Connectivity Maintenance in Mobile Wireless Networks via Constrained Mobility

Joshua Reich, Vishal Misra, Dan Rubenstein, and Gil Zussman

***Abstract*—We explore distributed mechanisms for maintaining the physical layer connectivity of a mobile wireless network while still permitting significant area coverage. Moreover, we require that these mechanisms maintain connectivity despite the unpredictable wireless propagation behavior found in complex real-world environments. To this end, we propose the *Spreadable Connected Autonomic Network (SCAN)* algorithm, a fully distributed, on-line, low overhead mechanism for maintaining the connectivity of a mobile wireless network. SCAN leverages knowledge of the local (2-hop) network topology to enable each node to intelligently halt its own movement and thereby avoid network partitioning events. By relying on topology data instead of locality information and deterministic connectivity models, SCAN can be applied in a wide range of realistic operational environments. We believe it is for precisely this reason that, to our best knowledge, SCAN was the first such approach to be implemented in hardware. Here, we present results from our implementation of SCAN, finding that our mobile robotic testbed maintains full connectivity over 99% of the time. Moreover, SCAN achieves this in a complex indoor environment, while still allowing testbed nodes to cover a significant area.**

***Index Terms*—Adaptive systems, Cooperative systems, Mobile ad hoc networks, Mobile robots, Network topology, Wireless networks.**

I. Introduction

WE FOCUS on a fundamental problem facing mobile wireless networks: How can such a network maintain its own physical-layer connectivity as its constituent nodes move about? Our exploration of *connectivity maintenance* is prompted by such specific examples as the recent DARPA LANdroids initiative to develop a self-configuring network that can deploy itself for temporary use in highly complex wireless environments [1]. More generally, full network connectivity may be required for a network's overall mission (as above), useful for that mission (e.g., coordinated search and rescue, perimeter monitoring), or simply provide a way to prevent nodes from becoming lost.

Given the practical nature of our motivation, we focus on designing a protocol that can be implemented on a wide variety of hardware and work in wide range of real-world environments. In many such environments, wireless propagation itself may be quite unpredictable, failing to correlate well with intuitive quantities like distance - due to multipath propagation, interference from outside networks, interference between network nodes, and RF-absorbing environmental features. Even if all of these factors can be successfully incorporated into a (suitably conservative) predictive model, the presence of obstacles in the environment may prevent actual connectivity from reflecting the model's predictions. With currently available techniques, practical deployment of a connectivity maintenance scheme for many complex environments (e.g., indoor settings, urban/semi-urban areas, woodlands, battlefields), requires either explicitly mapping the deployment arena, or using an algorithm that does not rely on knowledge of the wireless propagation patterns.

Fig. 1. A testbed node.

Our work takes this latter approach, as pre-deployment mapping of complex environments is often costly and sometimes infeasible. Online mapping is a challenging problem in-and-of itself [2] (let alone when combined with a real-time connectivity maintenance constraint). To this end, we have developed the *Spreadable Connected Autonomic Network (SCAN)* protocol to handle complex environments in real-time without any prior knowledge of the environment.

Contrastingly, most previous work (e.g., [3], [4]) has relied on simple broadcast models (e.g., deterministic, spherical broadcast). By leveraging geometric properties, tractable algorithms for optimizing the movement of the nodes under a connectivity maintenance constraint can be created. Our work, which does not rely on such broadcast models, cannot produce such optimizations. The trade-off is that while these approaches may be appropriate for networks comprising unmanned aerial vehicles, or mobile nodes on a large plain, they

Manuscript received 31 July 2011; revised 6 May 2012. This work was supported in part by NSF grants CNS-0916263, CCF-0964497, and CNS-1018379, DTRA grant HDTRA1-09-1-0057, DHS Task Order #HSHQDC-10-J-00204, and a gift from Google. A preliminary version of this work appeared in Proc. IEEE INFOCOM 2011, Shanghai China, April 10-15, 2011.

J. Reich is with the Department of Computer Science, Princeton University, Princeton, New Jersey 08544 (e-mail: jreich@cs.princeton.edu).

V. Mishra and D. Rubenstein are with the Department of Computer Science, Columbia University, New York, New York 10027 (e-mail: {misra,danr}@cs.columbia.edu).

G. Zussman is with the Department of Electrical Engineering, Columbia University, New York, New York 10027 (e-mail: gil@ee.columbia.edu).

