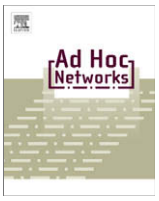
Ad Hoc Networks 7 (2009) 1347–1369

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



Evaluation of physical carrier sense based spanner construction and maintenance as well as broadcast and convergecast in ad hoc networks

Luke Ritchie [a], Sapna Deval [a], Martin Reisslein [a,*], Andrea W. Richa [b,1]

[a] *Dept. of Electrical Engineering, Arizona State University, Goldwater Center, MC 5706, Tempe AZ 85287-5706, United States*
[b] *Dept. of Computer Science and Engineering, Arizona State University, Box 878809, Tempe, AZ 85287-8809, United States*

ARTICLE INFO

Article history:
Received 31 October 2007
Received in revised form 23 September 2008
Accepted 27 January 2009
Available online 6 February 2009

Keywords:
Ad hoc network
Backbone network
Broadcast
Physical carrier sensing
Resilience
Variable sensing ranges

ABSTRACT

While the use of physical carrier sensing for medium access control in ad hoc wireless networks is well established, exploiting physical carrier sensing directly for network layer functions is largely unexplored. We conduct extensive simulation evaluations of recently proposed algorithms that directly exploit physical carrier sensing for backbone network (spanner) construction, broadcast, and convergecast in wireless ad hoc networks. Our algorithms accommodate interference ranges larger than transmission ranges, explicitly incorporate the medium access control and packet collisions, and do not require any prior knowledge of the network. For spanner construction, our algorithms include three self-stabilizing phases that establish leader nodes able to reach all nodes in one hop, assign the leaders non-interfering transmission rounds, and connect the leaders through gateway nodes. We evaluate the backbone construction and maintenance as well as broadcast and convergecast through simulations. We find that over 75% of the control messages for backbone network construction are received from physical carrier sensing. While the number of backbone nodes is relatively large, the backbone is very robust, quickly self-stabilizing, and only a fraction of the backbone nodes are used for broadcast.

1. Introduction

Sending data from one node to all others (*broadcast*) and gathering data from all nodes at a single sink (*convergecast*) are two fundamental operations in wireless ad hoc networks, including sensor networks. Broadcast is necessary for distributing program updates, route requests and data queries. Convergecast is used to collect data from sensors and maintain base station connectivity in multihop networks. Finding the optimal tree for broadcast/convergecast operations is an NP-hard problem, even when the network is assumed to be a simple bi-directional graph with all edge weights known ahead of time. However, this level of simplification is not realistic: nodes may interfere with each other even when they are too far apart to communicate reliably, and no node in an ad hoc network can be expected to possess global knowledge about the other nodes' connectivities.

In recent algorithm-theoretic work [1,2] we have proposed algorithms for constructing backbone networks so as to enable efficient broadcast/convergecast in wireless ad hoc networks.[2] Distinctions of our algorithms from existing approaches are that our algorithms consider a general network model with non-uniform transmission and interference ranges as well as interference ranges longer than transmission ranges. Previously existing algorithms were mainly developed for simpler, less detailed network models, such as the unit-disk model, which con-

* Corresponding author. Tel.: +1 480 965 8593; fax: +1 480 965 8325.
E-mail addresses: Luke.Ritchie@asu.edu (L. Ritchie), Sapna.Deval@asu.edu (S. Deval), reisslein@asu.edu (M. Reisslein), aricha@asu.edu (A.W. Richa).
URL: http://www.fulton.asu.edu/~mre (M. Reisslein).
[1] Tel.: +1 480 965 7555; fax: +1 480 965 2751.

[2] For brevity we refer to the algorithms proposed in [1,2] as *our* algorithms throughout this manuscript.

1570-8705/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2009.01.004

siders only a disk-shaped transmission range and ignores interference, and the packet radio model, which considers disk-shaped transmission and interference ranges whereby both ranges are equal. Furthermore, existing approaches exploit physical carrier sensing primarily for efficient medium access control, whereas our algorithms exploit physical carrier sensing for constructing a backbone network—a network layer task—while explicitly incorporating the medium access control. The backbone network connects all nodes with an approximately minimum number of hops and forms the basis for efficient broadcast/convergecast.

The main contribution of this paper is an original simulation study of the backbone network (spanner) construction and broadcast/convergcast algorithms proposed in [1,2]. Our algorithm-theoretic work [1,2] provided only asymptotic performance bounds, but did not examine the actual performance of the algorithms for typical network scenarios. In this paper, we evaluate the actual performance of the backbone network construction algorithms for typical network scenarios through simulations and contrast our approach with existing methods. We find that in a network of nodes without any neighbor or topology knowledge, our algorithms utilize mainly carrier sensing information to very quickly establish leader nodes that can reach all nodes through non-interfering transmission rounds. These non-interfering transmission rounds are then exploited to minimize collisions when sending detailed node address information to link the leader nodes into a backbone network. Over 65% of the information bits for backbone network construction are sent through the non-interfering transmission rounds, thus avoiding collisions.

Our simulations provide detailed insights into the performance of the individual phases of our algorithms. Furthermore, the simulation results indicate that our overall approach compares favorably with existing strategies that rely on a neighbor discovery protocol in conjunction with backbone network construction algorithms utilizing neighbor lists as well as existing strategies that form a backbone network from nodes with different prior knowledge (such as number of neighbors) and without exploiting physical carrier sensing. We also find that while our backbones are relative large, containing close to 50% of the nodes in a 900 node network, the backbones are robust, remaining functional even when roughly two thirds of the backbone nodes are lost. Also, broadcast involves typically less than about half of the backbone nodes in the 900 node network.

This paper is structured as follows: in the following subsection, we review related work. In Section 2, we present our network model, including the physical carrier sensing ranges. In Section 3, we give an overview of the spanner construction algorithm proposed in [1]. In Section 4, we present our simulation evaluations of the spanner construction, spanner resilience, as well as broadcast and convergecast. We summarize our conclusions in Section 5.

1.1. Related work

Broadcasting (in the sense of network-wide flooding) in ad hoc networks has been studied extensively; see, for instance [3–9]. While simulations, as employed in this paper, have been the primary means of performance evaluation, there have been a few mathematical analyses. Chlamtac and Weinstein [10] and Bar-Yehuda et al. [11] were among the first to present algorithms with formally analyzed complexity for respectively the centralized and distributed cases of broadcasting in wireless ad hoc networks. Tseng et al. [12] analyzed the difficulties of relying on simple flooding as a broadcast mechanism. Gandhi et al. [13] proved that minimum latency broadcast in ad hoc networks is NP-hard, and provide a broadcast algorithm with constant latency based on a unit-disk network model. A probabilistic analysis of two broadcast schemes is conducted in [14]. Simulation-based studies include work by Stojmenovic et al. [15], in which an efficient broadcasting scheme based on a dominating set is developed. Recently, Wu and Dai [16] presented an efficient broadcasting scheme for mobile networks, based on distributing topology information through “hello” messages with a larger than normal transmission range. These existing approaches have primarily been developed and evaluated for the unit-disk and packet radio models where the interference range does not exceed the transmission range. Jakllari et al. [17] have recently proposed a cross-layer ad hoc networking framework using cooperative transmissions with space–time block coding at the physical layer. They consider different transmission and interference ranges that depend on the achieved physical layer diversity gain. In [18] they examined this cooperative transmission strategy in the context of the counter-based broadcasting strategy proposed in [12], which does not involve a backbone or other topology control mechanisms. In contrast, we examine a suite of network layer protocols that first form a backbone network, which then can be used for broadcast, convergecast, and other network layer functions, while considering interference ranges larger than transmission ranges.

Convergecast and the related problem of data aggregation in wireless sensor networks have been considered in numerous studies, e.g., [19–24], which assume different sets of network conditions and performance priorities. Interference and channel contention at the MAC layer, which we explicitly incorporate into our algorithm designs, are generally ignored in these existing convergecast approaches.

The problem of finding an optimal backbone (overlay) network for broadcast or convergecast is usually expressed in terms of minimum connected dominating sets and spanning sets. Finding these sets is an NP-complete problem even with a basic unit-disk model of the network. Distributed algorithms that approximate the minimum connected dominating set with polynomial or polylogarithmic running time include, e.g., [15,25–29]. The first phase of our algorithm (described in Section 3.1) is an extension of the dominating set algorithm [30]. Comprehensive simulation comparisons of clustering and overlay network formation algorithms developed for unit-disk and packet radio models are reported in [31]. The comparison study [31] takes MAC packet collisions into consideration and classifies the algorithms according to their level of localization, i.e., over how many hops does the information for forming the overlay travel, a classification also employed in [32].

Approaches with a high level of localization that require information from within only a 2-hop local neighborhood, such as approaches based on [15,33], are compared with approaches that have lower levels of localization, such as algorithms based on [34,35]. It is found that highly localized approaches tend to give the best performance. Our approach, which is developed for a more realistic network model with interference ranges larger than transmission ranges, is highly localized. In our backbone construction, information never travels further than twice the interference range (via two physical carrier sensing communication hops in phase two, see Section 3.2) or three times the transmission range (via three actual packet transmissions in phase three, see Section 3.3).

In work by Kuhn et al. [36] and Parthasarathy and Gandhi [37,13], distributed algorithms are presented that compute constant factor approximations of a minimum dominating set in poly-logarithmic time. Both extend the unit disk model taking interference into account, but nodes need to know an estimate of the size of the network. In contrast, our approach does not require any estimates of network density, nor the total number of nodes in the network.

Physical carrier sensing has been studied from a variety of perspectives. In single-hop communication, physical carrier sense is used in many random-access schemes, such as 802.11's version of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Many lines of work focus on the MAC layer — studying or optimizing the effect of CSMA on throughput and power consumption, see e.g., [38–47]. The problem of topology control in ad hoc networks, see e.g., [48,49], has some similarity to our work in goals and approaches. For example, Muqattash and Krunz [50,51] use virtual and physical carrier sense with power control to increase throughput and energy efficiency. Similarly, Tavli and Heinzelman [52] employ a cross-layer approach spanning the MAC and network layer to minimize energy for broadcast. However, to the best of our knowledge, our protocol is the first to directly exploit physical carrier sense for the network layer task of distributed spanner construction.

2. Network model

Our model makes very few assumptions about the connectivity among nodes in the network. Unlike the relatively restrictive unit disk or packet radio models, the minimum assumptions for our algorithms can accommodate a number of real world scenarios and form the basis for a robust cross-layer design.

Assuming that all the nodes operate in the same frequency band and use the same data rate, there will be a *transmission range* r_t that increases with transmission power. For simplicity of discussion, we will assume all nodes use the same fixed power and have a transmission range defined by the same r_t. Any two nodes closer together than $r_t/(1 \pm \delta)$ (where δ, $0 \leqslant \delta < 1$ is a constant that represents the non-uniformity factor of the network) can reliably communicate, while nodes farther apart cannot. We define the *interference range* r_i, such that if a transmission fails due to interference, the interfering transmission must have originated at a node closer than $r_i/(1 \pm \delta)$ to the receiving node. Not all transmissions from nodes within r_i will cause interference, but transmissions from nodes outside of it never will. In a typical network, r_i is 2–3 times larger than r_t; our examples and simulations are generally based on a ratio of $r_i/r_t = 2$. Due to the δ factor, the actual transmission area can be an arbitrary shape with *approximate* radius r_t.

We also assume that each node can perform physical carrier sensing to detect when the medium is busy. In 802.11, this is based on Received Signal Strength Indicator (RSSI) measurements. In a similar manner to r_t and r_i, we define the *certain-sensing range* r_{st} and the *non-sensing range* r_{si} for the carrier sense operation: signals traveling less than $r_{st}/(1 \pm \delta)$ are sensed with probability close to one, whereas signals traveling farther than $r_{si}/(1 \pm \delta)$ are sensed with close to zero probability, and in between with arbitrary probability. We further assume that these ranges can be tuned by adjusting the SNR threshold at the receiver. In 802.11, the threshold can be adjusted through the Clear Channel Assessment rules. As with transmission ranges, $r_{si}/r_{st} \approx 2$–3 typically, and for ease of discussion, we set $r_{si}/r_{st} = r_i/r_t = 2$.

This model is a close match for the actual performance of current wireless interfaces. Forward error correction mechanisms allow for relatively sharp cutoffs between the area where messages are almost always received (*transmission range*; a range r such that the communication cost $r \cdot (1 \pm \delta)$ is less than the transmission range r_t, i.e., $r \cdot (1 \pm \delta) < r_t$), where they may still interfere (*interference range*; $r_t < r \cdot (1 \pm \delta) < r_i$), and where they never interfere ($r \cdot (1 \pm \delta) > r_i$). (For generalizations to communication costs that are not monotonic in the distance r we refer to [1].) We refer to a pair of nodes as *connected* if there exists a path between the two nodes (possibly including hops traversing intermediate nodes) such that the communication cost $r \cdot (1 \pm \delta)$ is less than the transmission range r_t for each hop along the path. We refer to a set of nodes as connected if each pair of nodes in the set is connected.

3. Spanner construction algorithm

The algorithm presented in [1] constructs an overlay network in three phases. In Phase I, the distributed election of "leader" nodes creates a dominating set: every node in the network is either a leader or within the transmission range r_t of at least one leader. Phase II consists of a distributed assignment of leader time slots such that each leader can communicate with neighboring non-leader nodes without interfering with other leaders' transmissions. (The same non-interfering assignment will be used later to reduce collisions during broadcast and convergecast.) In Phase III, leaders select gateway nodes to form a spanning set—a connected set of leaders and gateways. Selecting a minimum dominating set or spanning set is NP-hard, but our distributed algorithm selects a spanner with bounded density in poly-logarithmic time.

The timing structure of the algorithm is organized according to locally synchronized "rounds", such that each

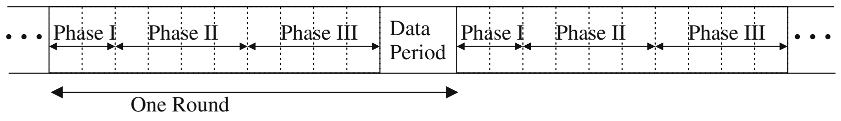


Fig. 1. Illustration of algorithm timing structure: a round contains slots for each phase of the algorithm. k rounds make up one frame.

round contains 10 slots to accommodate phases I–III (see Fig. 1). A sequence of k rounds forms a "frame", where the parameter k will be determined later (see Section 4.3). The beginning of each frame is not required to be synchronized between nodes, only the round/slot timing. More specifically, the beginning and duration of each individual round and slot must be synchronized among the nodes. However, each node may start its frame, i.e., the sequence of k consecutive rounds forming a frame, with a different round.

An important feature of the algorithm is that all three phases run in parallel. Each round contains slots for all three phases, as illustrated in Fig. 1. It is common in articles describing spanner construction algorithms made up of multiple phases to speak as if each phase does not begin until the previous phase has completed. This simplifies discussion, but in practice this type of synchronization would be infeasible without some means for every node to know that each phase has stabilized. In contrast, our algorithm is designed to run in parallel. Once each phase has stabilized, the next phase can self-stabilize without any global signal that this transition has occurred. Running in parallel is also important for the algorithm to adapt to mobility. The time between rounds is used for sending data messages (Fig. 1). If one or more nodes move, the algorithm has the ability to re-stabilize during subsequent rounds.

