig. 2, and Fig. 3.

Fig. 1a depicts the backbone size for $k = 0, 1, 2, 3, 4, \infty$ ($k = \infty$ is indicated with a "*" in the figures). The case $k = 0$ corresponds to the construction of the first, dense CDS: All nodes having two neighbors which are not in each other's transmission range are in the CDS. As the figure clearly shows, the application of the generalized rule with $k = 0$ is not very effective. As n (and, hence, the network density) increases, it becomes more and more difficult that each neighbor v of a generic node u is directly connected to all of u's neighbors. This motivates the increasing backbone size when n increases. When $n \geq 200$, basically all nodes are part of the CDS. The more k increases, the more effective the pruning rule: For removing u from the CDS C, a group of up to k of its neighbors in C is required to cover u and its neighbors. In dense networks ($n = 300$), the backbone obtained with $k \geq 3$ is 64 percent smaller than the backbone obtained when $k = 1$ and 71 percent smaller than the one obtained with $k = 0$.

Fig. 1b, Fig. 2a, and Fig. 2b refer to energy consumption, bytes overhead, and time for backbone formation, respectively, while k is varying. With increasing network density, the protocol performance degrades due to the higher number of nodes involved in the local exchanges of information, to the larger size of the message containing the neighbors list, and to the increased number of nodes competing for the radio channel. For all the considered scenarios, and independently of k, our results show excellent performance: WuLi is fast, has low overhead, and consumes little energy. This corresponds to the low



(a) (b)

Fig. 2. Average protocol overhead and duration as functions of k. (a) Overhead (bytes per node) and (b) protocol duration.

complexity of the protocol, to the little amount of information that each node needs to exchange with its neighbors, and to the high degree of localization of this protocol. We observe that, with respect to all these metrics, there is no significant performance difference for $k \geq 1$. Once information about the two hop neighborhood is exchanged, each node decides (and communicates to its neighbors) whether it belongs to the connected dominating set or not. This, together with the knowledge of the two-hop neighborhood, is the sole information needed to apply the generalized rule. A higher k only implies a (limited) increase in computational complexity. However, no extra message exchange is needed. The case $k = 0$ leads to reduced message overhead, shorter set up time, and lower energy consumption since the pruning phase of the protocol is skipped. However, the produced backbone is remarkably denser, which detrimentally affects the network operations and its lifetime because of increased routing overhead and of the larger number of nodes that must be kept awake for guaranteeing a connected communication structure.

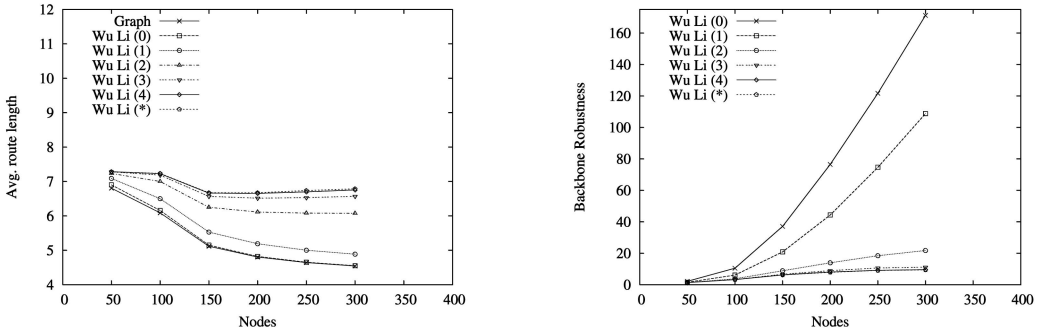
The average shortest paths length on the topology G_b induced by the backbone is depicted in Fig. 3a.

As k increases, the average shortest path length also increases. We observed that the routes on G_b are up to 48.6 percent longer than those of the network topology graph ($n = 300$ and $k \geq 4$). This reflects the fact the backbone topology gets sparser. Table 1 displays the average nodal degree in G_b. As expected, the higher the density, the shorter the routes.

The higher or the lower density of G_b results in a higher or a lower robustness in case of backbone nodes failures. This is shown in Fig. 3b. When $k \geq 3$, up to 10 randomly selected backbone nodes may die without affecting the backbone connectivity and the dominating property of the backbone. This number increases to over 20 when $k = 2$, and to over 100 in networks with 300 nodes when $k = 1$. Increased route length and decreased robustness are the price to pay for sparser backbones, leading to interesting, application-dependent trade offs between these two different metrics.

We proceed with the investigation of the WuLi protocol with respect to different rules for weight selection, namely, whether we choose a backbone node based on its ID [4] or on its degree, à la Stojmenovic [8]. In what follows, WuLi-D(k) indicates the WuLi protocol when we use Rule k and nodes' weights are their degrees (the node IDs are used to break possible ties). WuLi(k) denotes the WuLi protocol when we use Rule k and nodes' weights are their unique ID. Results are depicted in Fig. 4.

As expected, a degree-based weight leads to smaller backbones as higher degree nodes are more likely to be able to cover a node and its neighbors, pruning it from the backbone (Fig. 4a). While the WuLi(k) backbone contains the 51.6 percent of the n network nodes, the WuLi-D(k) contains only the



(a) (b)

Fig. 3. Average backbone route length and robustness as functions of k. (a) Route length and (b) robustness.

TABLE 1
Average Nodal Degree on the Backbone

n	50	100	150	200	250	300
WuLi(1)	2.48	3.89	6.03	8.29	10.76	13.2
WuLi(2)	2.28	2.79	3.26	3.53	3.63	3.71
WuLi(∞)	2.24	2.56	2.7	2.7	2.63	2.58
WuLi-D(1)	2.26	3.2	4.8	6.67	8.72	10.87
WuLi-D(∞)	2.17	2.55	2.86	3.01	3.08	3.18

46 percent of them when $n = 100$ and $k \geq 3$. We also have 36.7 percent (WuLi(k)) versus 33.6 percent (WuLi-D(k)) when $n = 200$ and $k \geq 3$, 66.7 percent versus 55.6 percent ($n = 100$ and $k = 1$), and 75 percent versus 63.2 percent ($n = 200$ and $k = 1$). This improvement is obtained at no extra cost in terms of overhead, energy consumption per node, and protocol duration. The number of protocol messages exchanged is exactly the same independent of the choice of weight. (The degree of a node is obtained "for free" from the size of its neighbor list.)