Digital Object Identifier 10.1109/JSAC.2012.120609.

may be unable to address many realistic scenarios of greater complexity - scenarios in which our technique can provide an *implementable working solution*. To demonstrate this claim, we have successfully implemented and tested SCAN on an 802.11-based robotic wireless networking testbed (Fig. 1). To our best knowledge, SCAN has been the first autonomous connectivity maintenance algorithm to have been deployed in hardware [5].

SCAN works by enabling individual nodes to determine when they must constrain their mobility in order to maintain connectivity. SCAN achieves this through an entirely distributed process in which individual nodes utilize only local knowledge (2-hop) of the network's topology to *freeze* their movement if SCAN's decision criterion indicates further movement risks network partition. Keeping with our focus on implementability, we have designed SCAN to work with commercially available off-the-shelf components. We have also designed SCAN to be agnostic to the particular movement goals of the nodes, allowing SCAN to accommodate differing mission goals. For example, a civilian self-deploying network might aim to maximize coverage area, while a military version may need to balance coverage with providing nodes the ability to move when threatened.

We assess SCAN with respect to two potential use scenarios:

- **Self-deploying mesh network**: In this application a SCAN spreads over some area to provide wireless coverage for previously disconnected, wireless (perhaps stationary) end nodes. Once it has covered these nodes, the SCAN can provision them with point-to-point connectivity. By ensuring that mobiles remain connected, a self-spreading mesh network infrastructure can provide wireless services in a more robust way than could a sporadically connected network. Moreover a system that maintains connectivity inherently prevents nodes from disconnecting - thereby preventing them from wandering off and becoming lost and unused. In this case, we evaluate performance with respect to the *coverage area* achieved: the total area covered by at least one mobile node that remains connected to the base station/uplink.
- **Search and rescue**: A wirelessly-communicating robotic search and rescue team deployed during an ongoing disaster must move to locate victims, but it must also remain connected to base-station points to immediately notify first responders when a victim has been found, in addition to providing ongoing feedback about the search environment. In this case, we evaluate performance with respect to *target detection*: the probability of detecting one or more targets located within a search area.

We conduct this assessment with analytic modeling, simulation, and implemented experiments. We use analytic modeling to explore SCAN's asymptotic properties. Through simulation, we verify these analytic results and extend them to scenarios more realistic than those captured by our modeling. Our hardware experiments examine the practical functioning of SCAN and its suitability for use in a real-world setting.

While SCAN is only a first step towards a comprehensive solution to an extremely complex and exciting problem, we believe it offers the correct jumping-off point for future research on practical methodologies for connectivity maintenance. Moreover, where previously discussed techniques are applicable, SCAN offers a robust fall-back mechanism. Additionally, SCAN addressess scenarios in which connectivity of very simple nodes is desired (e.g., micro-scale robots).

Our main contributions are:

- We propose SCAN as a baseline connectivity maintenance mechanism for challenging environments.
- We describe SCAN's intuition and properties under the most general (and challenged) settings. We also show how additional specialized information (e.g., RSSI) can be incorporated into SCAN's framework (Sec. IV).
- Through analysis and simulation, we evaluate SCAN's ability to maintain network connectivity while enabling significant area-coverage. We identify a phase-transition point at which SCAN networks transition between asymptotically frozen and always moving. We characterize that point as a function of node population and bounding region.
- We implement SCAN on our mobile robotic network testbed as proof-of-concept, and evaluate its performance for use as the connectivity maintenance facility for a self-deploying mesh network system (Sec. V).

II. Related Work

Connectivity maintenance for mobile networks has become an active area of research over the past several years. The control theory community began exploration of this area, in the context of motion planning algorithms. These motion planning algorithms attempted the maximization of some specific target function under the constraint that connectivity be maintained. Node movement patterns are determined completely by the specified controller(s). Early work focused on maximal coverage [6], and shortly thereafter, on continuously connected group movement [3]. A series of papers considers the use of potential fields to supply centralized [7], distributed [8], and centralized double integrator [4] schemes for ensuring connectivity while maximizing a metric encoded in those fields. More recently, [9] adds the consideration of collision avoidance, while [10] looks at the looser constraint that connectivity reoccur periodically. Most work in this area leverages either geometric properties or assumes perfect knowledge of the potential fields used for determining connectivity and utility. An exception, [11] uses only two-hop information to maintain connectivity, albeit by dividing the node population into backbone and regular nodes.