3.1. Leader election (Phase I)

The leader election algorithm works roughly as follows. Every node attempts to become a leader by announcing its candidacy for leadership if it has heard no other leader announcements. The timing of these announcements requires no prior knowledge about neighboring nodes, not even the number of nodes. In particular, a node listens and, if it has not heard from another potential leader for one full frame, it becomes "active" in the current round and sends out a leader message with a constant probability p (that is independent of the density of the network) in the second slot of that round every frame. If a leader candidate chooses to listen (with probability $1-p$) and senses or receives another leader message, it reverts to an inactive state. As long as a node remains active, it sends an active message in the first slot of that round in every frame, and possibly (with probability p) a leader message in the second slot. The algorithm can be proven to quickly converge to a stable and complete leader set for any reasonable p value [1].

More specifically, during the leader election phase, two different carrier sensing thresholds are used, depending on the state of the node. In the inactive listening state, a smaller listening range is used. In particular, the non-sensing range is set equal to the transmission range $r_{si}=r_t$, which is sufficient for assessing whether there are other leader candidates nearby. The idea is that a node should avoid becoming part of the leader set if there is another leader within its certain-sensing range r_{st}; this rule puts an upper limit on the number of members the leader set will include. Only transmissions from nodes within the r_t range can possibly be heard, and transmissions from within $r_{st}=r_t/2$ will almost certainly be heard (see Fig. 2a).

If, after listening for a full frame, a node has heard nothing with the smaller range, it uses a larger sensing range during the round it is attempting to claim. A range set large enough to sense a busy medium reliably at the interference range, $r_{st}=r_i$ (see Fig. 2b), has the effect of setting a lower limit on the distance between leaders with the same active slot. Based on this property, it can be shown that the leader set is stable; i.e., once a node becomes the only active node for a given round within its transmission range r_t, it will remain active.

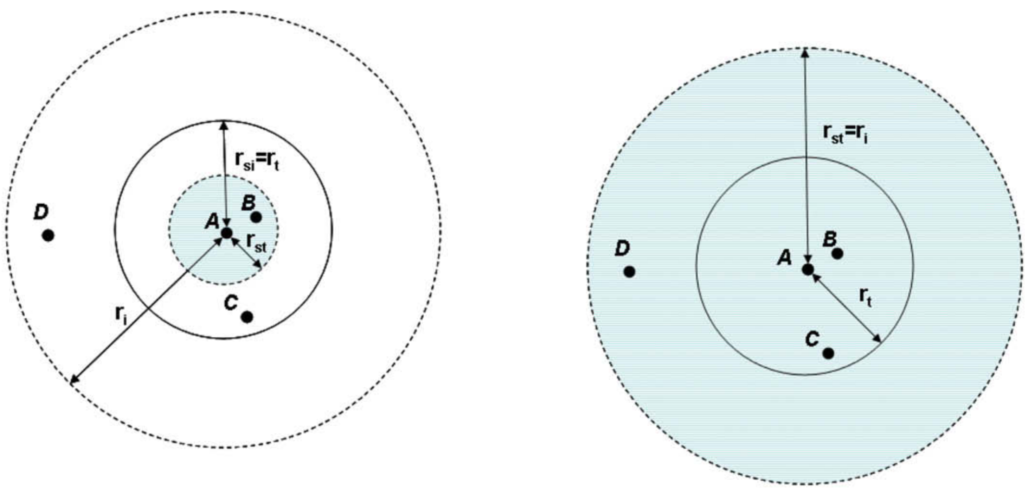
It is of course possible that multiple nodes will attempt to become active during the same round. The probabilistic exchange of leader messages in the second slot allows candidate leader nodes to discover other such nodes within their transmission radius r_t.

At the end of the leader election phase, all of the nodes will be "covered" by the transmission ranges of leader nodes. No two leaders will be closer than $r_t/2$, and no two leaders active in the same round will be within each others' transmission range r_t. In Phase II of the overlay network construction algorithm, the active rounds of the leader nodes are re-assigned such that interference of two leaders at any other node is impossible.

3.2. Non-interfering transmission round assignment (Phase II)

The goal of the second phase is to assign each leader node a non-interfering transmission round. In other words, Phase II is a distributed algorithm for creating TDMA assignments among leader nodes. The general requirement for non-interference is that no two nodes within an $r_i+r_t=3r_t$ range transmit simultaneously, i.e., A's transmission range does not overlap with B's interference range, and vice versa.

Basically, we are concerned with two nodes that cannot "hear" each other transmitting at the same time—a version of the *hidden terminal* problem. If two sources are out of the transmission range ("hidden" from each other), they may cause a collision (interfere) if they transmit simultaneously. For example, if two nodes A and B, that are further than the transmission range r_t apart, transmit during the same slot, there could be a collision at node C that is within the interference range r_i of both A and B. Listening only for complete transmissions with range r_t is insufficient to



(a) Inactive node A with sensing range $r_{si} = r_t$ will sense messages of active node B and never become an active node. Node A may sense messages of node C, but it will not sense messages of node D.

(b) Inactive node A with sensing range $r_{st} = r_i$ will sense messages of nodes B, C, and D.

Fig. 2. Illustration of effect of different carrier sensing ranges at an inactive node A. (The transmission, interference, and sensing ranges are drawn as idealized circles for ease of explanation; the model allows for non-uniform ranges as detailed in Section 2.)

avoid this problem, because nodes such as A and B that cannot reliably communicate can still cause collisions. Instead, our algorithm uses another strategy to extend the effective listening range: leader nodes set their certain-sensing range $r_{st} = r_i$, and non-leader nodes, also listening with certain-sensing range $r_{st} = r_i$, alert the leader nodes of sensed transmissions. As a result, at a range of $2r_i$ between active leaders in the final TDMA assignment, interference is impossible.

The hidden terminal problem has been extensively studied, and proposed solutions include physical carrier sense, virtual carrier sense, and out-of-band control. Virtual carrier sense (an RTS/CTS-style handshake) is relatively effective for unicast transmissions. But for broadcast transmissions that solution becomes unacceptably complicated as we must wait for acknowledgments from an indeterminate number of nodes. Our use of variable sensing ranges is an efficient tactic in this case because it makes use of existing capabilities while keeping the algorithm relatively simple.

More specifically, the Phase II algorithm uses variable sensing ranges as follows: initially every leader chooses a random round, setting its threshold such that $r_{st} = r_i$, with the goal of eventually becoming the sole "owner" of a round. Leaders transmit a message claiming their round in the second slot while non-leaders sense with a range $r_{st} = r_i$ and report back collisions to all leaders within the interference range r_i during the fourth slot. Leaders will therefore be able to hear about collisions in their round caused by any other leader within an $r_i + r_i = 4r_t$ range through two carrier sense communication hops. The first slot is used by leaders who have successfully become owners to send an ownership message in their active round. For this exchange, non-leaders sense with a range $r_{st} = 2r_i$ and respond in the third slot, notifying leader nodes in the volatile, non-owner state that the current round is occupied. This larger range ensures that leaders in the owner state remain owners. In both cases, when a leader hears about a collision or an occupied round, it randomly selects a new round. This algorithm soon converges to a stable set of non-interfering leader transmission round assignments.

In Phase III, these non-interfering leader transmission rounds are exploited to time the transmissions of leader and gateway node information so as to minimize collisions.

3.3. *Connecting leader nodes through gateway nodes (Phase III)*

The third and final phase is selection of gateway nodes that connect nearby leaders. For a given leader, every leader within a $3r_t$ range may be reachable by relaying messages through up to two gateway nodes. This implies that the spanner has a maximum *stretch factor* of 5: the shortest path between two nodes using only spanner links is no more than 5 times the shortest path using all network links. (In our extensive simulations presented in the next section we never encountered an actual stretch factor larger than 1.4.) Both leader and non-leader nodes maintain caches of local gateway information, which eventually converge to lists of gateway and leader nodes that will be used to route messages between nearby leaders.

More specifically, non-leader nodes initiate a four step process proceeding over four slots. In the first slot, each non-leader sends with a certain probability a ping to locate an active leader within transmission range r_t. If a leader responds with an acknowledgment in the second slot, the non-leader prepares gateway advertisements for the third and fourth slots. The third slot is used to pass leader information to other non-leaders, while the fourth slot is used to pass gateway information to the leaders themselves. Non-leaders that randomly choose not to send the initial ping listen during this slot. In a provably bounded number of rounds, all nodes have complete gateway information with high probability.

4. Performance evaluation

We present results for six simulation studies conducted for static ad hoc wireless networks with a custom-built simulator based on OMNeT++: (A) a study of how sensitive spanner stabilization time is to the probability p used in Phase I; (B) a study of the degree to which non-uniformity (as defined by the parameter δ) affects stabilization time; (C) a study of frame length; (D) a study of spanner construction; (E) a study of spanner resilience; (F) a study of the broadcast mechanisms developed for the spanner; and (G) a study of the convergecast mechanisms for the spanner.

4.1. Effect of p value used in Phase I on stabilization time

In several slots, our algorithm behaves probabilistically with respect to the timing for sending messages. A message for spanner construction may or may not be sent in a given slot depending on certain probabilities. In general, this probabilistic behavior is used to mix up the sets of senders and receivers for a given slot. A node that could not hear a message in one slot because it was sending will, with high probability, hear it later. In these cases, the probabilities are part of a strategy for multiple access.

The only such probability that was not explicitly fixed in [1] is the value of p used in slot 2 of Phase I. In this slot, active leader candidates send a message with probability p and consequently listen with probability $1-p$. The value of p that can be analytically proven to perform well, with high probability, for all network sizes is $p = 1/d$. The constant d is equal to the maximum number of *leader* nodes within the interference area of a node, which is bounded for any arbitrary network as shown shortly in this section. When p is chosen this way, the convergence time for Phase I is guaranteed to remain roughly the same, even as the network size and the network density increase. However, this "optimal" value of p does not necessarily guarantee the minimum stabilization time for a given network size, merely that the time will be bounded for all networks. In fact, we expect that values of p significantly larger than $1/d$ work well in most cases.

Fig. 3 shows the results from simulations that illustrate this effect. Consistent with typical simulation parameter settings in the literature, we consider a 200×200 m square network area with $N = 100$–900 nodes, each with transmission range $r_t = 30$ m. Furthermore, following the typical wireless propagation characteristics (see Section 2), we set the interference range to twice the transmission range, i.e., $r_i = 60$ m (and correspondingly $r_{si} = 2r_{st}$). For all considered p values, the stabilization time for Phase I is relatively stable for the considered wide range of node densities; especially for the mid-range of p, Phase I stabilization

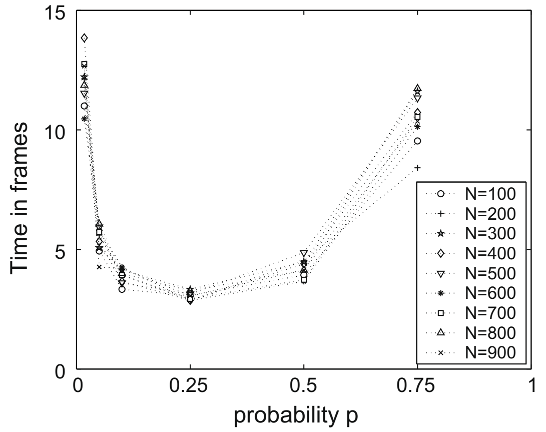


Fig. 3. Stabilization time for Phase I for various values of p and N. The first data point is $p = 0.017 \approx 1/60$.

time is independent of the network density. For our simulation setting, we have estimated a value of $d = 60$, as explained shortly. The shortest stabilization times occur for $p = 0.25$, rather than for $p = 1/60$. Since the times for Phase I stabilization comprise only a small fraction of the overall spanner stabilization time, and neither the p value nor the Phase I stabilization time affect the subsequent phases, we arbitrarily select the mid-range value $p = 0.5$ for the subsequent simulations.

The value of $d = 60$ was estimated for our simulation scenario as follows: By definition, d is the maximum number of leader nodes that can be within the interference range r_i of a node. If we consider the case in our simulation where $r_i = 2r_t$ is a uniform circle, and leaders may be as close as $r_s = r_t/2$ to each other, then the maximum value of d can be obtained by viewing this as a disk packing problem. Since disk packing traditionally deals with non-overlapping disks, we note that the problem of placing nodes no closer than r_s is equivalent to packing non-overlapping disks with radius $r_s/2$. Also, since a node at the maximum distance of r_i has half of its transmission range beyond r_i, our bounding circle has a radius of $r_i + r_s/2 = 9r_s/2$. Therefore, the problem is equivalent to packing unit disks into a circle of radius 9. In this case, the densest known packing (a *curved hexagonal* packing [53]) allows 61 disks to fit inside the larger circle. But our simulation is also bounded by the number of rounds in a frame, $k = 60$. Since each leader node in the spanner must claim a unique non-interfering round, the algorithm cannot create more than 60 stable leaders in the defined radius.

The constant d thus estimated is invariant with respect to network size, node density, and any other topology characteristic. Again, in [1], we formally prove that the p chosen according to the constant d indeed works for any network topology, and the simulation results validate the theoretical results of [1].

4.2. Non-uniformity and δ effects of sensing and interference ranges

Our spanner construction algorithm is based on a physical model that allows for two types of non-uniformity. The first type of non-uniformity involves a range where interference or carrier sensing may occur with arbitrary probability. These ranges are $(0,\ r_i)$ and (r_{st}, r_{si}), respectively. In order to conduct a worst-case analysis of interference for this type of non-uniformity, we let all transmissions within $(0,\ r_i)$ interfere. For carrier sensing, the situation is somewhat different. Multiple overlapping transmissions may increase the probability of detecting a busy carrier within the arbitrary range (r_{st}, r_{si}). As a simulation model for the carrier sensing range, we require at least two nodes to be transmitting within the (r_{st}, r_{si}) range in order for sensing to occur. (Only one transmitting node is required within the $(0, r_{st})$ range.) This simple model takes into account the effect of multiple transmissions on carrier sensing, and can be further refined in future work to simulate received power levels with finer granularity.

The second type of non-uniformity involves a non-uniform shape for each of these ranges: r_t, r_i, r_{st}, and r_{si} are each defined in terms of a cost function $c(u, v)$ which may take on values that are some factor δ larger or smaller than the Euclidean distance d between the nodes u and v. For a detailed discussion on how the cost function defines the transmission, interference, and sensing ranges we refer to [1]. The worst case for this type of non-uniformity occurs when δ takes on large values (close to 1). Separate simulations for different δ values are not necessary, as the effect of scaling up δ is equivalent to scaling up the node density as explained in the following.

If the cost function takes on values larger and smaller than the Euclidean distance with roughly equal probability, then the expected average transmission area is the average of the area of the maximum and minimum transmission ranges corresponding to the cases when $c(u, v) = (1 - \delta)d$ and $c(u, v) = (1 + \delta)d$, respectively. More specifically, the average area of two discs with radii $r_t/(1 + \delta)$ and $r_t/(1 - \delta)$. This average area is equal to $\pi r_t^2(1 + \delta^2)/(1 - \delta^2)^2$, which is a monotonically increasing function for $\delta \in [0, 1)$. Thus, increasing δ effectively increases the expected average transmission area.

Increasing the expected average transmission area has the effect of increasing D, the largest number of nodes within the transmission range of any node. The stabilization time and (by extension) byte overhead are bounded by D, which we can think of as the "maximum density" of the network. Therefore, we expect increasing δ to have a similar effect on overhead as scaling up D. In particular, our experiments that scale up the density of the network (in terms of increasing the total number of simulated nodes N in a fixed area) show the equivalent effects of scaling up D.