The degree-based weight selection results in denser backbones. This is confirmed by the values listed in Table 1, and it justifies why, for the same value of k, $k \geq 2$, routes in G_b are shorter than routes obtained when nodes are selected to the backbone based on their ID (up to 8 percent shorter in case $k \geq 4$).

Fig. 4c displays the robustness of the backbone for WuLi-D(i). The comparison between these values and the corresponding ones for WuLi(i) depicted in Fig. 3b shows that, independent of k, WuLi-D(i) is slightly less robust than WuLi(i). For higher ks (i.e., for denser backbones), this is explained by the fact that the backbone nodes are those with higher degrees, so removing x of them corresponds to eliminating more links from G_b with respect to those removed from the WuLi(i) backbone.

The outcome of the previous experiments motivates the two new ideas proposed in [8] by Stojmenovic et al. Given that the first rule of the WuLi protocol does not bring to the construction of a small CDS, why do we have to compute such a big set? If we could just apply Rule k to the two hop neighborhood, we would save the message exchange that according to WuLi, is needed for a node to communicate whether it is part of the CDS or not. This would have the beneficial effect of reducing protocol duration, protocol traffic, and energy consumption. The second idea has to do with the backbone size: Why don't we choose the backbone nodes based on their degree? In the following, we call "Stojmenovic" the protocol that combines these two ideas with WuLi's Rule k (where k has been set equal to ∞.)

In the last group of experiments for this first set of protocols, we compare the best performing protocols among WuLi(i) and WuLi-D(i) with Stojmenovic and with the extended version of MPR presented in [10], MPR-E. Results for the basic MPR protocols are not shown, since we observed that it is outperformed by MPR-E with respect to all the considered metrics. This is because MPR-E obtains a sparser G_b without affecting the protocol correctness and without requiring any extra message exchange. The main difference between the two versions is in the MPR-E rule for

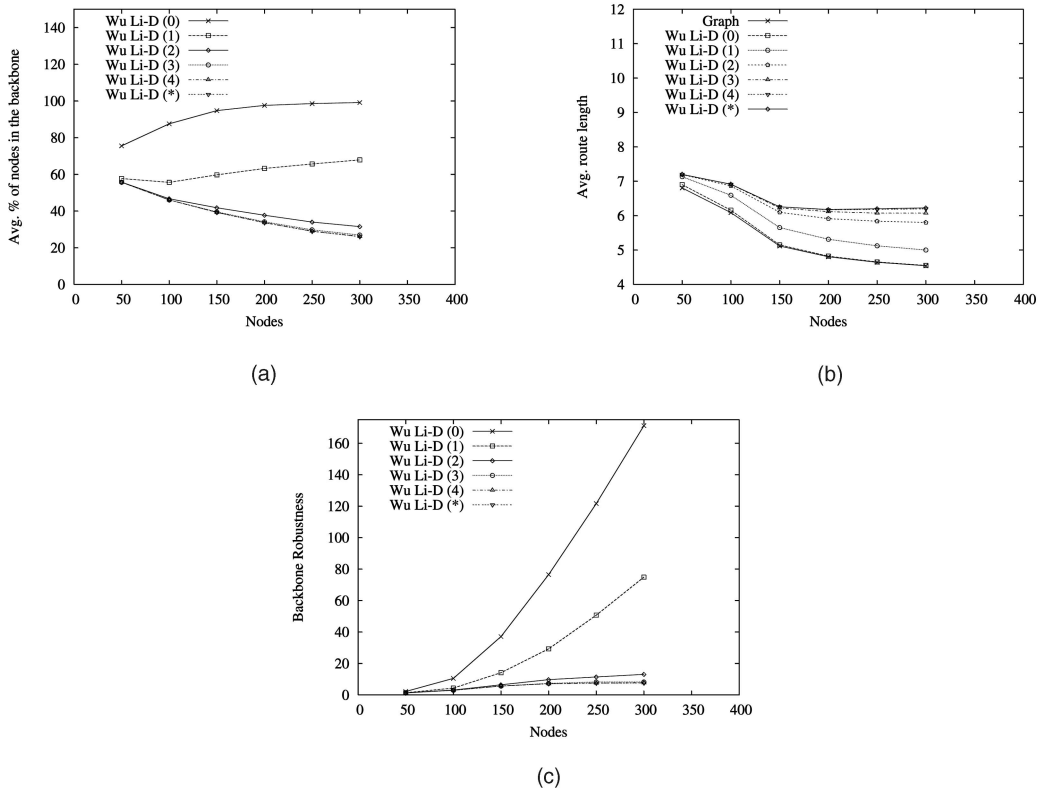


Fig. 4. Average backbone size, route length and robustness for WuLi-D(k) as functions of k. (a) Backbone size (percent of n), (b) route length, and (c) robustness.

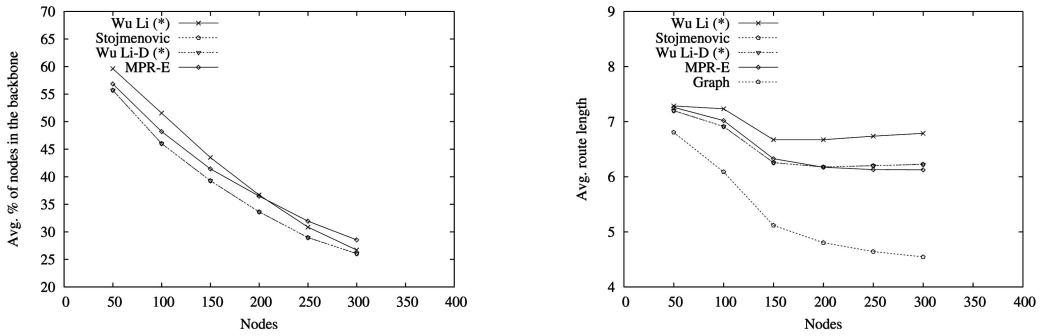


Fig. 5. Average backbone size and route length on the WuLi(∞), WuLi-D(∞), Stojmenovic, and MPR-E backbones. (a) Backbone size (percent of n) and (b) route length.

the selection of the multipoint relays which favors the inclusion of nodes which are not likely to be included in the backbone later on. As a result, MPR and MPR-E have the same performance in terms of overhead, protocol duration and energy consumption. MPR-E, however, generates smaller backbones: The percentage of nodes in the backbone decreases from 52.47 percent to 48.22 percent when $n = 100$, and from 37.93 percent to 36.49 percent when $n = 200$. The price to pay is a very small loss in backbone robustness which decreases from 13 to 12 in dense networks ($n = 300$).