However, all of the above approaches make restrictive assumptions regarding node connectivity - assumptions which are unlikely to be true in practice. By far the most popular such assumption is that node broadcast ranges are perfectly spherical, deterministic, non-interfering, and not subject to attenuation from obstacles or other environmental conditions. Notably, [12] does relax these assumptions, considering a fairly realistic broadcast model, but only for simple chain topologies.

Given the restrictive assumptions made by this body of work, it is perhaps unsurprising that only two related hardware evaluations have been attempted, [13] and [14]. Each conducted their hardware evaluation in a single, empty, rectangular room. [13] maintains connectivity by utilizing both

onboard IR sensing and an overhead camera that continuously transmitted each node's current global position. [14] does not attempt to maintain connectivity physical layer connectivity at all. Instead their goal to ensure they avoid partitioning the network at the IP-level. In this approach, robot movement is controlled via joystick by human operators who ensure the physical layer of the wireless network does not partition. The robots themselves are only responsible for estimating their relative coordinates and choosing which neighbors with whom they will communicate so as maintain IP-level communication.

Recently, several papers have noted the practical shortcomings of approaches reliant on simple broadcast assumptions and have taken heuristic approaches utilizing actual connectivity/signal strength information. Consequently, these approaches have been much more amenable to at least limited hardware prototyping and evaluation. [15] uses previously generated radio signal strength maps and hand-produced free space cell decompositions as input to their connectivity algorithm. [16] addresses a different problem - extending a connected network by having human operators drop "breadcrumb" routers when connectivity begins to weaken. Another related problem is considered by [17], which proposes mechanisms that repair disconnected networks, leveraging graph properties similar to those used by SCAN. Finally, [18], develops a distributed algorithm that is a slight modification of the *Neighbor Density (ND)* algorithm we presented previously [5] and use for comparison here. When evaluating this algorithm on a testbed built along the lines published in [19], [18] found eight nodes needed 35 minutes to converge - several times longer than needed by SCAN as described in V-B.

Finally, it is worth noting that, like the body of work above, SCAN does not provide a facility for IP-level routing. We believe most MANET routing protocols should be able to work alongside the SCAN protocol. However, we note [20] has found that even such protocols may perform poorly with mobile robots. Consequently we recommend choosing a protocol that fits well with the SCAN approach, localizing control messages as much as possible to the vicinity of topological changes.

III. Problem Setting

A. Operational Environment

SCAN was designed to operate in unknown and complex environments using commercially available hardware. As a result, SCAN must contend with both unpredictable wireless broadcast and unknown obstacles to broadcast (and also potentially to nodal movement). With respect to the former, wireless broadcast in the 802.11 spectrum is unpredictable, subject to cross-talk, multipath propagation, fading, and interference. These effects are only exacerbated by unknown features of the operational environment. Multipath effects are engendered by walls and obstacles, while fading increases in the presence of RF-absorbent surfaces. In such an environment, knowing where two nodes are *positioned* with respect to one another is often a very different matter from knowing whether they will be wirelessly *connected*. Moreover, many environments (e.g., indoors, underground, battlefield) are GPS denied. While there are techniques addressing indoor localization [21], [22], such systems require significant time to setup and/or leverage expensive hardware.

Automated techniques to create maps that allow such inference to be performed with reasonable confidence are just now being developed [2]. It is unclear how these might be incorporated into an algorithm that maps while simultaneously maintaining connectivity. Moreover, such a map can quickly become obsolete should any contributing factor of the environment change sufficiently. Our guiding philosophy behind SCAN is that the most effective and practical way to determine whether nodes will be connected in the *future* is by extrapolating from their *current* connectivity. Sec. V-A discusses the features of the particular operational environment used for our experiments.

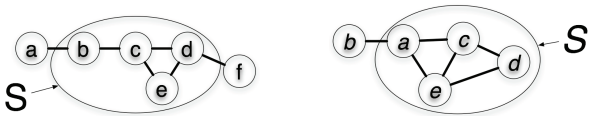
B. Robotic Testbed

The mobile robotic networking testbed used for our implementation and experiments is similar to recent systems such as [19], [23], and [24]. Each mobile node in our testbed is composed of an iRobot Roomba Create mobility and sensing platform (Fig. 1). On top of each Roomba we affix a Linksys WRTSL54GS wireless router running OpenWrt Linux. The Linksys router provides an integrated unit featuring a Broadcom 4704 processor running at 266MHz, 8MB flash, 32MB RAM, an integrated Broadcom wireless 802.11g radio, BCM5325 switch, and a USB 2.0 port. The WRTSL54GS provides communication, computation and memory, while the Roomba provides power, movement, and environmental sensing. The Roomba platform and the router are connected by a modified serial cable that allows the router to both poll the Roomba's sensors and control the Roomba's motors and actuators. The serial cable also provides a direct unregulated power feed to the Roomba's battery which we use to power the router. Typically a node can run without a recharge for several hours. This setup allows us to experiment with real mobile nodes whose broadcast and mobility decisions we can specify utilizing standardized programming languages.