To demonstrate this effect, we conduct a verification experiment with $N = 300$ nodes. We model the effect of the cost function and δ in the following manner: all edges with length $d \leqslant r_t/(1 + \delta)$ exist in the network graph, and edges with $r_t/(1 + \delta) < d < r_t/(1 - \delta)$ exist with probability 1/2. Note that the cost function is a general model that can accommodate a wide range of propagation models; we have chosen this model since it exhibits high variability and is reasonably simple to simulate. We observe from Fig. 4 that the stabilization times indeed increase with increasing values of δ; these increases of the stabilization time are well within the increases in stabilization time observed for increasing N values in Fig. 10d.

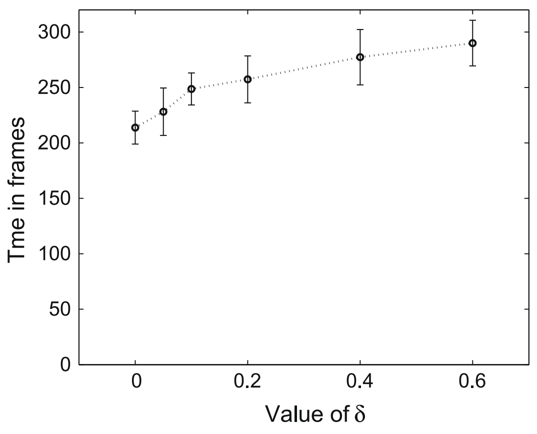


Fig. 4. Stabilization time for increasing values of δ for $N = 300$; the plotted 95% confidence intervals are obtained from 20 independent replications.

4.3. Frame length and measurement of overhead and information exchange

The spanner construction process makes use of both carrier sensing and packet transmission for information exchange between nodes. Additionally, carrier sensing allows the algorithm to form contention-free time slots for each leader node, greatly reducing the number of packet transmissions that are subject to collision. In order to measure the number of information bits exchanged through each mechanism, we use the rules described below for each communication slot. Each round consists of 10 slots, as illustrated in Fig. 1.

The duration of the individual slots depend on the information contained in them, which is either presence/absence of sensed physical carrier (neglected in Byte count), node address (counted as 6 Bytes), or time stamp (counted as 2 Bytes); in addition, we count 6 Bytes of overhead for each slot. Thus, the lengths of the control messages sent in the individual slots and the resulting phase lengths are as described below. See Table 1 for a summary.

During Phase I, the spanner algorithm uses a simple signal (indicating the presence of a transmission) in the first slot, and a message containing a node ID in slot 2. This gives $6 + 12 = 18$ Bytes. Note that all messages sent during Phases I and II only require carrier sensing, so these signals represent 1 bit of information sent for each transmitting node and 1 bit received for each sensing node. Similarly, during Phase II, the spanner algorithm uses a message containing an ID and a timestamp during the first slot, and simple signals during the other three slots. This gives $14 + 6 + 6 + 6 = 32$ Bytes for Phase II. The constant factor in the algorithm's running time could potentially be reduced by only using the minimal 6 Byte messages throughout Phases I and II, giving a total of $6 \times 6 = 36$ Bytes for both phases. However, the messages received in slot 2 of Phase I and slot 1 of Phase II can be used to collect

Table 1
Summary of slot lengths and information exchanged

Phase	Slot	Message vulnerable to collision	Exchanged information (bits/msg)	Message length (Bytes)
I	1	No	1	6
	2	No[a]	1 or 48[a]	12
II	1	No[a]	1 or 64[a]	14
	2	No	1	6
	3	No	1	6
	4	No	1	6
III	1	Yes	96 or 1[b]	18
	2	No	50 or 98	18
	3	Yes	112	20
	4	No	208n	6 + 26n

[a] Phase I slot 2 and Phase II slot 1 can be used to collect information in advance for Phase III, provided the messages do not collide. When this mechanism is used in simulation, the larger number of information bits is recorded.

[b] In Phase III slot 1, active leader nodes may only sense a busy or free carrier when the number of client nodes transmitting is not equal to 1. This carrier sensing determines the leaders' reply type in the next slot.

information in advance for Phase III, reducing the overall stabilization time. This refinement is incorporated into our simulation experiments and evaluations of information exchanged count these Bytes. Our stabilization time estimates are based on the longer slot lengths.

At the start of Phase III, assuming the previous phases have stabilized, each leader owns a contention-free time slot. However, intra-cluster transmissions from non-leader nodes are still subject to collisions. Therefore, the first two slots of Phase III contain a built-in contention mechanism. Non-leader nodes compete to receive an ACK message from the active leader in response to their CLIENT messages. Therefore, we measure the overhead needed for this slot by counting the total number of CLIENT messages sent versus the number of CLIENT messages that receive ACKs. The non-acknowledged CLIENT messages are not retransmissions in the usual sense because not all CLIENT messages need to be received in order for the spanner to stabilize. The 12 Bytes of information sent in each CLIENT message only need to be received from a subset of the non-leader nodes. On the leader's side, if only one client message is received, 12 Bytes of information are received; and if several client messages collide, 1 bit of information is sensed. Including the header, the total length of this slot is 18 Bytes.

In the second slot of Phase III, the leader replies to its non-leader nodes with one of three types of responses (ACK, COLLIDE, or FREE), which are of variable size. The header must therefore contain 2 bits of information to distinguish response type, and depending on the type, an additional 6 or 12 Bytes of payload information are sent and received. Because it is sent by a leader node, this transmission is not subject to collision. In any case, the total length of the slot is 18 Bytes, assuming the 2 bits of message type are absorbed into the header.

In the third slot of Phase III, a non-leader node which has received an ACK may send an ADV message containing 14 Bytes of topology information to other non-leaders.

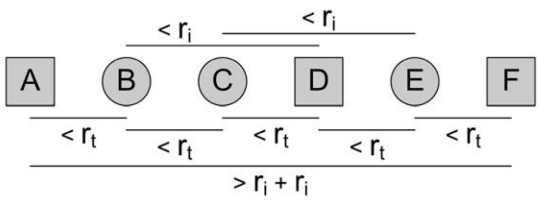


Fig. 5. Three leader nodes (gray squares A, D, and F) and three regular nodes (gray circles B, C, and E) with the relative distances depicted may result in a collision in Phase III slot 3.

Because these transmissions are between non-leaders, they are subject to collisions. Consider the example illustrated in Fig. 5, with nodes B and E having been authorized to send ADV messages by the leader nodes A and F, respectively. (Note that it is possible for both leader nodes A and F to be active in the same round because they are more than $r_i + r_i$ apart. Leader node D must be using a different active round.) Node C is one potential recipient of the ADV broadcast from B, but the reception at node C is subject to possible interference from node E. However, while collisions are possible in this slot, the spacing of active leaders established in Phase II greatly reduces the number of actual collisions (see Table 2 in Section 4.4.2).

In the fourth slot of Phase III, the node which sent an ADV will also broadcast a GATEWAY message to leader nodes. A GATEWAY message can always be received by the leader node that authorized it, due to the $r_i + r_i$ spacing of active leader nodes. Therefore, these messages are not subject to collisions. The size of the gateway message depends on the number n of 26 Bytes entries it contains. We observed in our simulations (see Fig. 6) average maxima ranging from 6 to 11.25 entries when slot 4 was used with network sizes ranging from 100 to 800 nodes, and we conservatively set $n = 12$ for our delay estimates.

We calculated the values in Fig. 6 using the following method. In each frame, the simulation averaged the number of gateway entries over all nodes and all rounds. Then, we took the maximum of these samples over all frames of all 20 runs for each value of N. Since these are maximum and not mean values, we do not include confidence intervals.

The duration of a frame also depends on the k, the number of 10-slot rounds in each frame. When choosing a good value for k, there is a tradeoff between high values of k which give nodes a large number of "colors" to use in the distributed coloring of Phase II, and low values of k which shorten the frame length. We observe in Fig. 7 that values of $k < 60$ do not allow spanner construction to complete in dense networks (large N with fixed area). For values of $k \geqslant 60$, the time it takes the spanner to stabilize generally increases as the frame length increases. Therefore, to minimize stabilization time, we should choose the smallest value of k that allows spanner construction to complete in dense networks. For the values we examined, the number of rounds per frame that meets these criteria is $k = 60$, which we use in all other simulations.

Based on a value of $k = 60$, a 1 Mbit/s transmission rate and a 20 μs spacing between slots, a frame is 215.5 ms long. We note that our algorithms could be further refined

Table 2
Average number of transmitted (tx), sensed (sx), collided, and received (rx) messages per node.

N	100	200	300	400	500	600	700	800	900
Phase I									
# of msgs tx	40 ± 16	32 ± 17	24 ± 8.6	15 ± 9.6	14 ± 5.6	12 ± 3.9	11 ± 4.3	12 ± 5.2	8.8 ± 3.1
# of msgs sx	296	478	509	412	482	487	495	636	474
=# of bits sx	± 112	± 244	± 190	± 266	± 213	± 175	± 208	± 295	± 199
# of msgs rx	0.10	0.063	0.046	0.026	0.028	0.024	0.021	0.014	0.016
	± 0.02	± 0.008	± 0.008	± 0.007	± 0.002	± 0.003	± 0.003	± 0.004	± 0.002
# of bits rx	4.9 ± 0.95	3.0 ± 0.37	2.2 ± 0.37	1.3 ± 0.34	1.3 ± 0.12	1.1 ± 0.15	1.0 ± 0.14	0.68 ± 0.17	0.77 ± 0.11
Phase II									
# of msgs tx	188	271	330	333	321	362	505	389	666
	± 78	± 107	± 77	± 131	± 110	± 80	± 239	± 148	± 279
# of msgs sx	1204	1416	1551	1520	1451	1593	2078	1647	2586
=# of bits sx	± 433	± 474	± 311	± 495	± 425	± 300	± 842	± 526	± 920
# of msgs rx	20 ± 8.2	27 ± 11	32 ± 7.7	33 ± 13	31 ± 11	35 ± 7.6	49 ± 22	38 ± 15	65 ± 27
# of bits rx	1259	1740	2072	2082	2002	2268	3111	2452	4162
	± 528	± 692	± 490	± 823	± 688	± 489	± 1440	± 944	± 1741
Phase III slot 1									
# of msgs tx	40 ± 5.0	110 ± 13	173 ± 16	257 ± 24	334 ± 26	379 ± 35	424 ± 49	510 ± 51	538 ± 71
# of msgs sx									
=# of bits sx	14 ± 1.8	23 ± 2.5	25 ± 2.1	26 ± 2.2	26 ± 1.9	25 ± 1.9	23 ± 2.6	24 ± 2.2	21 ± 2.4
# of msgs rx	38 ± 4.6	52 ± 5.5	47 ± 4.4	43 ± 3.7	40 ± 3.7	36 ± 3.8	31 ± 3.5	31 ± 2.9	26 ± 3.1
Kilobits rx	3.7 ± 0.44	5.0 ± 0.52	4.5 ± 0.43	4.1 ± 0.35	3.8 ± 0.35	3.4 ± 0.36	3.0 ± 0.34	2.9 ± 0.28	2.5 ± 0.30
# of collisions	8.1 ± 0.93	17 ± 1.9	20 ± 1.7	21 ± 1.8	21 ± 1.8	21 ± 1.6	20 ± 1.6	20 ± 1.9	18 ± 2.1
Phase III slot 2									
# of msgs tx	24 ± 3.0	34 ± 3.7	36 ± 3.0	36 ± 2.9	35 ± 2.5	33 ± 2.6	30 ± 3.3	31 ± 2.8	27 ± 3.1
# of msgs rx	16 ± 2.0	70 ± 8.1	116 ± 9.2	157 ± 13	188 ± 14	207 ± 18	218 ± 25	254 ± 24	245 ± 29
Kilobits rx	1.3 ± 0.15	4.1 ± 0.46	6.3 ± 0.50	8.3 ± 0.70	10 ± 0.74	11 ± 0.94	11 ± 1.3	13 ± 1.2	13 ± 1.5
Phase III slot 3									
# of msgs tx	9.98 ± 1.2	12.0 ± 1.3	10.9 ± 0.92	10.1 ± 0.82	9.03 ± 0.67	8.04 ± 0.72	7.01 ± 0.72	7.04 ± 0.66	6.01 ± 0.68
# of msgs rx	34 ± 4.2	104 ± 11	157 ± 13	204 ± 17	236 ± 17	257 ± 22	265 ± 28	306 ± 29	298 ± 34
Kilobits rx	3.8 ± 0.47	12 ± 1.2	18 ± 1.4	23 ± 1.9	26 ± 2.0	29 ± 2.5	30 ± 3.2	34 ± 3.3	33 ± 3.8
# of collisions	0.04 ± 0.01	0.03 ± 0.009	0.18 ± 0.08	0.068 ± 0.02	0.026 ± 0.009	0.36 ± 0.1	0.19 ± 0.08	0.16 ± 0.07	0.17 ± 0.08
Phase III slot 4									
# of msgs tx	9.98 ± 1.2	12.0 ± 1.3	10.9 ± 0.92	10.1 ± 0.82	9.03 ± 0.67	8.04 ± 0.72	7.01 ± 0.72	7.04 ± 0.66	6.01 ± 0.68
# of msgs rx	9.98 ± 1.2	12.0 ± 1.3	10.9 ± 0.92	10.1 ± 0.82	9.03 ± 0.67	8.04 ± 0.72	7.01 ± 0.72	7.04 ± 0.66	6.01 ± 0.68
Kilobits rx	28 ± 4.0	50 ± 6.1	54 ± 5.4	55 ± 5.3	53 ± 6.6	48 ± 6.2	42 ± 4.9	46 ± 5.7	39 ± 5.2
Phase III totals									
# of msgs tx	84 ± 10	168 ± 19	230 ± 21	313 ± 29	386 ± 30	428 ± 38	468 ± 54	554 ± 55	577 ± 75
# of msgs sx									
=# of bits sx	14 ± 1.8	23 ± 2.5	25 ± 2.1	26 ± 2.2	26 ± 1.9	25 ± 1.9	23 ± 2.6	24 ± 2.2	21 ± 2.4
# of msgs rx	98 ± 11	238 ± 25	331 ± 27	414 ± 34	472 ± 35	508 ± 44	521 ± 57	598 ± 56	576 ± 67
Kilobits rx	37 ± 4.9	71 ± 8.1	82 ± 7.6	90 ± 8.1	94 ± 10	91 ± 10	86 ± 9.5	96 ± 10	87 ± 10
Kilobits rx not vulnerable to collision	30 ± 4.1	54 ± 6.5	60 ± 5.8	63 ± 6.0	63 ± 7.4	59 ± 7.0	53 ± 6.1	58 ± 6.8	52 ± 6.6
Totals									
# of msgs tx	312	471	584	645	721	803	985	929	1252
	± 82	± 127	± 82	± 151	± 116	± 82	± 217	± 171	± 283
# of msgs sx	1515	1916	2085	1936	1958	2105	2596	2305	3081
=# of bits sx	± 480	± 712	± 483	± 731	± 617	± 423	± 892	± 790	± 998
# of msgs rx	118 ± 11	265 ± 28	363 ± 25	426 ± 57	504 ± 36	543 ± 45	570 ± 51	606 ± 85	641 ± 72
Kilobits rx	38 ± 4.8	73 ± 8.1	84 ± 7.5	88 ± 12	96 ± 9.5	93 ± 9.8	89 ± 9.0	94 ± 14	91 ± 11

in future work to reduce the number of gateway entries, thus shortening the frame duration.

4.4. Performance results for spanner construction

To demonstrate the performance of spanner construction, we present two types of results: (i) detailed sample-path data providing insights into the evolutions of the different phases of our algorithm, and (ii) aggregate data providing insights into the overall algorithm performance, both for 100 up to 900 node networks. For each simulation we conducted 20 independent runs, each starting with an independent random placement of nodes without any information about neighbors or topology in a 200×200 m square field. We use a transmission range of $r_t = 30$ m and an interference range of $r_i = 60$ m.