Fig. 5a depicts the backbone size of WuLi(∞), WuLi-D(∞), MPR-E, and Stojmenovic for networks of increasing size ($50 \leq n \leq 300$).

The WuLi-D(∞) and Stojmenovic protocols have the best overall performance, which says about the clear advantage of a degree-based selection rule. We observe that the MPR-E protocol is not able to build a backbone as sparse as the one produced by WuLi-D(∞) and Stojmenovic, especially when the network density increases: When $n \geq 200$, MPR-E constructs backbones that are even bigger than those produced by WuLi (ID-based selection). In networks with 300 nodes, the MPR-E backbone is, on average, around 10 percent bigger than the WuLi-D(∞) and Stojmenovic backbones.

As is clearly depicted in the figure, WuLi-D and Stojmenovic have exactly the same performance in terms of backbone size, independent of the value of k (here $= \infty$). To explain this result requires us to investigate the details of the implementation of Stojmenovic. As mentioned, the idea is for a node to skip the phase of communicating to its neighbors whether it belongs to the CDS or not. This, of course, does not mean that a node should not apply the first rule of the protocol, which allows it to check its membership to the CDS. If (and only if) a node u belongs to the backbone, then it runs Rule k to see if it should be removed. The difference between WuLi-D(∞) and Stojmenovic is in the set of neighbors which can cover node u and its neighbors. These are only the neighbors which said they are part of the CDS in WuLi-D(∞) and *all* the neighbors in Stojmenovic. However, as proven by the following result, whenever a node u is removed because of a group of neighbors with cardinality k, it exists a group of k connected neighbors of u that belong to the CDS, which cover u and u's neighbors. In other words: *A node that is removed by the CDS by Stojmenovic is also removed by WuLi-D(∞).* This proves the equivalence of the two protocols, due to the "opposite" claim: *A node which is removed from the CDS by WuLi-D(∞) is also removed by Stojmenovic* is straightforward (as the set to which Stojmenovic applies Rule k includes the corresponding WuLi-D set). In the following theorem, we consider the CDS as formed by applying the first rule of WuLi, and we indicate with $C(u)$ a minimal connected group of a node u's neighbors which covers u and its neighbors.

Theorem 1. *Let u be a node that runs Rule k according to the operations of Stojmenovic and is pruned. Then, there exists a connected subset S of u's neighbors such that 1) $|C(u)| = |S|$, 2) all the nodes in S are part of the CDS, and 3) the nodes in S cover u and u's neighbors.*

Proof. For sake of contradiction, let us assume that there exists a node u such that all the minimal subsets S of cardinality $|C(u)|$ contain at least a node x which is not part of the CDS. We distinguish two cases, depending on the cardinality of $C(u)$. Case 1: $|C(u)| = 1$ (C(u) contains only x). Since $x \notin$ CDS, its neighbors are pairwise connected. Let $N(x)$ denote x's one-hop neighbors. As x covers u and its neighbors, these nodes belong to $N(x)$. Thus, u's neighbors are pairwise connected. But, this is impossible, since u applies Rule k (as all and only the nodes in the CDS after applying the first rule of WuLi). Case 2: $|C(u)| \geq 2$. In this case, there must be at least a connected neighbor y of node x in $C(u)$. Since x is not in the CDS, all its neighbors are pairwise connected. This implies that $N(x) \subseteq N(y)$. The set $C(u) \setminus \{x\}$ still satisfies the rules for pruning node u (as it still covers u and its neighbors and it is still connected), and it is smaller than $C(u)$. This leads to a contradiction since $C(u)$ would not be minimal, as required. □

In terms of overhead, time, and energy consumption, MPR-E and Stojmenovic save over 50 percent with respect to the same metrics of WuLi(∞) or WuLi-D(∞). This is because Stojmenovic is able to skip the communications of the nodes in the CDS generated by the first WuLi rule. MPR-E, instead, obtains this improvement because of the very limited number of messages needed to build a local MPR set (a node u informs only its MPR neighbors for

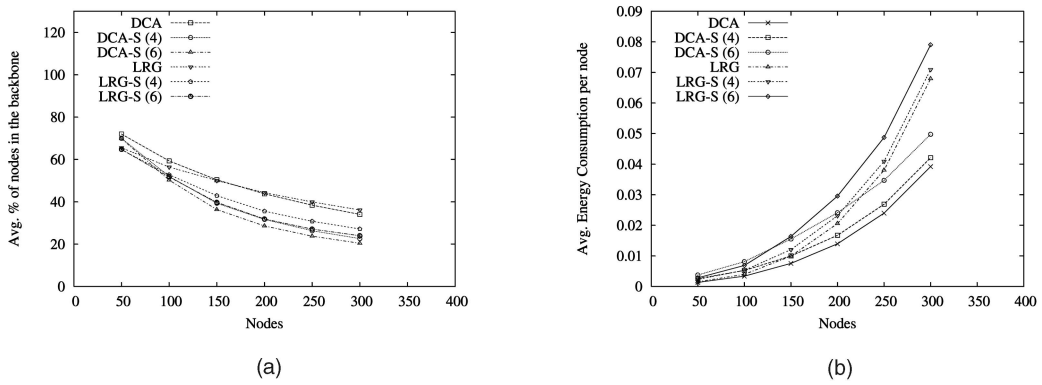


Fig. 6. Average backbone size and energy consumed by DCA, LRG, and their variants. (a) Backbone size (percent of n) and (b) energy per node.

which it is the lowest ID neighbor of the fact that they have been selected as MPR). MPR-E, Stojmenovic, and WuLi-D(∞) also build backbones with shorter routes with respect to those of WuLi(∞) backbones (up to 9 percent shorter).