Our nodes do not utilize GPS, although we are working to incorporate RSSI. Aside from brief discussion regarding these efforts presented in Sec. IV-E, the current work focuses on utilizing the presence/absence of connectivity and not the relative strength of that connectivity. This focus was driven by our desire to devise a general-purpose mechanism that can be used today. Currently, proprietary WiFi drivers hide signal strength information (in our case Broadcom drivers hid all but an instable RSSI reading of average channel strength which was essentially useless for estimating signal strength with any given peer).

C. Network Model

Assume our network contains N mobile robots, each with a (mean) transmission range r, and deployed within an area A. We represent the network as a graph in which each node corresponds to a mobile robot. Two nodes u and v are *neighbors* and are *directly connected* via a *link*, if they can directly, mutually, and consistently communicate over a wireless channel (if u can receive v's broadcast but not vice versa then u and v are not directly connected).



(a) a and b are neighbors. a and d are connected by path (a, b, c, d).

(b) A network with both robust and fragile connections.

Fig. 2. Network connectivity and robustness.

We define $N(u)$ as the set of nodes that are node u's neighbors with $u \notin N(u)$, and assume, as is the case in our testbed, that each node u knows $N(u)$, and is also informed of $N(v)$ for each of its neighbors $v \in N(u)$. u and v are *connected* if there is a *path* from u to v across a series of links. A network is globally *connected* when every two nodes u and v in the network are connected.

Within each fixed-length *broadcast cycle*, each node assesses its current local connectivity, and, based on this state decides to either *move* or *freeze* until the next broadcast cycle completes. We do not require that nodes make their decisions simultaneously, nor do their respective T's need to match exactly (clock drift is permissible).

Over time, new links can *form* and existing links can *fail*. However, the state of a link between two nodes u and v can change only if at least one of the nodes is moving. If both u and v are frozen, then an existing link between them cannot fail, nor can a link be added when there is none. In other words, only the mobility of node pairs significantly alters whether a pair can communicate.

D. Neighbor Detection

The first challenge in implementing `SCAN` on our testbed lies in determining when two Roombas should be considered "connected". Consider two Roombas, which we label u and v. If u and v can receive (a majority of) one another's transmissions, then clearly they should be connected, and if u and v cannot hear one another's transmissions, they should not be connected. But what about cases in between? In particular:

- u could hear v, but the reverse need not hold true.
- u might hear from v only intermittently, their connection being too sporadic to allow for communication that consumes any significant bandwidth.

To determine whether a given pair of nodes can directly, mutually, and consistently communicate over a wireless channel, we use the following neighbor detection protocol. Each node runs repeatedly broadcasts and listens over a fixed period of time we refer to as a *broadcast cycle*. We make no attempt to synchronize the start time of a cycle across nodes, or to eliminate drift across node clocks, as such tight synchronizations are not needed for our use of the broadcast cycle. The remainder of this subsection discusses the details of how we determine when two nodes are connected within a broadcast cycle, and how a node learns of both its neighbors and 2-hop neighbors (i.e., its neighbors' neighbors). This discussion can safely be skipped by the reader who is not concerned with these details.

In our testbed, a broadcast cycle lasts 1.5 seconds. In each cycle, a node u classifies a node v whose transmissions it hears during the previous or current cycle into one of three possible classes: *heard*, *reciprocally-heard*, and *confirmed* as a neighbor. v is *heard* by u if u receives transmissions from v. v is *reciprocally-heard* if in addition, u receives a broadcast from v indicating u is currently heard by v. Finally, v is *confirmed* as a neighbor if u has reciprocally-heard from v in two consecutive broadcast cycles. Once v is confirmed as a neighbor, v remains in u's set of neighbors until such time as u fails to reciprocally hear v for two consecutive broadcast cycles.

The messages sent in a broadcast cycle, a node broadcasts a sequence of 3 `heard` messages followed by a sequence of 3 `confirmed` messages. A `heard` message begins with the prefix "2" followed by a list of IP addresses nodes heard by the sender in its previous broadcast cycle. A `confirmed` message begins with the prefix "3" and is followed only by a list of IP addresses belonging to the sender's confirmed neighbors. Either `heard` or `confirmed` messages received can be used to classify a node as being reciprocally heard. However, only nodes whose addresses appear in a `confirmed` message can be used for `SCAN`'s calculations.