4.4.1. Sample-path simulations

We plot in Fig. 8a–c the means for the number of leader nodes determined by Phase I, the number of leader nodes owning non-interfering transmission rounds determined by Phase II, and the number of gateway nodes determined

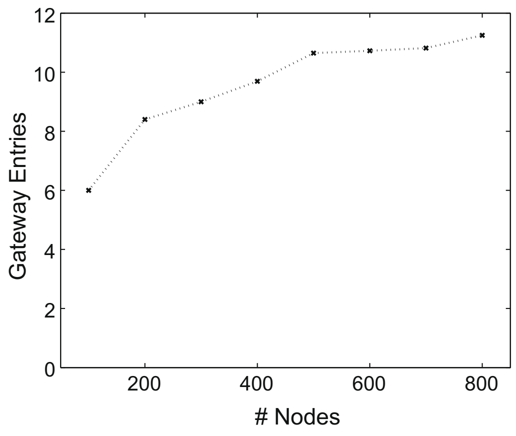


Fig. 6. Average maximum number of GATEWAY entries when slot 4 is used.

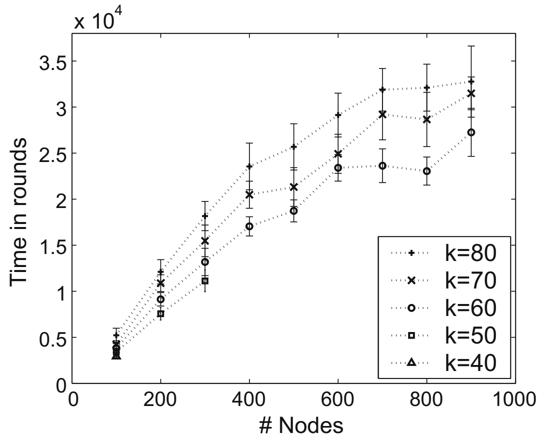


Fig. 7. Stabilization time for various values of number of rounds per frame k and number of nodes N (95% confidence intervals from 20 replications).

by Phase III of our algorithm, respectively, as a function of time in frames. Each line depicts data for a different number of nodes, ranging from 100 at the bottom to 900 at the top.

We observe from Fig. 8a and b that the leader nodes and the ownership of non-interfering transmission rounds are completely stabilized in all cases within approximately 10 and 15 frames, respectively, whereas the gateway nodes stabilize hundreds of frames later. Specifically for the $N = 300$ node scenario, the leader nodes and the ownership of non-interfering transmission rounds stabilize within about two and eight frames, respectively, whereas the gateway nodes stabilize after close to 215 frames. We therefore conclude that for this scenario (increasing nodes in a fixed area), the completion times for Phases I and II are small and relatively constant. On the other hand, we observe from Fig. 8c that the time it takes for Phase III to complete is relatively large and increases with the number of nodes/density. We also note that most of the gateway nodes have been discovered well before the spanner is completely stable. If we consider it sufficient that 90% or 95% of possible gateway nodes are established, the overall stabilization time is much smaller. Specifically, as examined in detail in Fig. 10d, for the $N = 300$ scenario, the backbone is completely constructed after about 46.4 s, while 90% of the gateway nodes have been discovered after approximately 86 frames (18.5 s), whereas 95% and 98% of the gateway nodes are established after 127 frames (27.3 s) and 162 frames (34.9 s), respectively, indicating that the spanner is close to complete at these earlier times.

Comparisons with existing backbone formation algorithms are complicated due to a number of crucial differences and can therefore only give a rough indication of the relative performance of our algorithm. For instance, two extensively studied algorithms in [31], namely an algorithm based on [15,33] and an algorithm based on [34], assume that each node has a complete list of other nodes in its transmission range, whereas we do not assume any prior neighbor or topology knowledge by the nodes. Creating such neighbor lists requires neighbor discovery through the exchange of hello messages and is examined in recent studies, see e.g., [54–59], which report time durations for completing the neighbor discovery ranging from several seconds to a minute, with delays on the order of 10s of seconds being most typically reported. Also, our approach produces a backbone network in conjunction with a schedule of non-interfering (MAC packet collision-free) transmission rounds for the leader nodes in the backbone network thus greatly facilitating the use of the backbone, whereas the exiting approaches only identify the nodes in the backbone, but do not further facilitate the use of the backbone. For 300 nodes with neighbor lists and a 30 m transmission range in a 200 by 200 m area, [31] reports average protocol durations of a little less than 2 s for the [15,33] based algorithm and about 12 s for the [34] based algorithm.

Comparing the overall strategies for constructing a backbone starting from a network of nodes without any topology information, we note that the [15,33,34] based approaches in [31] first employ a neighbor discovery protocol to obtain the neighbor lists at each node, and then construct the backbone relatively quickly based on the detailed node address information in the neighbor lists. On the other hand, our approach starts from nodes without any neighbor or topology information and first employs physical carrier sensing (I) to establish leader nodes that can reach every node in the network in one hop, and (II) to assign these leader nodes non-interfering transmission rounds. With this "infrastructure" which is put in place very quickly, as observed from Fig. 8a and b, we then (in our Phase III) exchange detailed node address information to learn the detailed local topology for the establishment of the gateway nodes. This detailed node address exchange in our Phase III makes Phase III by far the most time consuming phase, as observed in Fig. 8c, but is aided by the underlying "infrastructure" that minimizes collisions for the long control messages with the detailed gateway information sent in the third and fourth slot of Phase III.

The comparison study [31] also considers an algorithm based on [35], which requires nodes to know the number of neighbors (obtained through techniques such as [60]),

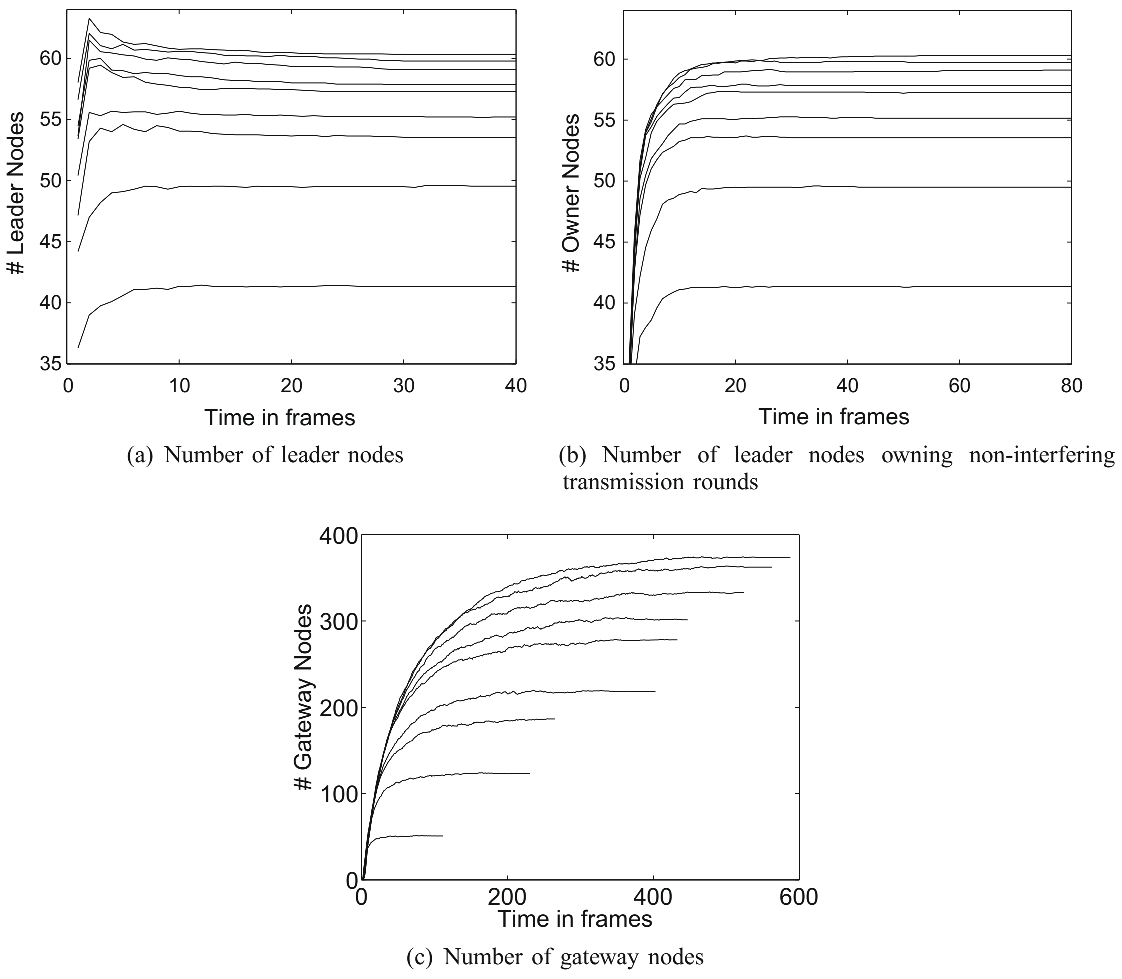


Fig. 8. Detailed sample-path simulations: evolution of numbers of leader and gateway nodes backbone creation as a function of time in frames for 100 (bottom curve), 200, 300,...,900 nodes (top curve).

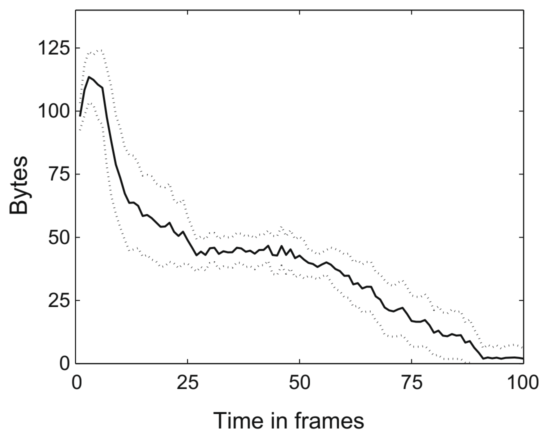
and reports a duration of 77 s for the 300 node network. The [35] based algorithm requires the propagation of control information over several transmission hops, resulting in a low degree of localization, and involves the construction of a spanning tree. In contrast, our approach is highly localized and does not involve spanning trees, resulting in significantly faster backbone construction in a network of nodes without any topology information.

The parallel phases in our self-stabilizing algorithm ensure a robust backbone that automatically recovers from changes in the network, such as might occur due to node mobility and node failure. In other words, our algorithm has built-in functionality for both creating and maintaining a backbone network. For a reasonably fair comparison of the overhead (in terms of number of transmitted Bytes per node) with existing backbone creation algorithms, we report in Fig. 9a–f the means and 95% confidence intervals of the numbers of per-frame and cumulative transmitted Bytes contributing toward backbone creation. Specifically, we count the transmissions of Phase I, from the start of the simulation until the leaders are stable. Similarly, we count the transmissions of Phase II from the start of the simulation until the round ownerships are stable. Then we count the Phase III transmissions from when the round ownerships are stable until the gateways are stable. We observe from Fig. 9a, c, and e a brief spike in the per-frame transmissions for the completion of the leader election and non-interfering transmission round assignment. The transmissions then slowly taper off as the gateway nodes are found. Referring back to Fig. 8c, we note that most of the gateways are discovered early in the stabilization process.

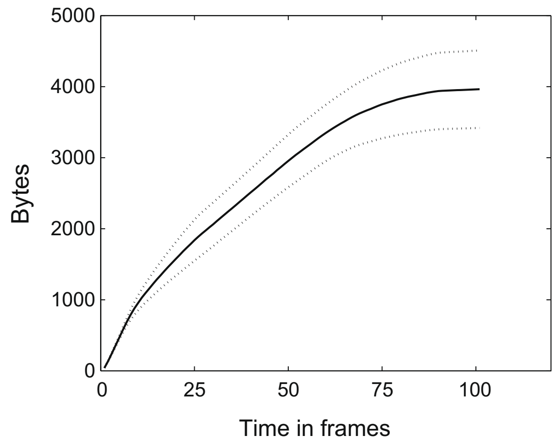
Comparing Fig. 9b, d, and f we observe that the trend is similar for large and small networks: a brief period where transmissions occur rapidly and a longer period where transmissions are less frequent, eventually tapering off. We also note that the rate of transmissions during gateway discovery, indicated by the slope in these figures, is almost identical for $N = 300$ and $N = 700$. This trend will be investigated further in the next section.

4.4.2. Aggregate simulations

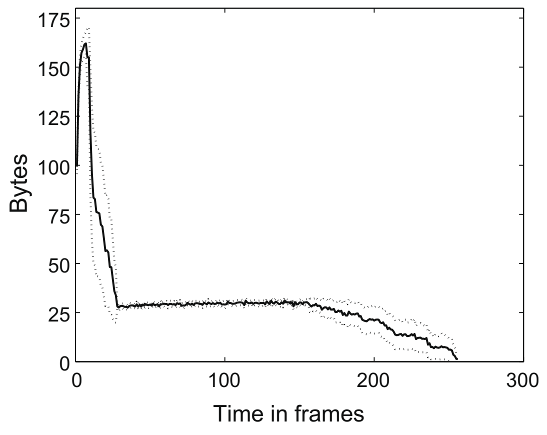
In this section we present aggregate simulation results characterizing the created backbone network and the effort required for creating the backbone. We consider net-



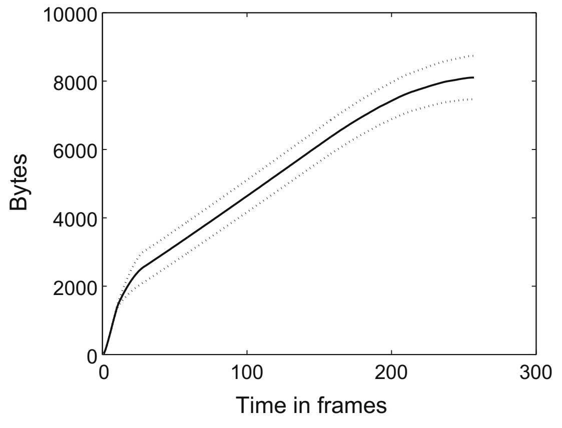
(a) Number of transmitted Bytes by a node for backbone creation per frame for $N = 100$



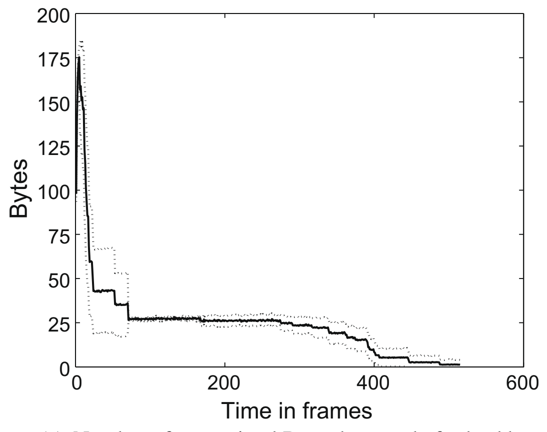
(b) Cumulative number of transmitted Bytes by a node for backbone creation for $N = 100$



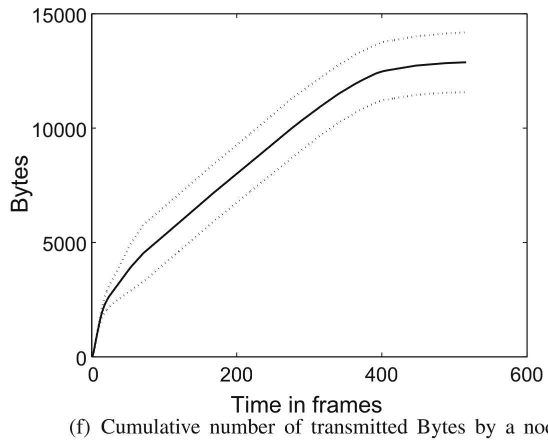
(c) Number of transmitted Bytes by a node for backbone creation per frame for $N = 300$



(d) Cumulative number of transmitted Bytes by a node for backbone creation for $N = 300$



(e) Number of transmitted Bytes by a node for backbone creation per frame for $N = 700$



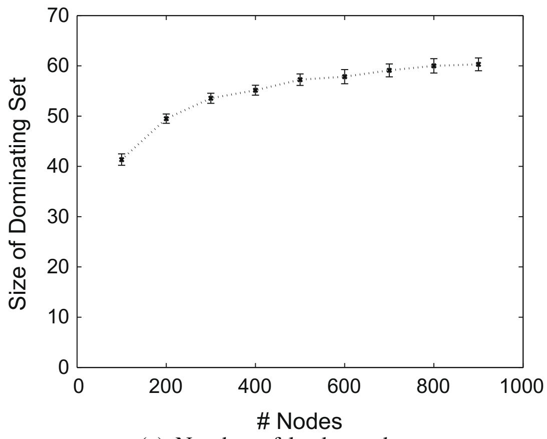
(f) Cumulative number of transmitted Bytes by a node for backbone creation for $N = 700$

Fig. 9. Detailed sample-path simulations: evolution of number of transmitted Bytes for backbone creation as a function of time in frames for 100, 300, and 700 nodes.