In terms of backbone size, the sparser the backbone, the less robust it is. An average of 7.5 node failures are enough to disconnect the Stojmenovic backbone (or to isolate ordinary nodes) when $n = 300$. This number increases to 9.5 in the case of WuLi(∞) and to 12 in MPR-E backbones.

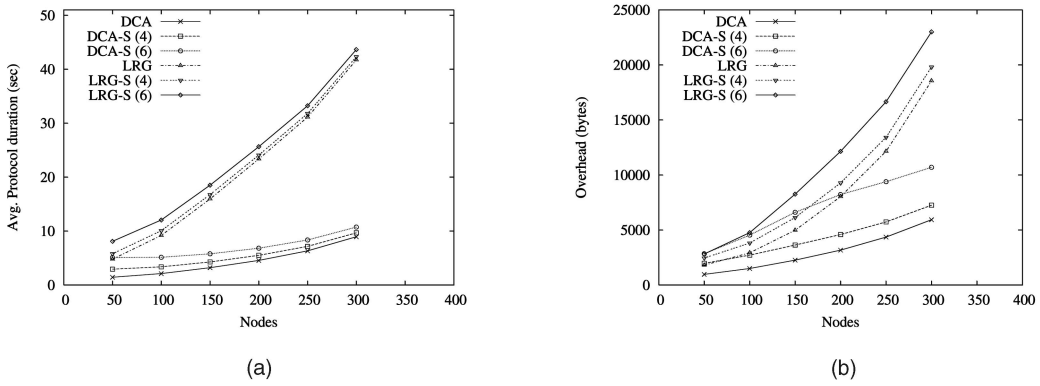
As a representative of the protocols that exhibit a high degree of localization, we choose Stojmenovic, given its slim backbones with short routes, its low overhead and energy consumption, and its fast operations.

3.2.2 Lower Degree of Localization: DCA and LRG-Based Schemes

Fig. 6, Fig. 7, and Fig. 8 display the performance, in terms of all the relevant metrics, of the DCA and LRG basic protocols as well as of their sparse versions DCA-S and LRG-S. With DCA-S(i) and LRG-S(i), we denote the protocols obtained by applying the sparsification rule proposed in [5] to the DCA and LRG backbones, respectively. We use the sparsification rule for breaking the cycles at most i-hop long in the auxiliary graph H.[1] Such a cycle is broken by deleting its "smallest," where smallest here is in accordance to a total ordering on the set of the network links. Since discovering cycles is both energy consuming and time demanding, we have limited our investigation to the case when $i \leq 6$. The cases when $i = 4$ and $i = 6$ are the only ones displayed in the figures since we observed that these two cases outperform the cases when $i = 3$ and $i = 5$ in terms of all the relevant metrics. Consider DCA-S(3) and DCA-S(4) (the same reasoning applies also to LRG). These two protocols are equivalent in terms of energy consumption, the duration of the sparsification phase, and byte overhead. This is because the discovery of i-hop long cycles, $i = 3, 4$, requires each clusterhead to send/receive messages to/from its neighbors clusterheads distant at most two hops in H. Hence, the message complexity of the cases $i = 3, 4$ and, similarly, $i = 5, 6$, is the same. However, DCA-S(4) and DCA-S(6) lead to smaller backbones than DCA-S(3) and DCA-S(5), respectively. Hence, we only consider the former here.

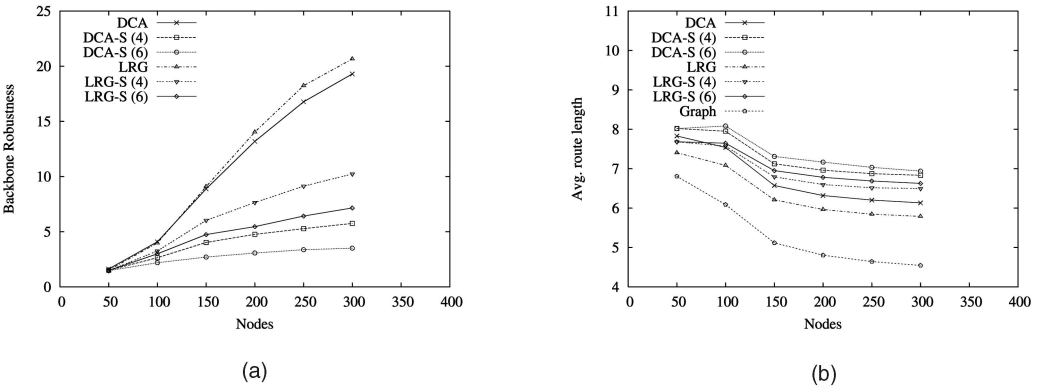
Fig. 6a compares the protocols performance with respect to producing a sparse connected backbones. All protocols obey the basic rule that the backbone size decreases as n increases: Growing network densities implies bigger cluster size and a wider choice of gateways. The DCA-S and LRG-S curves demonstrate the effectiveness of the sparsification rule applied to the DCA and LRG backbones. The percentage of nodes in DCA-S (LRG-S) backbones obtained by breaking triangles and squares ranges from 70 percent (65 percent) when $n = 50$ to 22.6 percent (27.2 percent) when $n = 300$. With respect to the nonsparsified case, the DCA-S (LRG-S) backbones are 3 percent (1.3 percent) to 33.5 percent (24.8 percent) smaller. When cycles up to six hops long are broken, we obtain a further 10 to 11 percent reduction ($n = 300$). In all considered scenarios, DCA and LRG have comparable backbone sizes and comparable number of clusterheads. The application of the sparsification rule, however, appears to be more effective on DCA-generated backbones. To justify this result, we analyzed the number of *virtual links* with length one and two (crossing one or two gateways) in the backbones generated by DCA and LRG. (A *virtual link* is a path interconnecting adjacent clusterheads.) A gateway may have more than one virtual link incident to it. All these have to be eliminated for the gateway to be pruned by the sparsification rule. In Table 2a and Table 2b, we have listed the number $Num_v(l)$ of virtual links of length l which were selected to be deleted by the sparsification rule. The number of virtual links for which the traversed gateways were actually removed from the backbone as a consequence of removing the virtual link is also listed (these tables are for the DCA-S(4) and LRG-S(4) protocols). In case of virtual links of length two, we distinguish between the number of virtual links in which only one of the two gateways was removed, denoted as $Num_v(2, 1)$, and the number of virtual links in which both gateways were removed, denoted as $Num_v(2, 2)$. $Num_v(1, 1)$ indicates the number of virtual links traversing only one gateway that were removed. The number of virtual links of length one in the backbone generated by DCA is equal to 13 when $n = 50$, and 52.8 when $n = 300$. The number of virtual links traversing two gateways is 5.5 when $n = 50$. This increases to 25.8 when $n = 300$. LRG has a much smaller number of virtual links of length one (6.7 when $n = 50$ and