We found this scheme to be very effective in both locally communicating the information used by `SCAN` and providing neighbor sets that were fairly stable against random fluctuations on the wireless channel - without artificially restricting the set of neighbors detected.

E. Node Mobility Pattern

Aside from sometimes requiring the Roombas to freeze, we place no other explicit restrictions on their movement. As our primary interest lies in networking, rather than robotics, we have focused our main efforts on `SCAN`. To spread our nodes across the test area we use the simple movement algorithm shown in Fig. 3. Each robot moves straight ahead until it is stopped by an obstacle. Each robot has two sensors which allow it to determine whether the obstacle is off to the side or in front. If the obstacle is to a side, the robot rotates a random angle and continues. If the obstacle is in front, the robot backs up slightly, rotates a random angle, and proceeds. To further encourage network spreading, once every few seconds robots will randomly turn several degrees.

This algorithm ran as a separate thread on each of our nodes. At the end of each broadcast cycle each node would reassess the `SCAN` criterion. If its mobility state changed from freezing to moving, the main thread would notify the mobility thread to begin again. Conversely, if the mobility state changed from moving to frozen, then the thread would be paused and a freeze command sent to the Roomba platform.

The important point is that the default node movement pattern is *oblivious* to any network connectivity requirement. Consequently, we believe that `SCAN`'s success at maintaining connectivity while providing for reasonable area coverage under this movement pattern, will generalize well to many other movement patterns.

IV. Algorithms for Connectivity Maintenance

Our approach to constructing a connectivity maintenance algorithm utilizes a three-step structure. The first step is to

```
while move = True do
  if sensorInput = hitBothBumpers then
    ROOMBA ← back_up(0.5)
    ROOMBA ← turn(random(180))
  else if sensorInput = hitRightBumper then
    ROOMBA ← arc(left)
  else if sensorInput = hitLeftBumper then
    ROOMBA ← arc(right)
  else
    ROOMBA ← forward
    sleep_unless_input_changed(random(5))
    if inputChanged then
      continue
    end if
    ROOMBA ← turn(random(10))
  end if
end while
```

Fig. 3. Node mobility algorithm.

gather data on the current network topology. The second is to assess the robustness of that topology. The final step uses this assessment to then determine whether further movement endangers future network connectivity. If so, we require that the node(s) for whom this is true refrain from further movement by freezing until such time as continued movement no longer poses this risk.

To this end we introduce two algorithms leveraging this basic structure: a naive *Neighbor Density (ND)* algorithm which uses a very simple metric for assessing connectivity robustness, and SCAN which conducts a still simple, yet significantly more powerful assessment of robustness.

Both ND and SCAN share the common assumption that, while the quality of the wireless channel may fluctuate unpredictably over space, it will remain relatively constant over time: relative movement of two nodes may effect their connectivity, but time-wise fluctuations will not have a significant effect. In situations where the quality of the wireless channel is fluctuating wildly over time (e.g., significant and varying external interference, fast fading) more specialized or conservative techniques will be required.

A. Neighbor Density Algorithm

The *Neighbor-Density (ND)* algorithm (shown in Fig. 4(a)) serves as a naive parameterizable heuristic solution to the connectivity problem. ND utilizes nodal density (or more precisely valence) to achieve connectivity: if a node has more than k neighbors it considers local connectivity robust and consequently may move; if it has fewer, it must freeze[1]. ND's messages are constant in the number of neighbors, as only the sending node's ID need be sent.

B. Spreadable Connected Autonomic Network Algorithm

The *Spreadable Connected Autonomic Network (SCAN)* algorithm (shown in Fig. 4(b)) takes the greedy approach that a node's movement should only be constrained in direct response to a perceived lack of robustness in the (local) network connectivity structure. To get a feel for when we may wish to freeze a node, consider the example in Fig. 2(b).

Node a is connected to 3 neighbors. To disconnect a from any of the neighbors to its right (or in fact any of the nodes to which it is connected) at least 2 links in S must be broken. In contrast, only a single link needs to fail to disconnect node a from node b. When links fail infrequently (e.g., 1 failure during a broadcast cycle), then if a were only concerned about the nodes in the set S, it could continue to move. However, there is a high likelihood that any movement could cause the single link between a and b to fail, ending the connection with b, thereby partitioning the network. To keep the network partition-free, both a and b should freeze.