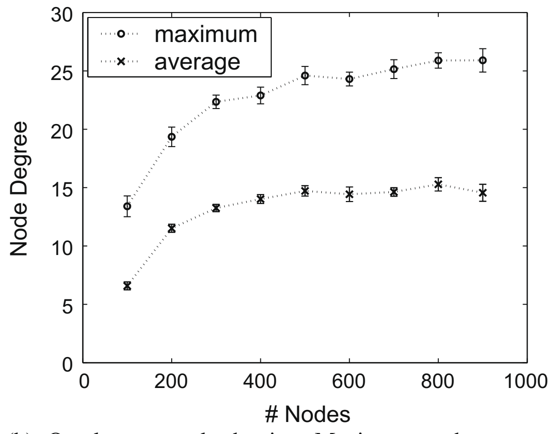
works with 100–900 nodes, whereby each node has the transmission and interference ranges $r_t = 30$ m and $r_i = 60$ m (and $r_{si} = 2r_{st}$). Each data point in the following plots represents the average of 20 runs, each with independent random node placement, with the error bars representing 95% confidence intervals. In these aggregate simulations, we define a spanner to be stable in terms of its sample-path data from the previous section. Intuitively, we define a spanner to be stable when the remainder of the curve shown in Fig. 8c is "flat". Specifically, we allow each run to continue until the time t_1 when the status of each node (leader/owner state and gateway list length) has remained unchanged for 10 frames. In general, this overestimates the stabilization time. We then calculate the true stabilization time to be the time t_s such that the variance from the mean for the number of gateway nodes from t_s to t_1 is less than 0.01.

In Fig. 10a we plot the mean and 95% confidence intervals for the number of leader nodes. In Fig. 10b we plot the node degree defined as the number of connections each leader has to other leaders through attached outgoing gateway links. Smaller leader set size and node degree indicate that the overlay network will be able to distribute data messages more efficiently, with fewer total transmissions. We observe that the number of leader nodes (Fig. 10a) and the number of neighbors each leader node has in the backbone (Fig. 10b) initially both increase, then level off toward a maximum for denser networks. This means that the quality of the backbone network generated by our algorithm will not degrade in highly dense networks. The node degrees achieved in our simulations are in the same range as those in the simulations in [25]. For a 100 node unit-disk model network with similar area and transmission range as our network, the algorithms in [25] give average node degrees between 1.2 and 4 and maximum node degrees between 4 and 41, compared to average and maximum node degrees of approximately 6.6 and 13.4 with our algorithm. This is one indication that our algorithm constructs a fairly efficient backbone, while considering a more detailed physical layer model.

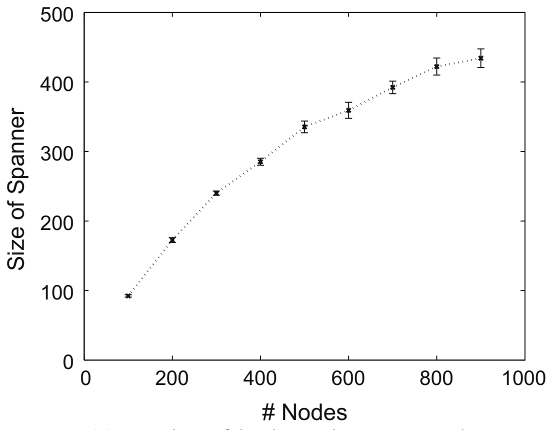
In Fig. 10c, we plot the total size of our spanner, including leader nodes and gateway nodes. These represent large



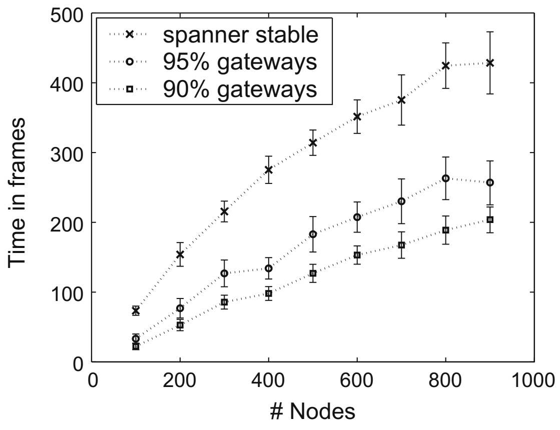
(a) Number of leader nodes



(b) Overlay network density: Maximum and average n umber of outgoing gateway node links at a leader node



(c) Number of leader and gateway nodes



(d) Time efficiency: Average length of time (in frames) to complete spanner formation. Using our time estimate of 1 frame = 215.5 ms, the 800 node network stabilizes in ≈85 s.

Fig. 10. Aggregate simulation: backbone network characteristics as well as required backbone formation time for 100–900 nodes.

percentages of the total network size and large numbers of gateway nodes compared to other spanner construction algorithms. The backbones for the 300 node networks in [31] contain 60–75 nodes, compared to on average 54 leader nodes and 186 gateway nodes for a total of 240 nodes in the backbone with our algorithm. One main reason for our relatively high number of gateway nodes is that in an effort to keep complexity low and topology information exchanges local, our algorithm does not force two leader nodes A and B to use the same gateway node(s) for communication in both directions between A and B. More specifically, from additional simulations for 300 node networks we found that for almost a third of all A↔B connections, the A→B connection has one gateway node while the B→A connection has two gateway nodes. Typically, one third of such A↔B connections do not share any gateway node, i.e., employ three unique gateway nodes to connect leaders A and B in both directions. Furthermore, typically more than half of all A↔B connections employ two gateway nodes in each direction, whereby in turn more than half of such connections traverse at least one different gateway node in the different directions. For close to a tenth of all connections, node A identified unique nodes C and D as its gateway path to B, while B identified unique nodes E and F as its gateway path to A. Moreover, roughly 4/5 of all one-node connections use a different gateway in each direction. These results indicate that integrating a gateway "pruning" algorithm, e.g., [33], is one possibility for future work on this topic. However, we also note that redundant gateway connections make the spanner more robust to node loss and do not hinder the broadcast or convergecast algorithms, as we explore in subsequent sections.

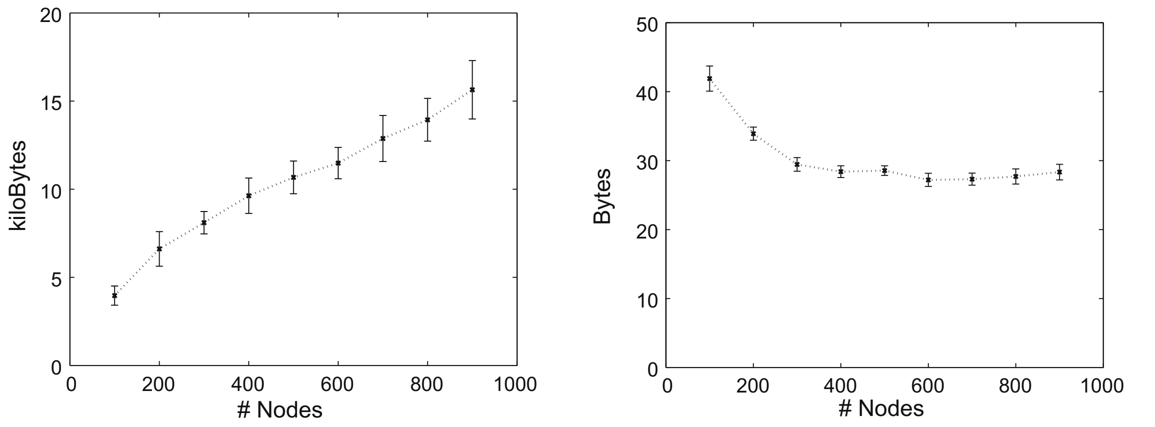
In Fig. 10d, we plot the stabilization time for the spanner construction alongside graphs of the points in time when 90% and 95% of the total gateways have been discovered. We observe that the stabilization time increases somewhat slower than linearly with the network size, which matches our theoretically shown poly-logarithmic complexity results [1]. This type of efficiency is important in any backbone formation algorithm, but stabilization time is especially important when nodes become mobile. As indicated in the previous section, the 90% and 95% lines show that the spanner is mostly complete well before it is completely stable. Using our time estimate of 1 frame = 215.5 ms, even the 800 node network is 90% complete in approximately 40 s.

In Fig. 11a, we plot the total number of Bytes transmitted per node for the backbone creation during the stabilization of the spanner. For better comparison with algorithms that do not run in parallel, we only count packets sent during "active" phases of the algorithm (as in the previous section). We observe that similar to stabilization time, communication cost increases somewhat slower than linearly with the network size. Again, this matches our theoretically shown poly-logarithmic complexity results [1]. Specifically, for $N = 300$, we observe from Figs. 9d and 11a a cumulative overhead of $8\ 104 \pm 606$ (for 95% confidence interval) transmitted Bytes per node for the backbone creation, which compares to less than 1000 Bytes with the [15,33] based approach, roughly 7500 Bytes with the [34] based approach, and 36,000 Bytes with the [35] based algorithm in [31]. It is very important to keep in mind in these comparisons that [31] considers interference ranges equal to transmission ranges, whereas we consider interference ranges twice as large as the transmission ranges. To illustrate the impact of the larger interference range, we simulated the $N = 300$ node network with equal transmission and interference ranges $r_t = r_i = 30$ m for which the cumulative overhead is roughly cut in half to $3\ 475 \pm 525$ Bytes per node for the backbone creation. Further, with the $r_t = r_i$ setting, the stabilization times for the leader nodes and round ownerships are slightly reduced, while the gateway nodes still require around 215 frames to stabilize. It is also important to note in the comparisons that the algorithms examined in [31] assume knowledge of neighbor lists or number of neighbors, whose establishment requires significant transmission overhead [54–59]. In contrast, our algorithms do *not* require any prior knowledge. Since the energy consumed for the overlay network construction is generally proportional to the exchanged control traffic, the results in Fig. 11a indicate that the energy consumed to set up the overlay network increases close to linearly with the network size.

In Fig. 11b, we plot the average number of Bytes transmitted per node per frame during Phase III of the spanner construction. As mentioned in the previous section, even as we increase network size, the rate of Bytes sent per node is relatively constant during this phase. The rate is actually higher for sparser networks ($N < 400$) because a larger percentage of the total nodes are being identified as gateways.

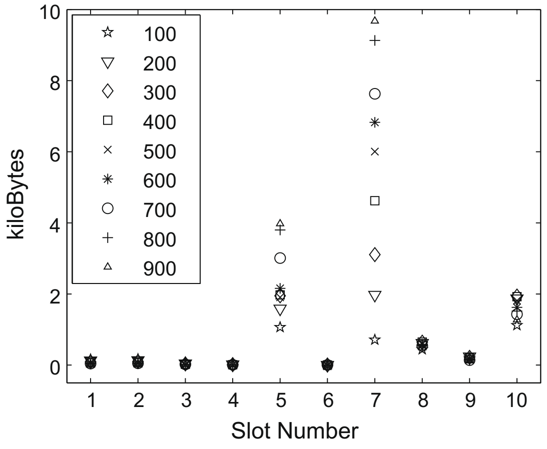
In Fig. 11c, we plot the relative use of each slot during spanner formation. We observe relatively small numbers in most slots, since only a small fraction of nodes use these slots. The exceptions are slots 5 and 7. Slot 5 is the third slot of Phase II, and all inactive nodes within r_i of an owner node broadcast a busy message. As in all other plots, only messages sent during "active" phases of the spanner construction are counted. If we counted Phase II maintenance activity during the entire spanner formation time, slot 5 and not slot 7 would represent the largest fraction of communication overhead. Slot 7 is the first slot of Phase III, when potential gateway nodes contend for the attention of active leader nodes. Unlike slot 5, only inactive nodes covered by leaders in their active round transmit in this slot, and fewer Bytes are transmitted during slot 7 each round. However, the relatively long duration of Phase III allows this slot to contribute more toward the total overhead in the long run. Slot 10 is used to transmit large gateway list messages, but the contention mechanism of Phase III effectively limits the number of nodes sending these in each round.

In Table 2, we present the average number of transmitted, sensed, collided, and received messages per node, broken down by phases and slots. For each statistic, messages are totaled over the time period each phase is active during spanner construction, averaged across 20 runs, and divided by the total number of nodes in the network. For this analysis, we define a collision to be an event in which a node that could have received a message is only able to sense



(a) Communication cost: Average number of Bytes sent per node during spanner creation

(b) Communication cost: Average number of Bytes sent per node per frame during gateway discovery



(c) Cost per slot: Average number of Bytes sent per slot per node during spanner creation

Fig. 11. Aggregate simulation: backbone formation overhead for 100–900 nodes.

a busy carrier. We emphasize that these collisions do not imply retransmissions: spanner formation merely depends on the fact that sufficient topology information is received with high probability.

Also, in Table 2, we compare information reception that is due to physical carrier sensing only versus fully receiving a message by expressing both quantities in bits. While the majority (over 75%) of messages sent are used for carrier sensing, each of these messages conveys only one bit of information. The larger gateway discovery messages in Phase III comprise the majority (over 90%) of information bits received.

Examining the numbers of messages transmitted for the three phases in Table 2 for increasing number of nodes N, we observe a decrease for Phase I, while there is an increase for Phase II and the totals for Phase III. The decreasing number of transmitted Phase I messages for increasing node density is due to the probabilistic leader election mechanism of [1], which scales back the number of messages transmitted per node, despite using the same p value for all considered network densities. On the other hand, the increasing numbers of Phases II and III messages for increasing node density conform with the poly-logarithmic complexities of these algorithms [1]. It is interesting to observe that the increase in the messages for Phase III is mainly due to slot 1 of Phase III; the numbers of transmitted messages for slot 2 stay roughly constant, while the numbers of transmitted slot 3 and 4 messages (which are the longest messages exchanged by the algorithms slightly decrease with increasing node density).