1. We recall that the graph H has the clusterheads as its nodes, and includes an edge (u, v) if and only if clusterheads u and v are at most three-hops apart in the network topology graph.



(a) (b)

Fig. 7. Average protocol duration and overhead in DCA, LRG, and their variants. (a) Protocol duration and (b) protocol overhead (per node).



(a) (b)

Fig. 8. Average backbone robustness and increase in route length in DCA, LRG, and their variants. (a) Backbone robustness and (b) route length.

44.5 when $n = 300$) and a higher number of virtual links of length two (5.5 when $n = 50$ and 33.5 when $n = 300$). This reflects the fact that DCA, creating a set of independent nodes, distributes the clusterheads more evenly throughout the network, with a nonnegligible number of clusterheads that are two hops apart (virtual links of length one).

In the dominating set produced by LRG, clusterheads can be neighbors. We observe that LRG tends to produce groups of clusterheads which are neighbors interconnected by three-hop paths. The values in Table 2a and Table 2b confirm these observations.

In particular, DCA-S(4) erases a higher number of virtual links of length one (from more than six times as many when $n = 50$ down to three times as many when $n = 300$) with respect to the number of virtual links of the same length deleted by LRG-S(4). The percentage of gateways belonging to a virtual link of length one selected for elimination which are actually removed from the backbone is also higher in DCA-S(4) than in LRG-S(4) (from two thirds down to one third when $n = 300$). LRG-S(4) removes a higher number of virtual links of length two: An average of 29.41 at $n = 300$ versus the 25.47 removed on average by DCA-S(4). However, the advantage of removing more virtual links is balanced by the fact that DCA-S(4) successfully removes both gateways more often. The comparison between these results and the sizes of DCA-S(4) and LRG-S(4) backbones shows that the difference between backbone sizes is even more remarkable than the difference between the number of virtual links whose gateways are removed. In the results displayed in Table 2a and Table 2b, each gateway g pruned by the sparsification rule is counted multiple times: Once for each virtual link g belongs to. Thus, the fact that the difference in backbone size is higher than the difference between the number of deleted virtual links (the latter weighted according to the number of gateways actually removed) implies that, on average, gateways in LRG-S(4) belong to a higher number of virtual links. This in turn justifies the fact that it is more difficult to actually remove nodes from the LRG backbone using the sparsification rule.

Fig. 6a, Fig. 7a, and Fig. 7b depict the average energy consumption per node, the average protocol duration, and the overhead of DCA, LRG, DCA-S, and LRG-S. In terms of protocol duration, the DCA (and DCA-S(4)) protocol(s) show reasonably good performance, requiring a time which ranges from 1.43s (2.93s) when $n = 50$ to 8.94s (9.68s) when $n = 300$.

TABLE 2
Virtual Links Removed by (a) DCA-S(4) and (b) LRG-S(4)

n	50	300
$Num_v(1)$	0.55	19.55
$Num_v(1,1)$	0.27	11.63
$Num_v(2)$	1.79	25.47
$Num_v(2,1)$	0.62	11.31
$Num_v(2,2)$	0.11	9.95

(a)

n	50	300
$Num_v(1)$	0.1	9.81
$Num_v(1,1)$	0.04	3.7
$Num_v(2)$	0.78	29.41
$Num_v(2,1)$	0.29	13.38
$Num_v(2,2)$	0.05	7.88

(b)

The increase (up to 150 percent) in duration with respect to the WuLi protocol is due to the fact that the DCA operations require a node to wait for all its neighbors with bigger weight to communicate their role before it can decide its own. Also, once clusterheads are elected, the messages exchanged between potential gateways and clusterheads for backbone construction may require significant information exchange and, hence, extra time (although much less than for clusterhead selection). When $n = 100$ ($n = 200$) 1.2s (1.75s) are needed for joining clusterheads into a connected backbone.

It might appear counter-intuitive that DCA-S has a duration similar to that of DCA. This is because no dependency chains slow down the communication of information during the sparsification phase and that only a lower number of nodes participate in this phase. In other words, the sparsification phase has a high degree of localization. Since results are obtained averaging over the time required by all the nodes to complete the protocol, the extra delay required by some nodes to implement the "S" part of the protocol's name is balanced by many nodes quitting the protocol immediately after backbone formation. Only the nodes that are selected as part of the backbone in DCA need to proceed to the extra phase in which cycles are identified and broken. Our experiments show that the S phase is quite short: No longer than 2.16s (average) in DCA-S(4). DCA-S(6) has worse performance than DCA-S(4) since the exchange of information to detect longer cycles require more time. In this case, the time needed by the backbone nodes for completing the S phase averages at 5.3s.

The duration of LRG is much higher than that of DCA. This is because LRG needs several rounds for clusterhead selection and each round is implemented by the exchange of several messages (six messages are sent by every active node in a round). Since the number of active nodes decreases with the number of rounds, the time needed to complete each round becomes faster and faster. The average number of rounds needed by the LRG protocol ranges from 2 ($n = 50$) to 3.5 ($n = 300$). The corresponding protocol duration is up to 500 percent that of DCA. LRG-S does not take much longer than LRG for reasons similar to those discussed for DCA and DCA-S.