But what if, during a broadcast cycle, $k \geq 1$ links can be expected to fail? How do we then determine whether nodes must freeze to ensure network connectivity is maintained? It is this question that SCAN is designed to address.

Formally, SCAN is pre-configured with a parameter k, such that a node u is allowed to move as long as $|N(u) \cap N(v)| \geq k$ for every $v \in N(u)$. In other words, u moves if it shares k neighbors with each of its neighbors v. If this property does not hold for even a single neighbor, u must freeze.

C. Global Connectivity

Consequently, if a directly connected pair of nodes lack a sufficient number of redundant paths routed through mutual one-hop neighbors, SCAN concludes that movement on either of their part may in the worst case sever all one-hop routs between these two nodes. Even in such a scenario other longer routes may, in fact, remain, in which case these nodes might still remain connected. However, SCAN conservatively assumes that only the known routes can be relied upon in assessing connectivity robustness and SCAN only provides nodes with up-to-date local two-hop topologies. Another algorithm could, of course, utilize more topological information, but only at the cost of propagating a greater amount of information and ensuring this information is up-to-date. In fact, ND can be thought of as an algorithm in this family that uses only local one-hop topology information.

We now make rigorous the argument that by assessing connectivity robustness between each pair of nodes on a local basis, SCAN will likely prevent any pair of directly connected nodes from severing all of the locally known paths between them. Applied across all nodes this property prevents global network partition (albeit with lower probability than that of any individual node partitioning).

Lemma 1: In any graph, if u and v are directly connected and $|N(u) \cap N(v)| = m$, if fewer than m links fail, then there is a path of at most 2 hops from u to v.

Proof: Clearly, if u and v remain directly connected, then there is a 1-hop path from u to v. To remove 2-hop paths, a link must fail between each node in $N(u) \cap N(v)$ and either u or v. Hence, at least $m + 1$ links must fail to remove all 1 and 2-hop paths. ■

For the following Lemma, we remind the reader that, as stated in Section III-C, links cannot fail between a pair of frozen nodes.

[1]Both [16], [18] use slight variations on ND for maintaining connectivity.

if $|N(u)| \geq k$ **then** move **else** freeze

(a) ND

if $|N(u) \cap N(v)| \geq k,\ \forall v \in N(u)$ **then** move **else** freeze

(b) SCAN

Fig. 4. Mobility criteria.

Lemma 2: Consider a network that is connected at some time t, with all nodes using SCAN with parameter k. If at most k links can fail near a node during its broadcast cycle, then the network remains connected for the duration of the broadcast cycle.

Proof: We note that links may form during the period in which k links fail. If we ignore these forming links and show the network remains connected, then clearly the network remains connected when we add these forming links back in.

We proceed by contradiction. Let t be the time that the graph partitions, and let the partition result from the edge connecting u and v failing. This means that either u was moving or v was moving under SCAN. WLOG, assume u was moving at time t. This implies that $|N(u) \cap N(v)| \geq k$ at the start of u's broadcast cycle containing time t. Since at most k links can fail in a broadcast cycle, at most k links can fail between the start of this broadcast cycle and time t. By Lemma 1, u and v must be connected (within at least 2 hops) even after k links fail, and hence cannot reside in separate partitions at time t. ■

Lemma 2 states that, by using SCAN, even if k links suddenly drop simultaneously (and are connected to at least one moving node), the way that SCAN chooses nodes to move and freeze ensures that these k dropped links, each of which must drop between two nodes where at least one is moving, will not disconnect the network.

Theorem 1: If a node reconsiders its decision to move or freeze at the end of each of its broadcast cycles, and at most k links can fail within its broadcast cycle, then a SCAN that is initially connected will always remain connected.

Proof: The proof is by induction on the iteration of the broadcast cycle. Our assumption is that for the initial broadcast cycle, the network is connected. Using the inductive assumption, assume the network remains connected by the end of the ith broadcast cycle of node u. u reassesses its local connectivity at the end of the cycle, and moves in the $i+1$st broadcast cycle only if $|N(u) \cap N(v)| \geq k$ for all $v \in N(u)$. We simply apply Lemma 2 with t equal to the start time of the $i+1$st broadcast cycle to complete the inductive step. ■

D. Choosing SCAN's k Parameter

Since SCAN disconnections only occur when at least $k+1$ nearby links fail simultaneously, the best value of k depends upon how many neighboring links are expected to fail (i.e., nodes move out of communication range) within a broadcast cycle. As one increases the speed of a node, decreases the range of transmission, or increases the broadcast cycle time, a larger k is needed to ensure connectivity. In general, as k increases, it becomes less likely that the network will partition, but the expected time that nodes spend moving is reduced as well, which can delay, or even prevent, achievement of the goal for which mobility of nodes is required in the first place. In Section VI we find that for network whose nodes move slowly $k = 2$ is more than sufficient while for more volatile settings $k > 4$ appears extremely robust.