We further observe from the results for Phases I and II in the table that these phases rely largely on sensed information. Recall from Section 4.3 that Phases I and II only require sensed information for forming the leader set and the non-interfering transmission rounds, but through intact message reception can collect information in advance for Phase III. We observe that Phase I contributes very little toward the information collection for Phase III, while Phase II makes substantial information collection contributions.

Examining closer the transmitted and received messages in slots 3 and 4 of Phase III, we observe that the number of messages transmitted in slot 3 is equal to the number of transmitted messages in slot 4, which in turn is equal to the number of received messages in slot 4. These results confirm the expected behavior of the algorithms in that any node that sends a so-called ADV message in slot 3, sends a so-called GATEWAY message in slot 4 [1]. Furthermore, a slot 4 GATEWAY message is sent from a gateway node to its leader node, and is not vulnerable to collision. A possible refinement over the algorithm specified in [1] could be achieved by having other inactive leader nodes "overhearing" the GATEWAY message transmission from a gateway node to its leader. However, additional simulations (not included in detail here due to space constraints) indicated that this refinement does not significantly reduce stabilization time.

Turning to the amount of control information exchanged through physical carrier sensing relative to the information exchanged through intact message receptions we make the following observations from Table 2. First, we observe from the total numbers of messages sensed and messages received that over 75% of the messages are sensed (out of the total of sensed plus received messages). Furthermore, we observe from the Phase III totals that the majority (typically over 65%) of all bits received (from received messages) in Phase III are sent during slots that are not vulnerable to collision. These results confirm that the output of Phase II's carrier sensing operations, namely a non-interfering schedule of leader transmission rounds, helps to reduce overhead in Phase III by greatly reducing the number of potential collisions. Furthermore, actual collisions are also reduced by the schedule of leader transmission rounds, as seen in slot 3 of Phase III.

4.5. *Performance results for spanner resilience*

To test the resilience of the spanner we conducted two types of simulations: one evaluating the passive resilience of the spanner, without any spanner maintenance; and another evaluating active resilience, measuring how well the spanner maintenance function of the algorithm performed. Throughout we continued to use the same simulation settings as before. We simulated 100–900 nodes with the transmission and interference ranges $r_t = 30$ m and $r_i = 60$ m (and $r_{si} = 2r_{st}$).

4.5.1. *Passive resilience*

To examine the passive resilience of the constructed backbone network, we followed the approach of Basagni et al. [31] and their "robustness" metric: we measured the number of randomly selected nodes in the backbone that could be removed while maintaining its two main properties: (i) being a connected network, and (ii) being a dominating set of the overall network. The backbone is defined to be connected when a path exists between every pair of nodes in the backbone. The backbone is defined to be dominating when every node in the network is at most one hop away from a backbone node. While this robustness metric allows us to make quantitative comparisons with existing results, it is not directly applicable to our network model since we first construct a set of leader nodes that dominate the network, and then add gateway nodes that connect the leaders but have no role in dominating inactive nodes. Our gateway nodes would need to be "promoted" to leader status in cases where the local leader node was disabled. We allow for this type of "easy repair" of the spanner in our definition of "passive" resilience, so that the robustness metric of [31] becomes applicable to our model.

Table 3 first gives the average total number of nodes in our backbone, as well as the respective numbers of leader and gateway nodes, prior to disabling any nodes. Throughout, we report 95% confidence intervals based on 75 replications. Next, the table gives the average total number of randomly selected backbone nodes that could be disabled while the remaining backbone nodes still formed a functional backbone; the split of the disabled nodes into leader and gateway nodes is also provided. Table 3 furthermore gives the average number of "easy repairs", i.e., promotions of gateway nodes to leader nodes, required to maintain the functionality of the backbone after the disabling of backbone nodes. Comparing the numbers for overall backbone size and disabled nodes, we observe that for all network sizes a large fraction of the nodes in the backbone can be lost while maintaining the connected and dominating properties. However, unlike the original spanner, this "damaged" backbone would not be guaranteed to have a stretch factor of 5, and would require active maintenance to regain full capability. Easy repairs alone only handle the loss of the dominating property, and cannot guarantee these other properties of the spanner.

Table 3
Backbone robustness: the number of nodes that can be disabled in a backbone network without compromising its connectedness or dominating property, for various network sizes. Easy repairs represent instances when a gateway node was promoted to act as a dominating node.

N	100	200	300	400	500	600	700	800	900
Nodes in bb.	92 ± 1.7	173 ± 3.5	240 ± 3.0	285 ± 5.0	335 ± 8.5	359 ± 12	392 ± 9.0	422 ± 12	437 ± 15
# leaders	41 ± 1.1	50 ± 0.93	54 ± 1.0	55 ± 0.99	57 ± 1.1	58 ± 1.4	59 ± 1.3	60 ± 1.4	61 ± 1.7
# gateways	51 ± 1.5	123 ± 3.1	186 ± 2.8	230 ± 3.9	278 ± 7.8	301 ± 11	333 ± 8.4	362 ± 11	376 ± 14
Avg. disabl.	8 ± 1.7	59 ± 5.8	114 ± 6.4	169 ± 8.3	213 ± 7.8	253 ± 7.2	267 ± 9.9	250 ± 15	300 ± 9.3
# leaders	3.9 ± 0.88	18 ± 1.7	27 ± 1.6	32 ± 1.8	37 ± 1.5	40 ± 1.2	38 ± 1.7	39 ± 2.3	42 ± 1.3
# gateways	4.3 ± 0.93	41 ± 4.2	87 ± 5.0	137 ± 6.77	176 ± 6.5	213 ± 6.4	228 ± 8.4	211 ± 13	257 ± 8.3
Easy repairs	0.283	3.507	9.867	16.32	21.11	24.83	26.99	26.95	33.67
% of fail.									
Not conn. (%)	93.3	94.7	80.0	77.3	62.7	68.0	60.0	41.3	45.3
Not domin. (%)	6.7	5.3	20.0	22.7	37.3	32.0	40.0	58.7	54.7

Compared to the other protocols with a high degree of localization evaluated in [31], our algorithm produces a large backbone: approximately 80% of the nodes in our $N = 300$ node network belong to the backbone, while 20–25% of the nodes belong to the backbone with the protocols of [15,33–35]. However, our spanners are also significantly more robust: almost half of the nodes in our backbone can be lost while maintaining spanner properties. On the other hand, the spanners produced by the algorithms in [15,33–35] cannot generally survive the loss of more than 8 backbone nodes in the $N = 300$ node network. Importantly, as we will see in Section 4.6, our broadcast and convergecast algorithms efficiently use only a fraction of the gateway nodes, effectively reducing the size of the active backbone.

From Table 3, we also find that the relative number of failures due to loss of connectivity versus loss of dominating set property decreases for larger network sizes. As pointed out in Section 4.4.2, our spanner does not force two leaders to use the same set of gateways for communication between the two in both directions. In a denser network, there is greater possibility of a leader node uncovering a few inactive nodes when it gets disabled, while the backbone has a higher chance of remaining connected through the higher number of gateways.

4.5.2. Active resilience

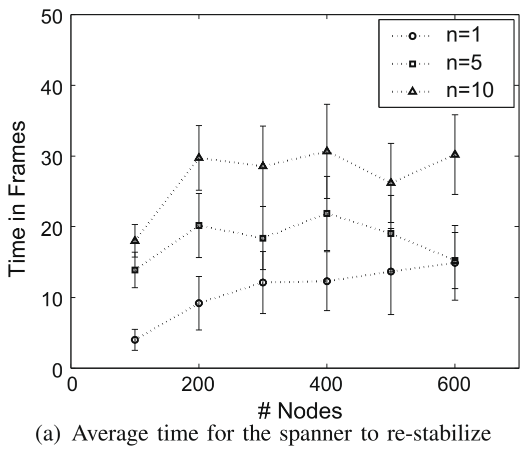
We present in this subsection a brief preliminary evaluation of the active resilience of our spanner, a more comprehensive active resilience evaluation is left for future work. We disable a number of the nodes in the backbone and measure the time it takes for the spanner to re-stabilize. Compared to our passive resilience scenario, this is a more demanding condition than merely requiring the spanner to be connected and a dominating set after removing nodes. Here, we are measuring the time until the spanner is fully repaired with all the possible gateway connections that contribute to the low stretch factor property of the backbone. We expect this process to take some time, even when we remove fewer nodes than the previous section's results seem to permit.

For this evaluation, we focus on the removal of backbone nodes. In our spanner formation, broadcast, and convergecast algorithms the leaders do not maintain cluster membership lists. Therefore, regular nodes may join, leave, or move within the coverage area of different leader nodes without incurring any repair costs to the backbone.

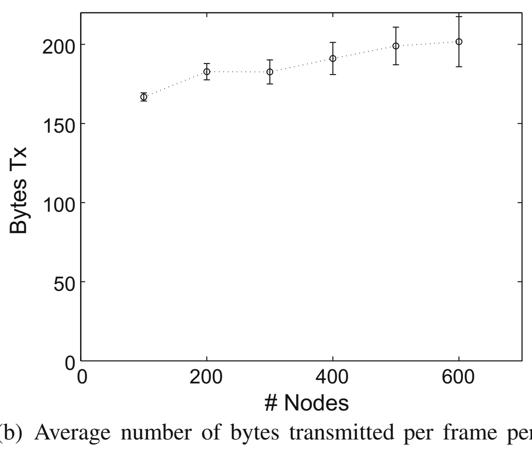
Fig. 12a shows the average time (95% confidence intervals based on 15 replications) it takes the spanner to restabilize is primarily dependent on the number of removed nodes n and not the total size of the network N. These results demonstrate that the spanner repair is conducted mainly locally, involving only the immediate neighborhood of the removed nodes and not the entire network. Fig. 12 shows the average per-node overhead during spanner maintenance (95% confidence interval based on 45 replications), which is independent of the number of disabled nodes. We see that the overhead for active repair is roughly equal to the peak spanner construction overhead of 200 Bytes per frame per node. This is consistent with the earlier results since we are now counting messages for all three phases of the spanner formation algorithm as they run in parallel to repair the backbone. We note that by extending the data period in the round (see Fig. 1) whereby a frame contains k rounds, i.e., k data periods, the relative overhead for spanner maintenance can be reduced, at the expense of reduced maintenance responsiveness. A detailed examination of these trade-offs of active resilience is left for future work.

4.6. Broadcast

To evaluate the broadcast mechanism, we sequentially simulate the broadcasting of single-packet messages from one source node to all other nodes in the network. We conduct 90 independent replications and present 95% confidence intervals. For each replication, the source node is chosen at random from among all network nodes, includ-



(a) Average time for the spanner to re-stabilize



(b) Average number of bytes transmitted per frame per node during spanner repair, which is independent of the number of disabled nodes

Fig. 12. Backbone robustness: time and Bytes spent while the spanner re-stabilizes after $n = 1, 5, 10$ nodes have been disabled.

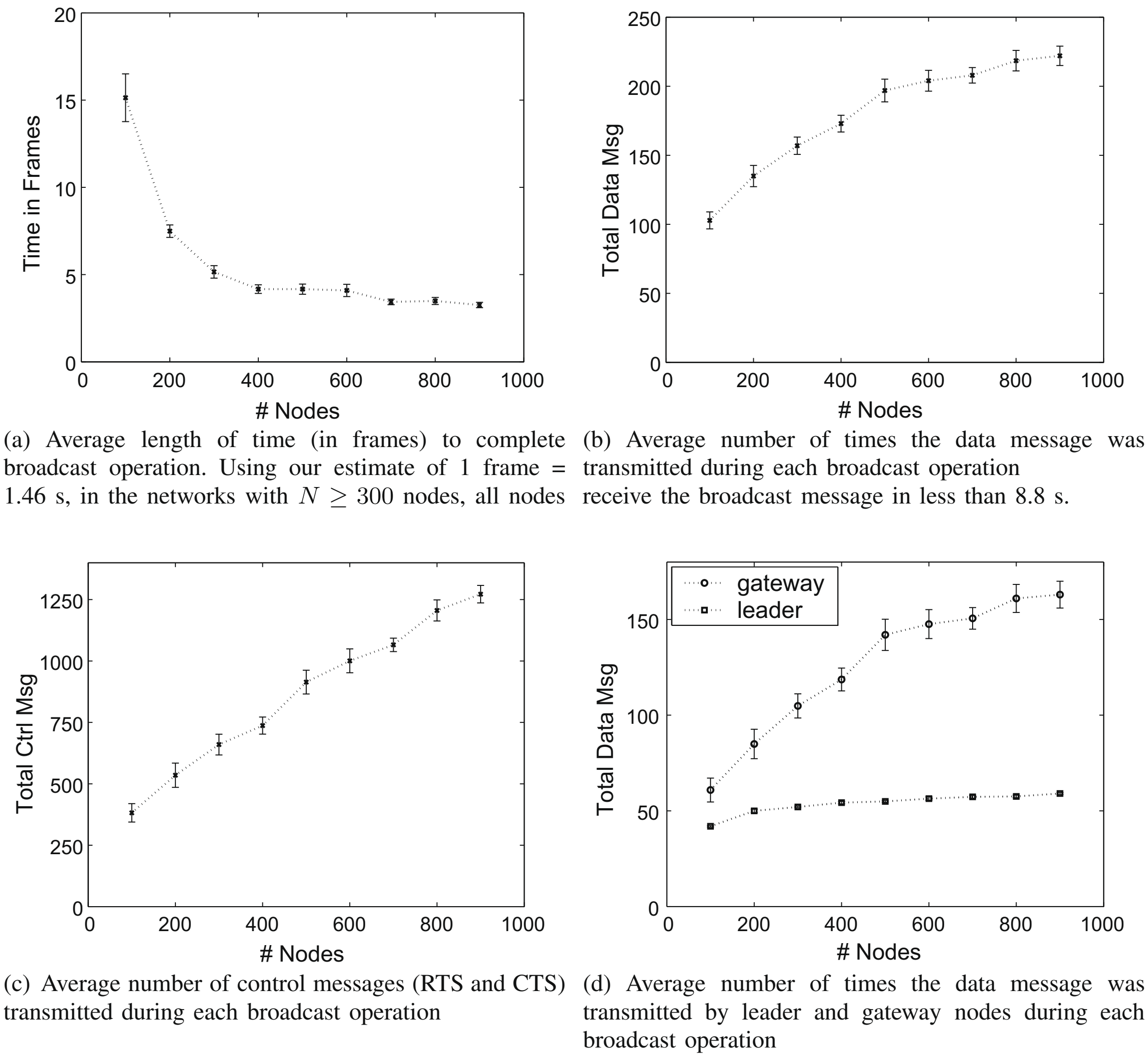
ing leader, gateway, and inactive nodes. As before, we simulated 100–900 nodes with the transmission and interference ranges $r_t = 30$ m and $r_i = 60$ m (and $r_{si} = 2r_{st}$).

A broadcasting frame contains k four-slot rounds, using the non-interfering round assignments from Phase II. The first slot is reserved for leaders to forward the broadcast message. The second and third slots are used by gateway nodes to probabilistically initiate an RTS-CTS exchange. Gateway nodes that have received a broadcast data message send an RTS with probability p. The fourth slot is reserved for gateways to forward the message after a successful CTS. We proved in [2] that selecting $p = 1/2d$ guarantees bounded time to deliver the message to all nodes, where d is the number of leaders within the interference range r_i of any node. Since we estimated in Section 4.1 a maximum possible value of $d = 60$, we conservatively set $p = 0.01$ for these simulations.