Fig. 7b depicts the overhead of the different protocols for networks of increasing size. The advantages of a higher degree of localization is evident: DCA requires almost four times the bytes transmitted by WuLi. Due to its more involved operations LRG has a bigger overhead: Over three times as much as DCA's. DCA-S(4) and LRG-S(4) show a moderate increase in overhead with respect to DCA and LRG, respectively, since the number of nodes involved in the S phase of the protocol is quite limited. However, we observe a significant increase for DCA-S(6) and LRG-S(6), confirming the fact that identifying cycles is a time and message consuming operation: As the cycle length i increases the cost of identifying all the cycles up to i increases quickly. For the S phase to be practical, the maximum length of the cycle has to be limited (we saw that $i \leq 4$ is acceptable). Moreover, applying the sparsification rule over the DCA and LRG backbone is effective in decreasing the overhead of the S phase, maintaining the effectiveness of the sparsification rule.

As expected, energy consumption results follow closely those for the overhead, given the dependency of energy consumption on the number of bytes transmitted and received by each node. In networks with 300 nodes DCA-S(6), DCA-S(4), DCA, LRG-S(6), LRG-S(4), and LRG nodes require 0.049J, 0.042J, 0.039J, 0.079J, 0.07J, and 0.067J, respectively.

Finally, Fig. 8a and Fig. 8b show the backbone robustness and the increase in the average route length with respect to the same metric in the network topology graphs. An increase in the average route length is expected, as clustering might force nodes that are neighbors in the network topology graph to communicate through their common clusterhead (if they belong to the same cluster) or through a possibly long backbone route in case they belong to different clusters.

This is made clear in Fig. 8b, which shows an average increase in route lengths that ranges in DCA backbones from 15 percent ($n = 50$) to 34.5 percent ($n = 300$), and, for LRG, from 8.8 percent ($n = 50$) to 27.3 percent ($n = 300$) with respect to the average route length in the network topology graph. DCA-S and LRG-S produce backbones that are sparser than DCA and LRG backbones, thus imposing routes which are even longer. DCA-S(4) has routes up to 11 percent longer than DCA's, while LRG-S(4) routes are an average of 12.1 percent longer than LRG's.

As expected, the backbone robustness increases with the backbone density. In case of sparse network topology graphs, a single node failure may lead to disconnections even in the network topology graph. As the network density increases, it gets more and more likely that when a backbone node failure occurs, the routes going through it can be rerouted to different backbone nodes. The denser the backbone, the more resilient it is to node failures. Dense and heavily meshed backbones such as DCA and LRG backbones may tolerate up to 20 failures in network with 300 nodes. This number reduces to only to 6 (10) for DCA-S(4) (LRG-S(4)).

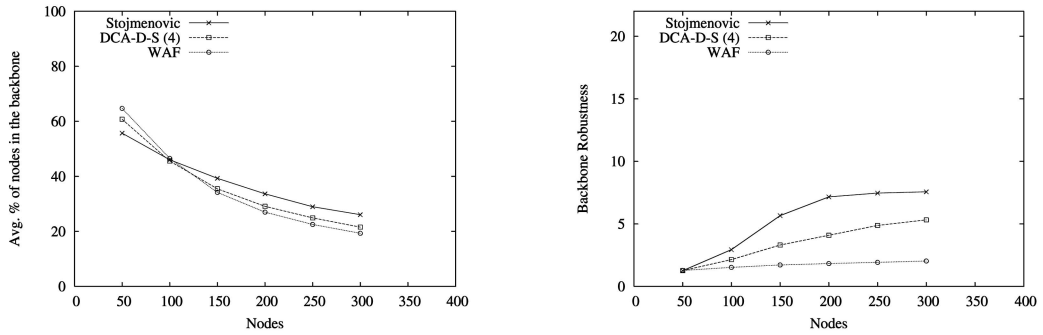
Our second batch of experiments show quite clearly that DCA and its S variant outperform the LRG protocols with respect to all selected metrics. The DCA-S is an effective way to further reduce the backbone size at the cost of additional time, energy, and overhead. We observe that a good compromise between the extra cost and good performance is obtained when the length i of the cycle to be broken is four. The backbone produced by DCA-S(4) is quite small and maintains an acceptable robustness at the price of a limited increase in energy consumption, overhead, time, and route length. Because of this, we selected DCA-S(4) as the "champion" for this set of protocols.

3.2.3 Comparison among Protocols with Different Degrees of Localization

In the third group of our experiments, we compare the performance of the champions of the two sets of protocols previously investigated with the WAF protocol presented in [9].

This allows us to show the effects of the different degrees of localization on protocols performance and to identify the best performing protocols among all the ones we considered.

Since the Stojmenovic idea of using a degree-based weight turned out to be effective in obtaining sparser backbones, for fairness of comparison, we have implemented a variant of the DCA-S(4), where clusterheads are