Pre-determining the optimal k for an arbitrary setting is quite difficult because: (i) link failures will often exhibit co-dependence, (ii) disconnection of a nearby nodes are dependent, (iii) $k+1$ nearby link failures permit *but do not necessitate* severing of all locally known paths between two neighbors, and (iv) even if all locally known paths connecting a node to some neighbor are severed, there may still be a global path connecting them.

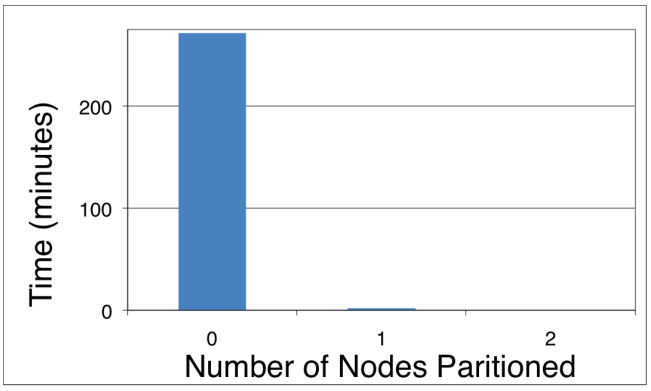
Consequently, we will use a back-of-the-envelope calculation to give some insight into how k should be set. We begin by setting the probability p of a link breaking during a given SCAN broadcast cycle to the distance a node travels in a period divided by the mean broadcast radius. Roughly speaking, $k+1$ or more links break with probability order p^{k+1}. Assuming a disconnection lasts for a mean time of τ broadcast cycles, the expected fraction of time the network is fully connected is $1 - \tau p^{k+1}$. k should be chosen such that τp^{k+1} is less than the tolerated level of partitioning. The interested reader may observe our algorithm in action and test the effect of changing k using a simplified applet version of our simulation environment [25].

E. Incorporating Additional Information

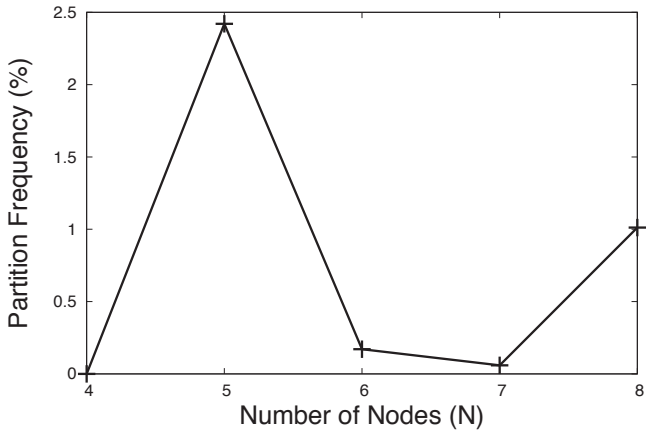
While we do not address utilizing additional sources of information such as GPS or RSSI in this paper, we do wish to briefly describe how they might be incorporated in SCAN's general approach. SCAN simply views each connection as a binary value, 1 if connected, 0 if not, corresponding to the presence or absence of an edge in our network. A version of SCAN utilizing additional information (SCAN+) could use weighted edges, whose weights correspond to normalized RSSI values or relative distance measurements. One possibility would be to require that the weighted sum of the of the paths connecting any neighbor and a given node be greater than a constant k' allowing the presence of strong connections to offset lower absolute numbers of common neighbors. Clearly more sophisticated schemes might be devised leverage an increasingly nuanced view of the connectivity topology for improved performance.

V. Testbed Experiments

Our goal was to develop connectivity maintenance techniques that could actually be implemented and tested on hardware in a noisy, challenging environment. To this end, we have implemented SCAN on our Roomba robotic testbed. Over the course of weeks, we ran trials on our testbed, collecting several hours worth of experimental data. Recall from Sec. III-E that our nodes operated in a GPS-denied environment and explored the area randomly. It seems reasonable to expect that if our blindly moving nodes could stumble into a successful configuration, nodes with more robust mobility routines tailored for a particular application should do at least as well.