If we consider the maximum size for a broadcast message of 1500 Bytes, and that the RTS and CTS frames are 20 and 14 Bytes, respectively (as in 802.11), then the duration of a broadcast frame can be estimated as in Section 4.3. Based on $k = 60$ rounds/frame, a 1 Mbit/s transmission rate and a 20 μs spacing between slots, a broadcast frame is approximately 1.46 s long.

In Fig. 13a, we see that the time it takes for the message to reach all nodes is relatively constant for dense networks and is somewhat higher for sparse networks. While our small value of p results in lower performance for sparse networks, it does establish a density-independent upper bound on broadcast latency. Using our estimate of 1 frame = 1.46 s, the broadcast message reaches all nodes in our $N \geqslant 300$ node networks in less than 8.8 s. Our results are comparable with those in [13] in that both algorithms achieve constant broadcast latency for fixed-area networks, and they therefore approximate the minimum latency.

Figs. 13b and c show the average number of times the data message or control messages (RTS and CTS) were transmitted, respectively. We note that the overhead levels off for large networks, which corresponds to a decreasing overhead/node ratio for denser networks.



(a) Average length of time (in frames) to complete broadcast operation. Using our estimate of 1 frame = 1.46 s, in the networks with $N \geq 300$ nodes, all nodes receive the broadcast message in less than 8.8 s.

(b) Average number of times the data message was transmitted during each broadcast operation

(c) Average number of control messages (RTS and CTS) transmitted during each broadcast operation

(d) Average number of times the data message was transmitted by leader and gateway nodes during each broadcast operation

Fig. 13. Broadcasting overhead for $N = 100$–900 nodes.

Finally, Fig. 13d shows the average number of times the broadcast message was transmitted by leader nodes and gateway nodes during broadcast operations. The broadcast protocol allows leader nodes to transmit exactly once, but a given gateway node may retransmit the same broadcast packet multiple times. Thus, the number of data packet transmissions plotted in Fig. 13d gives the number of leader nodes and an upper bound on the number of gateway nodes involved in the broadcast. More specifically, Table 4 gives the average number of backbone nodes involved in a given broadcast. Here we see an important property of the broadcast mechanism. Even though the $N = 900$ node networks have on average 437 backbone nodes, on average only 188 of these nodes are used during a broadcast. This effectively mitigates the negative effects of selecting a large number of gateway nodes, because only a fraction are active in broadcasting.

Comparisons of our physical carrier sensing based broadcast mechanism with existing broadcast approaches are complex, mainly because the existing approaches do not consider interference ranges larger than transmission ranges and also do not conduct the broadcast over a self-stabilized (active resilience) backbone. To give some indications of the performance of existing algorithms, we note that the most efficient of the broadcast protocols studied in [15] also uses a fraction of the nodes in a network that decreases with increasing network density. Their "gateway" protocol uses only 45% of all network nodes for broadcasting in a network with $degree = 10$. In our $N = 200$ node network, which is a similar degree, we use 52% of all nodes to achieve network-wide broadcast. In [8], several broadcast algorithms using varying amounts of topology and location information are evaluated through simulation. The densest network evaluated has a degree of 21.2 (roughly the density of our $N = 400$ node network) and between 18 and 35 of 110 nodes retransmit the broadcast message for the various protocols, compared to 36% of the nodes in our $N = 400$ network.

We also note that the transmissions by leader nodes in our backbone, which account for close to half of the transmissions for broadcasts in small networks and about one quarter of the transmissions in large networks, are not subject to collisions due to the non-interfering round assignments from Phase II. More specifically, in slot 1, leaders relay the data message in their reserved active round, which means that all neighboring nodes can receive it without interference. In slot 2, gateway nodes with the broadcast packet send an RTS message with probability p, which may result in a collision. In slot 3, a leader or gateway that successfully receives an RTS sends a CTS signal, which the originating gateway receives through physical carrier sensing, and is thus not vulnerable to collision. In slot 4, only gateways that sense a CTS signal relay the broadcast message, again without collision. We observe from Table 4 that as a result of primarily our spanner infrastructure, less than 5% of the Bytes transmitted during broadcast are vulnerable to collisions.

4.7. Convergecast

We define convergecast to be a many-to-one operation in which each of m source nodes has a packet to send to a sink node. In our evaluation of the convergecast algorithm, for each replication, m source nodes and the sink are chosen uniformly at random from among all network nodes, including leader, gateway, and inactive nodes. We conduct 15 replications each for six random networks, for a total of 90 replications and report 95% confidence intervals. Throughout, we continue to use the same simulation settings as before, that is, we simulated 100–900 nodes with the transmission and interference ranges $r_t = 30$ m and $r_i = 60$ m (and $r_{si} = 2r_{st}$).

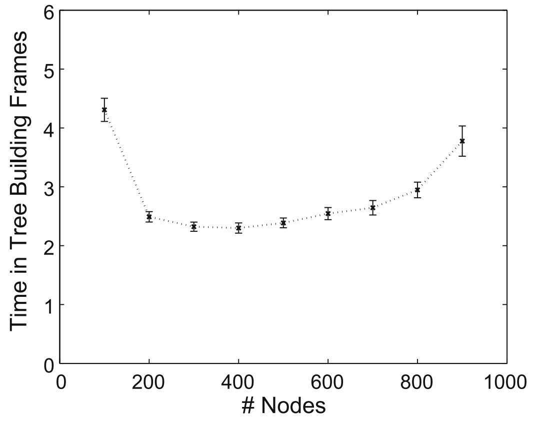
Convergecast proceeds in two phases. The first phase is a tree-building operation, which is basically the broadcast operation with a ROUTE message instead of a broadcast data message. Tree-building frames similarly contain k rounds, using the non-interfering round assignments from Phase II. The difference is that both leaders and gateways participate in the probabilistic sending of RTS messages whenever they receive a ROUTE message reporting a shorter path to the sink. We therefore need only three slots for tree building: one for sending an RTS with probability p, one for responding with a CTS signal, and one for sending ROUTE messages.

The second phase is gathering using the completed tree. Convergecasting uses its own four-slot round, again making use of the non-interfering round assignments. The first slot is reserved for leaders to forward the convergecast message. The second and third slots are used by gateway and inactive nodes to probabilistically initiate an RTS-CTS exchange. Non-leader nodes that have a convergecast data message in their queue send an RTS with probability p. The fourth slot is reserved for these non-leader nodes to forward a messages after a successful CTS. We proved in [2] that selecting $p = 1/d$ guarantees bounded time to deliver all messages to the sink. For contrast with our broadcast results, we aggressively set $p = 0.05$ for these simulations.

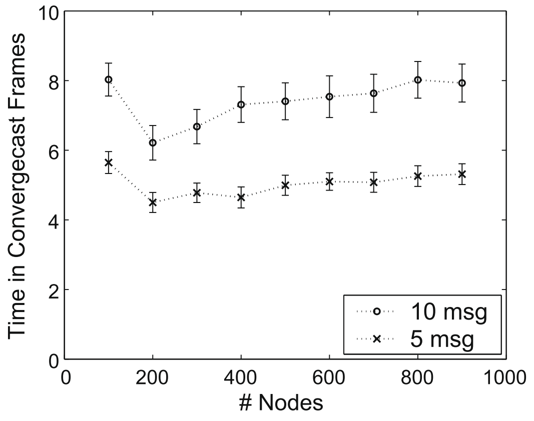
Unlike spanner construction, these two phases cannot run in parallel, since we assume the tree is complete before gathering begins. One possible distributed method to transition between these phases would be for source nodes to start a timer upon receiving a ROUTE message during tree building. With a timer duration set using estimates of tree building latency (see Fig. 14), the source node could be reasonably certain the tree was complete before gathering begins. It would also be possible to build a tree for a given

Table 4
Average number of nodes involved in a broadcast and average number of kBytes transmitted during a broadcast operation of a 1500 Bytes packet.

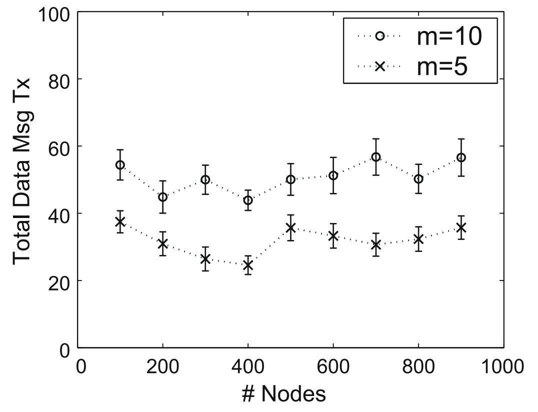
N	100	200	300	400	500	600	700	800	900
# of broadcast nodes	71 ± 1.1	106 ± 1.5	129 ± 2.8	143 ± 3.2	162 ± 3.7	166 ± 3.7	175 ± 3.3	183 ± 4.2	188 ± 4.6
kBytes tx not vuln. to coll.	155 ± 9.3	205 ± 12	239 ± 9.5	264 ± 9.1	301 ± 12	313 ± 11	320 ± 8.5	337 ± 11	344 ± 11
kBytes tx vuln. to coll.	6.1 ± 0.70	7.6 ± 0.92	8.3 ± 0.80	8.1 ± 0.62	9.6 ± 0.85	9.9 ± 0.90	9.3 ± 0.46	10.5 ± 0.78	10.2 ± 0.57



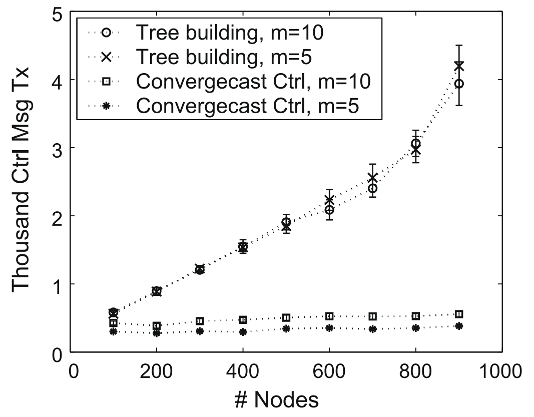
(a) Average length of time to complete tree building for convergecast. Using our estimate of 1 tree building frame = 29.5 ms, it takes less than 177 ms to build a tree for a given sink.



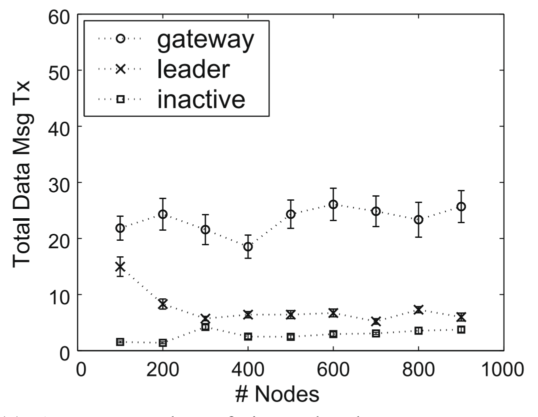
(b) Average length of time to complete convergecast over built tree for $m = 5$ and $m = 10$ sources. A convergecast frame contains $k = 60$ rounds of 4 slots each and is 1.46 s with our settings.



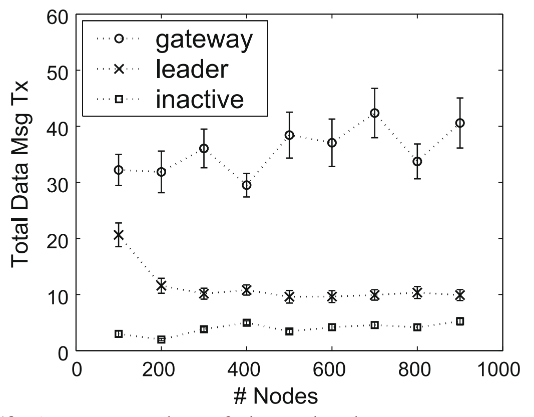
(c) Average number of times the data messages were transmitted during each convergecast operation



(d) Average number of control messages for tree building and convergecast transmitted during each convergecast operation



(e) Average number of times the data messages were transmitted by leader, gateway, and inactive nodes during each convergecast operation with $m = 5$ sources



(f) Average number of times the data messages were transmitted by leader, gateway, and inactive nodes during each convergecast operation with $m = 10$ sources

Fig. 14. Convergecasting overhead for $N = 100$–900 nodes.

sink and re-use it for several iterations of gathering, as long as there are no significant changes in topology.

To obtain an estimate of convergecast latency, we assume that the ROUTE message is a simple 20 Bytes control packet (it only needs to contain the address of the sender and an advertised number of hops to the sink), and that the size of the RTS, CTS, and data packets are 20, 14, and 1500 Bytes, respectively, as in broadcasting. Using similar assumptions, a tree building frame is 29.52 ms, and a convergecast frame is approximately 1.46 s.

In Fig. 14a, we plot the time it takes to build a convergecasting tree for each network size. Using our estimate of 1 frame = 29.5 ms, it takes less than 177 ms to build a tree for a given sink. For the $N \geqslant 200$ node networks, the estimated delay is less than 90 ms. In Fig. 14b, we plot the time it takes to gather messages from $m = 5$ and $m = 10$ source nodes over an established tree. The total delay for a convergecast is obtained by adding the tree building delay from Fig. 14a and convergecast delay from Fig. 14b. As indicated in our theoretical results [2], the time to complete convergecast depends both on the number of messages and the number of nodes.

In Figs. 14c and d, we compare the use of data messages and control messages, counting RTS, CTS, and ROUTE messages separately for the tree building and gathering phases. While the number of data message forwards appears to be relatively constant for each value of m, the number of tree building control messages increases roughly linearly with N and is nearly the same for both values of m. This is because tree building involves all nodes in the network and does not depend on m. On the other hand, the control message overhead for convergecasting is largely independent of the number of nodes N. Furthermore, the number of times we forward the larger data message toward the sink is limited—it depends mostly on the distance from source to sink [2].

We note that the overhead (time and data messages sent) to gather messages from $m = 10$ sources is significantly less than double the time to gather messages from $m = 5$ sources. This is very likely due to the fact that we are sampling the source–destination distance non-uniformly by choosing a new sink node and m new source nodes uniformly at random for each convergecast operation.

The data forwarding operations are further broken down in Figs. 14e and f, which show the types of nodes responsible for forwarding convergecast messages. We note that since the gathering tree is built using only the backbone nodes, inactive nodes will always have a backbone node as a parent in the tree. This effectively limits the number of data transmissions by inactive nodes to only the first hop of convergecasting. We also note that each node involved in the convergecast transmits a given data message only once.

5. Conclusion

We have evaluated the actual performance of recently proposed physical carrier sensing based backbone (spanner) construction algorithms [1] and broadcast/convergecast algorithms [2] for ad hoc networks through simulations for typical network scenarios; only asymptotic performance bounds existed previously. We have found that through judiciously adjusting the carrier sense threshold used in physical carrier sensing, our algorithms are able to accommodate interference ranges larger than transmission ranges, packet collisions at the MAC layer, and nodes without any prior knowledge of neighbors while achieving good network layer performance. Our algorithms achieve this performance by exchanging typically over 75% of the messages required for the overlay construction through physical carrier sensing. Physical carrier sensing is extensively exploited to quickly (in about 2.4 s in a typical 300 node network) find leader nodes that can reach all nodes with non-interfering transmission rounds. The exchange of detailed node address information for linking the leader nodes into a backbone network is the most demanding part of the backbone construction (requiring close to 47 s in a typical 300 node network) but is significantly facilitated by the non-interfering transmission rounds; less than 35% of the information bits for backbone formation are transmitted in packets that are vulnerable to collisions.