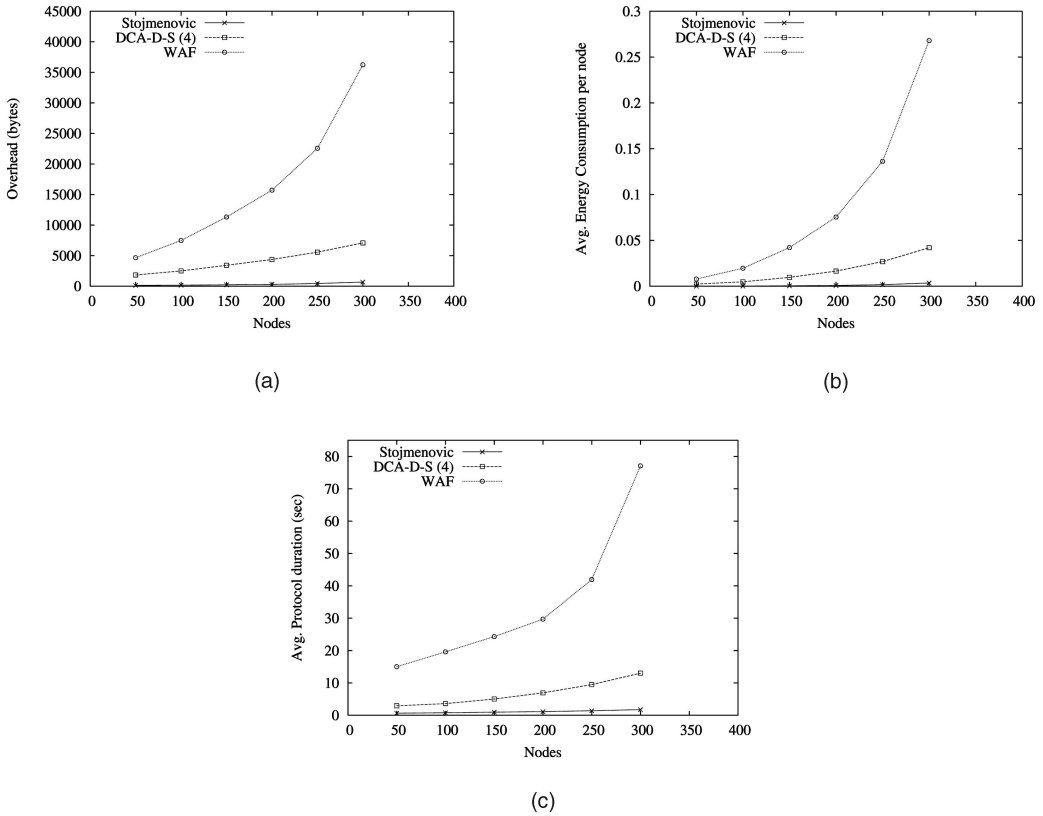
(a) (b)

Fig. 9. Average backbone size and robustness for DCA-D-S(4), Stojmenovic, and WAF. (a) Backbone size (percent of n) and (b) backbone robustness.

selected based on their nodal degree. We call this variant of the DCA the DCA-D-S(4) protocol.

The results of our comparison are depicted in Fig. 9 and Fig. 10. The average percentage of nodes in the backbones is shown in Fig. 9a. WAF generates a thin "tree-like" backbone resulting, as expected, in a backbone considerably smaller than DCA-D-S(4) and Stojmenovic. However, the DCA-D-S(4) backbone size is only 11.8 percent bigger than WAF's in the denser networks we considered ($n = 300$). This clearly shows the effectiveness of using the described sparsification algorithm. Breaking cycles appears to be more effective than the sparsification rule used by Stojmenovic. The Stojmenovic backbones are up to 21 percent bigger, on average, than DCA-D-S(4) backbones. Slimmer backbones result in a slight increase in route lengths and in reduced robustness. The slim WAF backbone cannot survive many failures: The failure of two nodes is, on average, more than enough to disconnect it or leave some ordinary nodes uncovered. Although DCA-S generates backbones almost as thin as WAF backbones, its resilience to nodes failures is higher: An average of six failures can be tolerated without disconnecting the network ($n = 300$). The denser Stojmenovic backbone survives up to 7.5 backbone nodes removals when $n = 300$.

Fig. 10a, Fig. 10b, and Fig. 10c show the overhead, energy consumption, and time needed by the three protocols.



(a) (b)

(c)

Fig. 10. Average overhead, energy consumption and duration of DCA-D-S(4), Stojmenovic, and WAF. (a) Protocol overhead (bytes sent per node), (b) energy consumption (per node), and (c) protocol duration.

These figures clearly show the gains that can be achieved by means of higher degrees of localization, and simple, "light" protocols. Stojmenovic has a strong advantage with respect to these three metrics over all the other protocols. This is due to its low complexity, the reduced number of information that each node needs to exchange with its neighbors, and the possibility to exchange this information in parallel (high degree of localization). The case of the WAF protocol clearly exemplifies the limits of an approach with a very low degree of localization. It is by far the most energy consuming protocol, and, requiring up to 77s to produce a connected backbone, it is by far the slowest one. (Stojmenovic never requires more than 2s to construct the backbone.) The long duration of WAF is mostly due to the nontrivial complexity of the first phase of the protocol (leader election) that requires a very large number of messages to be exchanged for progressively joining network fragments and to identify a common leader. The distributed implementation of this phase is time and overhead demanding, intrinsically sequential (very low degree of localization), and constitutes the real obstacle to adopt WAF in practical scenarios. In networks with 300 nodes, it takes 67.5s to complete leader election. The 85 percent of the overall energy consumed is spent in this phase. In terms of message overhead, WAF requires up to 53 times the bytes of Stojmenovic. This is again due to the overly high complexity of the WAF leader election phase, which requires information to be propagated to all the nodes in a fragment and to nodes in adjacent fragments every time two fragments are merged into a new one. When $n = 300$, the leader election phase accounts for 84.8 percent of the total WAF overhead.

One may observe that, in a wireless sensor network, the time and energy consuming leader election phase could be skipped by automatically designating the network sink as the leader. This significantly reduces overhead, energy consumption, and WAF duration. We observe that, if the leader election phase can be avoided the WAF duration, its energy consumption and overhead are comparable to those of DCA-D-S(4).

Skipping the leader election phase is quite natural for WSN with a single sink. In case of multiple sinks operating independently, no changes in the protocols performance occur in DCA-D-S(4) and Stojmenovic. All sinks are connected to (included in) the produced backbone. In case of multiple sinks, the WAF protocol without the leader election phase has each sink constructing separate trees. We have observed that the backbone obtained by merging the trees has a very large number of nodes. Over 50 percent (32 percent) of the nodes of a network with four (two) sinks are in the backbone. The average backbone is obtained by interconnecting the dominating sets resulting from the execution of the WAF protocol by h sinks (randomly selected among the nodes), $h = 1, \ldots, 4$. The h sinks run the backbone formation phase of the WAF protocol independently. The final backbone contains the union of the backbone nodes of the tree-like connected dominating sets generated by each of the sinks. As h increases, the backbone size rapidly increases, affecting the effectiveness of the network operations.