While the constructed backbones are relatively dense, e.g., the backbone contains up to 80% of the nodes in the 300 node network and close to 50% of the nodes in the 900 node network, the backbones are very robust, remaining functional when roughly half of the backbone nodes in the 300 node network and close to two thirds of the backbone nodes in the 900 node network fail. For the 900 node network, our broadcast mechanism uses less than about half of the backbone nodes, i.e., less than about one quarter of the network nodes, to deliver a 1500 Bytes broadcast message with 1 Mb/s transmission rate within approximately 8.8 s to all network nodes. Our convergecast mechanisms deliver data messages from 10 source nodes towards a common sink by transmitting the data messages through less than 10% of the nodes in a network with 500 or more nodes.

Directions for future work include evaluating the active resilience of the spanner in the context of mobile ad hoc networks as well as developing routing and route maintenance algorithms exploiting the physical carrier sensing with adjustment of the carrier sensing threshold.

Acknowledgement

Supported in part by the National Science Foundation through Grant Career ANI-0133252.

References

[1] K. Kothapalli, C. Scheideler, M. Onus, A. Richa, Constant density spanners for wireless ad hoc networks, in: Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Las Vegas, NV, July 2005, pp. 116–125.
[2] M. Onus, A. Richa, K. Kothapalli, C. Scheideler, Efficient broadcasting and gathering in wireless ad hoc networks, in: Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN), Las Vegas, NV, December 2005, pp. 346–351.
[3] A. Chiganmi, M. Baysan, K. Sarac, R. Prakash, Variable power broadcast using local information in ad hoc networks, Ad Hoc Networks 6 (5) (2008) 675–695. Jul..
[4] A. Keshavarz-Haddad, V. Ribeiro, R. Riedi, Color-based broadcasting for ad hoc networks, in: Proceedings of IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, April 2006, pp. 1–10.
[5] T.J. Kwon, M. Gerla, V.K. Varma, M. Barton, T.R. Hsing, Efficient flooding with passive clustering—an overhead-free selective forward

mechanism for ad hoc/sensor networks, Proceedings of the IEEE 91 (8) (2003) 1210–1220.
[6] W. Lou, J. Wu, Toward broadcast reliability in mobile ad hoc networks with double coverage, IEEE Transactions on Mobile Computing 6 (2) (2007) 148–163.
[7] W.-Z. Song, X.-Y. Li, Y. Wang, O. Frieder, W. Wang, Localized topology control for unicast and broadcast in wireless ad hoc networks, IEEE Transactions on Parallel and Distributed Systems 17 (4) (2006) 321–334.
[8] B. Williams, T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, in: Proceedings of ACM MOBIHOC, Lausanne, Switzerland, June 2002, pp. 194–205.
[9] Y. Yi, M. Gerla, T.J. Kwon, Efficient flooding in ad hoc networks: a comparative performance study, in: Proceedings of IEEE International Conference on Communications (ICC), May 2003, pp. 1059–1063.
[10] I. Chlamtac, O. Weinstein, The wave expansion approach to broadcasting in multihop radio networks, IEEE Transactions on Communication 39 (3) (1991) 426–433.
[11] R. Bar-Yehuda, O. Goldreich, A. Itai, On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization, Journal of Computer and System Sciences 45 (1) (1992) 104–126.
[12] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, Wireless Network 8 (2/3) (2002) 153–167.
[13] R. Gandhi, S. Parthasarathy, A. Mishra, Minimizing broadcast latency and redundancy in ad hoc networks, in: Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2003, pp. 222–232.
[14] H. Zhang, Z.-P. Jiang, Modeling and performance analysis of ad hoc broadcasting schemes, Performance Evaluation 63 (12) (2006) 1196–1215.
[15] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbors elimination-based broadcasting algorithms in wireless networks, IEEE Transactions on Parallel and Distributed Systems 13 (1) (2002) 14–25.
[16] J. Wu, F. Dai, Efficient broadcasting with guaranteed coverage in mobile ad hoc networks, IEEE Transactions on Mobile Computing 4 (3) (2005) 259–270.
[17] G. Jakllari, S. Krishnamurthy, M. Faloutsos, P. Krishnamurthy, O. Ercetin, A cross-layer framework for exploiting virtual MISO links in mobile ad hoc networks, IEEE Transactions on Mobile Computing 6 (6) (2007) 579–594.
[18] G. Jakllari, S. Krishnamurthy, M. Faloutsos, P. Krishnamurthy, On broadcasting with cooperative diversity in multi-hop wireless networks, IEEE Journal on Selected Areas in Communications 25 (2) (2007) 484–496.
[19] E.J. Duarte-Melo, M. Liu, Data-gathering wireless sensor networks: organization and capacity, Computer Networks 43 (4) (2003) 519–537.
[20] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, IEEE/ACM Transactions on Networking 11 (1) (2003) 2–16.
[21] A. Kesselman, D.R. Kowalski, Fast distributed algorithm for convergecast in ad hoc geometric radio networks, Journal of Parallel and Distributed Computing 66 (4) (2006) 578–585.
[22] Y. Yu, B. Krishnamachari, V. Prasanna, Energy-latency tradeoffs for data gathering in wireless sensor networks, in: Proceedings of IEEE INFOCOM, vol. 1, March 2004, pp. 244–255.
[23] S. Upadhyayula, V. Annamalai, S. Gupta, A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks, in: Proceedings of IEEE GLOBECOM, December 2003, pp. 3525–3530.
[24] Y. Zhang, Q. Huang, Coordinated convergecast in wireless sensor networks, in: Proceedings of IEEE Military Communications Conference (MILCOM), October 2005, pp. 1152–1158.
[25] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, O. Frieder, Geometric spanners for wireless ad hoc network, IEEE Transactions on Parallel and Distributed Systems 14 (4) (2003) 408–421.
[26] X. Cheng, D. Huang, W. Li, W. Wu, D.Z. Du, A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks, Networks: An International Journal 42 (4) (2003) 202–208.
[27] J. Gao, L. Guibas, J. Hershberger, L. Zhang, A. Zhu, Geometric spanners for routing in mobile networks, IEEE Journal on Selected Areas in Communications 23 (1) (2005) 174–185.
[28] Y. Wang, W. Wang, X.-Y. Li, Efficient distributed low-cost backbone formation for wireless networks, IEEE Transactions on Parallel and Distributed Systems 17 (7) (2006) 681–693.
[29] J. Wu, F. Dai, Virtual backbone construction in MANETs using adjustable transmission ranges, IEEE Transactions on Mobile Computing 15 (9) (2006) 1188–1200.
[30] M. Luby, A simple parallel algorithm for the maximal independent set problem, SIAM Journal on Computing 15 (4) (1986) 1036–1055.
[31] S. Basagni, M. Mastrogiovanni, A. Panconesi, C. Petrioli, Localized protocols for ad hoc clustering and backbone formation: a performance comparison, IEEE Transactions on Parallel and Distributed Systems 17 (4) (2006) 292–306.
[32] H. Liu, X. Jia, P.-J. Wan, X. Liu, F. Yao, A distributed and efficient flooding scheme using 1-hop information in mobile ad hoc networks, IEEE Transactions on Parallel and Distributed Systems 18 (5) (2007) 658–671.
[33] F. Dai, J. Wu, An extended localized algorithm for connected dominating set formation in ad hoc wireless networks, IEEE Transactions on Parallel and Distributed Systems 15 (10) (2004) 908–920.
[34] S. Basagni, Distributed clustering for ad hoc networks, in: Proceedings of International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), Perth, Australia, June 1999, pp. 310–315.
[35] P.-J. Wan, K. Alzoubi, O. Frieder, Distributed construction of connected dominating sets in wireless ad hoc networks, ACM Mobile Networks and Applications 9 (2) (2004) 141–149.
[36] F. Kuhn, T. Moscibroda, and R. Wattenhofer, Radio network clustering from scratch, in: Proceedings of European Symposium on Algorithms (ESA), 2004.
[37] S. Parthasarathy, R. Gandhi, Distributed algorithms for coloring and domination in wireless ad hoc networks, in: Proceedings of Foundations of Software Technology and Theoretical Computer Science, 2004, pp. 447–459.
[38] R. Bruno, C. Chaudet, M. Conti, E. Gregori, A novel fair medium access control for 802.11-based multi-hop ad hoc networks, in: Proceedings of the 14th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN), September 2005, pp. 1–6.
[39] J. Deng, B. Liang, P.K. Varshney, Tuning the Carrier Sensing Range of IEEE 802.11 MAC, in: Proceedings of IEEE Global Telecommunications Conference (Globecom), November/December 2004, pp. 2987–2991.
[40] T.-S. Kim, H. Lim, J.C. Hou, Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks, in: Proceedings of ACM MobiCom, September 2006, pp. 366–377.
[41] H. Ma, H. Alazemi, S. Roy, A stochastic model for optimizing physical carrier sensing and spatial reuse in ad hoc networks, in: Proceedings of IEEE International Conference on Mobile Ad hoc and Sensor Systems, November 2005, pp. 615–622.
[42] K. Sanzgiri, I.D. Chakeres, E.M. Belding-Royer, Determining intra-flow contention along multihop paths in wireless networks, in: Proceedings of First International Conference on Broadband Networks, October 2004, pp. 611–620.
[43] E. Wong, R. Cruz, A spatio-temporal model for physical carrier sensing wireless ad hoc networks, in: Proceedings of IEEE Sensor and Ad Hoc Communications and Networks (SECON), September 2006, pp. 276–285.
[44] X. Yang, N. Vaidya, On physical carrier sensing in wireless ad hoc networks, in: Proceedings of IEEE INFOCOM, Miami, FL, March 2005, pp. 2525–2535.
[45] Y. Yang, J.C. Hou, L.-C. Kung, Modeling of physical carrier sense in multi-hop wireless networks and its use in joint power control and carrier sense adjustment, in: Proceedings of IEEE Infocom, May 2007, pp. 2331–2335.
[46] H. Zhai, Y. Fang, Physical carrier sensing and spatial reuse in multirate and multihop wireless ad hoc networks, in: Proceedings of IEEE Infocom, April 2006, pp. 1–12.
[47] J. Zhu, X. Guo, L.L. Yang, W.S. Conner, Leveraging spatial reuse in 802.11 mesh networks with enhanced physical carrier sensing, in: Proceedings of the IEEE International Conference on Communications, June 2004, pp. 4004–4011.
[48] R. Ramanathan, R. Rosales-Hain, Topology control of multihop wireless networks using transmit power adjustment, in: Proceedings of IEEE INFOCOM, 2000, pp. 404–413.

[49] C.-C. Shen, C. Srisathapornphat, R. Liu, Z. Huang, C. Jaikaeo, E.L. Lloyd, CLTC: a cluster-based topology control for ad hoc networks, IEEE Transactions on Mobile Computing 3 (1) (2004) 18–32.
[50] A. Muqattash, M. Krunz, A distributed transmission power control protocol for mobile ad hoc networks, IEEE Transactions on Mobile Computing 3 (2) (2004) 113–128.
[51] A. Muqattash, M. Krunz, POWMAC: a single-channel power-control protocol for throughput enhancement in wireless ad hoc networks, IEEE Journal on Selected Areas in Communications 23 (5) (2005) 1067–1084.
[52] B. Tavli, W. Heinzelman, Energy and spatial reuse efficient network-wide real-time data broadcasting in mobile ad hoc networks, IEEE Transactions on Mobile Computing 5 (10) (2006) 1297–1312.
[53] B.D. Lubachevsky, R.L. Graham, Curved hexagonal packings of equal disks in a circle, Discrete and Computational Geometry 18 (2) (1997) 179–194.
[54] G. Alonso, E. Kranakis, R. Wattenhofer, P. Widmayer, Probabilistic protocols for node discovery in ad hoc, single broadcast channel networks, in: Proceedings of IEEE International Parallel and Distributed Processing Symposium, April 2003, pp. 1–8.
[55] L. Galluccio, G. Morabito, S. Palazzo, Analytical evaluation of a tradeoff between energy efficiency and responsiveness of neighbor discovery in self-organizing ad hoc networks, IEEE Journal on Selected Areas in Communications 22 (7) (2004) 1167–1182.
[56] V. Giruka, M. Singhal, Hello protocols for ad hoc networks: overhead and accuracy tradeoffs, in: Proceedings of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), Taormina, Italy, June 2005, pp. 354–361.
[57] E.B. Hamida, G. Chelius, E. Fleury, Revisiting neighbor discovery with interferences consideration, in: Proceedings of ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks, Terromolinos, Spain, October 2006, pp. 74–81.
[58] M.J. McGlynn and S.A. Borbash, Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks, in: Proceedings of ACM MobiHoc, Long Beach, CA, 2001, pp. 137–145.
[59] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, Distributed topology construction of Bluetooth wireless personal area networks, IEEE Journal on Selected Areas in Communications 23 (3) (2005) 633–643.
[60] I. Cidon, O. Mokryn, Propagation and leader election in a multihop broadcast environment, in: Proceedings of 12th International Symposium on Distributed Computing, 1998, pp. 104–118.

Luke Ritchie received the BS and PhD degrees in electrical engineering from Arizona State University (ASU), Tempe, in 2003 and 2007, respectively. He is currently a network R&D engineer at DataSoft Corp., Scottsdale, AZ. His research interests are in the areas of interaction between medium access control (MAC) and routing in mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs). He is a student member of the IEEE.

Martin Reisslein is an Associate Professor in the Department of Electrical Engineering at Arizona State University (ASU), Tempe. He received the Dipl.-Ing. (FH) degree from the Fachhochschule Dieburg, Germany, in 1994, and the M.S.E. degree from the University of Pennsylvania, Philadelphia, in 1996. Both in electrical engineering. He received his Ph.D. in systems engineering from the University of Pennsylvania in 1998. During the academic year 1994-1995 he visited the University of Pennsylvania as a Fulbright scholar. From July 1998 through October 2000 he was a scientist with the German National Research Center for Information Technology (GMD FOKUS), Berlin and lecturer at the Technical University Berlin. From October 2000 through August 2005 he was an Assistant Professor at ASU. He served as editor-in-chief of the IEEE Communications Surveys and Tutorials from January 2003 through February 2007 and has served on the Technical Program Committees of IEEE Infocom, IEEE Globecom, and the IEEE International Symposium on Computer and Communications.
He has organized sessions at the IEEE Computer Communications Workshop (CCW). He maintains an extensive library of video traces for network performance evaluation, including frame size traces of MPEG-4 and H.264 encoded video, at http://trace.eas.asu.edu. His research interests are in the areas of Internet Quality of Service, video traffic characterization, wireless networking, optical networking, and engineering education.

Andrea W. Richa is an Associate Professor at the Department of Computer Science and Engineering at Arizona State University, Tempe, since August 2004. She joined this department as an Assistant Professor in August 1998. Prof. Richa received her M.S. and Ph.D. degrees from the School of Computer Science at Carnegie Mellon University, in 1995 and 1998, respectively. She also earned an M.S. degree in Computer Systems from the Graduate School in Engineering (COPPE), and a B.S. degree in Computer Science, both at the Federal University of Rio de Janeiro, Brazil, in 1992 and 1990, respectively. Prof. Richa's main area of research is in network algorithms. Some of the topics Dr. Richa has worked on include packet scheduling, distributed load balancing, packet routing, mobile network clustering and routing protocols, and distributed data tracking. Prof. Richa's data tracking (or name lookup) algorithm has been widely recognized as the first benchmark algorithm for the development of distributed databases in peer-to-peer networking, having being references by over 130 academic journal or conference publications to date, and being implemented as part of two of the current leading projects in peer-to-peer networking. Dr. Richa's was the recipient of an NSF CAREER Award in 1999. For a selected list of her publications, CV, and current research projects, please visit http://www.public.asu.edu/ aricha.