Overall, our results show that the "S" rule for sparsification is a very effective technique for pruning CDSs without compromising their connectivity and dominance. In terms of backbone size, the performance of protocols that can apply such a rule is superior to the performance of all the variants of the WuLi and MPR protocols. However, when the other relevant metrics are taken into account, a higher degree of localization really makes the difference, significantly reducing the protocol duration, the overhead, and the energy consumed for the backbone set up. Stojmenovic is an excellent compromise with respect to all these metrics.

4 Conclusions

In this paper, we presented a thorough ns2-based comparative performance evaluation of protocols for clustering and backbone formation in large ad hoc networks with energy-constrained nodes. In particular, we considered protocols that are among the most representative of this area of ad hoc research with a particular emphasis on investigating the effects of their "degree of localization" on protocol performance, i.e., how being able to execute the protocol based only on local information improves the overall performance. Extensive ns2-based simulation results show that highly localized protocols are rewarded with good performance with respect to all metrics of interest which include protocol duration, energy consumption, message overhead, route length, and backbone size.

Acknowledgments

This work was supported in part by the European Project E-SENSE (Cooperating Embedded Systems for Exploitation and Control Featuring WSNs).

References

[1] C. Adjih, P. Jacquet, and L. Viennot, "Computing Connected Dominated Sets with Multi-Point Relays," *Ad Hoc & Sensor Wireless Networks 1,* vol. 1, pp. 27-39, 2005.

[2] S. Basagni, "Distributed Clustering for Ad Hoc Networks," *Proc. 1999 Int'l Symp. Parallel Architectures, Algorithms, and Networks (I-SPAN '99),* pp. 310-315, June 1999.

[3] I. Chlamtac and A. Faragó, "A New Approach to the Design and Analysis of Peer-to-Peer Mobile Networks," *Wireless Networks 5,* vol. 3, pp. 149-156, May 1999.

[4] F. Dai and J. Wu, "An Extended Localized Algorithms for Connected Dominating Set Formation in Ad Hoc Wireless Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 15, no. 10, pp. 908-920, Oct. 2004.

[5] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, "Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons," *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA),* pp. 717-724, Jan. 2003.

[6] L. Jia, R. Rajaraman, and T. Suel, "An Efficient Distributed Algorithm for Constructing Small Dominating Sets," *Distributed Computing,* vol. 15, no. 4, pp. 193-205, Dec. 2004.

[7] B. Liang and Z. Haas, "Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management," *Proc. 19th IEEE Infocom,* vol. 3, pp. 1293-1302, Mar. 2000.

[8] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbors Elimination-Based Broadcasting Algorithms in Wireless Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 1, pp. 14-25, Jan. 2002.

[9] P.-J. Wan, K.M. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Sets in Wireless Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Applications (MONET) 9,* no. 2, pp. 141-149, Apr. 2004.

[10] J. Wu, "An Enhanced Approach to Determine a Small Forward Node Set Based on Multi-Point Relay," *Proc. 58th IEEE Semiann. Vehicular Technology Conf., VTC 2003-Fall,* vol. 4, pp. 2774-2777, Oct. 2003.
[11] J. Wu and H. Li, "On Calculating Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks," *Proc. Third ACM Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm. (Dial M 1999),* pp. 7-17, Aug. 1999.

Stefano Basagni received the PhD degree in electrical engineering from the University of Texas at Dallas (December 2001) and the PhD degree in computer science from the University of Milano, Italy (May 1998). He received the BSc degree in computer science from the University of Pisa, Italy, in 1991. Since Winter 2002, he has been on the faculty of the Department of Electrical and Computer Engineering at Northeastern University, Boston, Massachussetts. From August 2000 to January 2002, he was a professor of computer science in the Department of Computer Science at the Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas. Dr. Basagni's current research interests concern research and implementation aspects of mobile networks and wireless communications systems, Bluetooth and sensor networking, definition and performance evaluation of network protocols, and theoretical and practical aspects of distributed algorithms. He has published more than three dozen referred technical papers. He is also coeditor of two books and coauthors of three book chapters. He served as a guest editor of the special issue of the *Journal on Special Topics in Mobile Networking and Applications (MONET) on Multipoint Communication in Wireless Mobile Networks*, of the special issue on mobile ad hoc networks of *Wiley's Interscience's Wireless Communications & Mobile Networks* journal, and of *Elsevier's Journal Algorithmica* on algorithmic aspects of mobile computing and communications. He serves regularly as a member of the editorial board and of the organizing and technical program committee of ACM and IEEE journals and international conferences. He is a member of the ACM (including the ACM SIGMOBILE), the IEEE, the IEEE Computer Society, and the IEEE Communication Society.

Michele Mastrogiovanni received the laurea degree in computer science with honors from the University of Rome "La Sapienza," Italy, in May 2004. He is currently a PhD student in computer science at the University of Rome "La Sapienza." His current research interests include wireless sensor and ad hoc networks, large scale ad hoc networks, and the design and implementation of ad hoc protocols.

Alessandro Panconesi received the PhD degree in computer science from Cornell University. He is a professor of computer science at the University of Rome "La Sapienza." He was awarded the ACM Danny Lewin Award in 1992. His main research interest concerns the design and analysis of probabilistic and distributed algorithms. He is the director of BICI (Bertinoro International Center for Informatics).

Chiara Petrioli received the Laurea degree with honors in computer science and the PhD degree in computer engineering in 1998, both from Rome University "La Sapienza," Italy. She is currently an assistant professor in the Computer Science Department at Rome University "La Sapienza." Her current work focuses on ad hoc and sensor networks, Bluetooth, energy-conserving protocols, QoS in IP networks, and content delivery networks. Prior to Rome University, she was a research associate at the Politecnico di Milano. She was also working with the Italian Space agency (ASI) and Alenia Spazio. Dr. Petrioli is an area editor of the *IEEE Transactions on Vehicular Technology*, *ACM/Kluwer Wireless Networks Journal*, the *Wiley InterScience Wireless Communications & Mobile Computing Journal*, and the *Elsevier Ad Hoc Networks Journal*. She has served in the organizing and technical program committees of leading conferences in the area of networking and mobile computing including ACM MobiCom, ACM MobiHoc, and IEEE ICC. Dr. Petrioli was a Fulbright scholar and is a member of the ACM and the IEEE.