(a) Run time vs. size of partition.



(b) % Network partitions vs. # nodes.

Fig. 5. SCAN partitioning statistics.

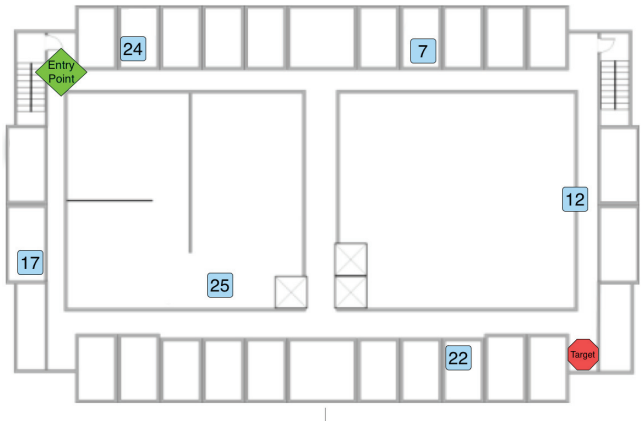


Fig. 6. Indoor experimental setting.

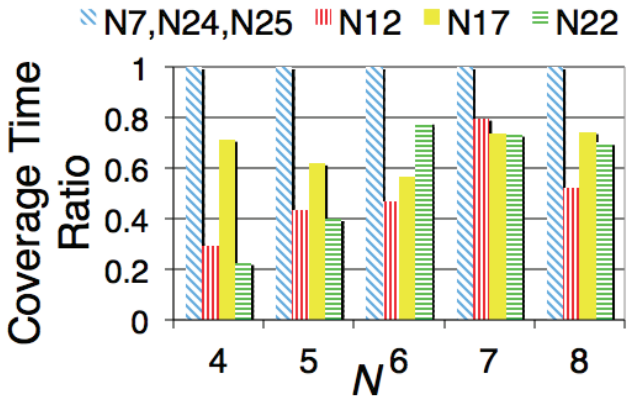


Fig. 7. Fraction of time clients covered.

We found that SCAN could provide connectivity in a remarkably robust fashion while also providing latitude of movement sufficient to cover clients scattered throughout our test environment. Out of 273 minutes and 50 seconds of experiments, our network remained connected in all but 2 minutes and 23 seconds. Moreover, the network partitions we encountered comprised a single node disconnecting from the rest of the network. The sole exception was one 15-second period during which a pair of nodes partitioned themselves as a connected component. This result is shown graphically in 5(a) which compares the total time (y-axis) during which zero, one, or two nodes partitioned from the main networks (x-axis). Fig. 5(b) shows the partition frequency (y-axis) broken down by the number of nodes in experiments (x-axis).

A. Experimental Setup

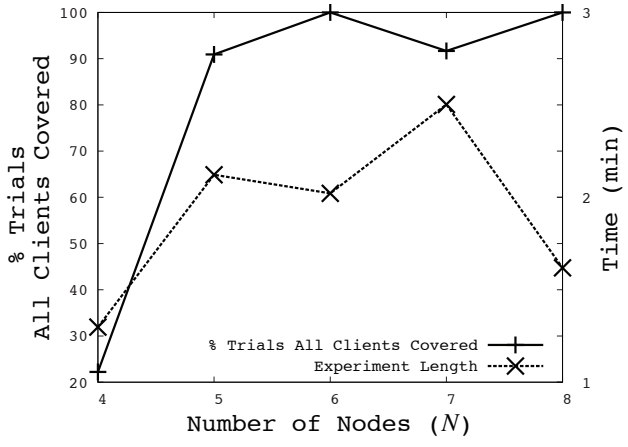
Our experiments were run on the 8th floor of the CEPSR research building, covering approximately 1900 m^2. A dozen or so wireless networks were competing for use on this particular floor, providing a moderate level of interference.

As previously discussed, we assess SCAN's performance on our primary metric: the *coverage area* SCAN allows while maintaining connectivity with bounded (in this case 99%) probability. Since measuring the total coverage area of our combined nodes with instrumentation was not feasible in our test environment, we instead placed wireless clients around the floor and measured the total number of clients covered as Fig. 6 illustrates. Secondly, we examine SCAN's ability to support a *target detection* task by measuring the percentage of trials in which at least one of the nodes would reach the target area before either the network froze or the 20 minutes elapsed (this timeout was reached only once).

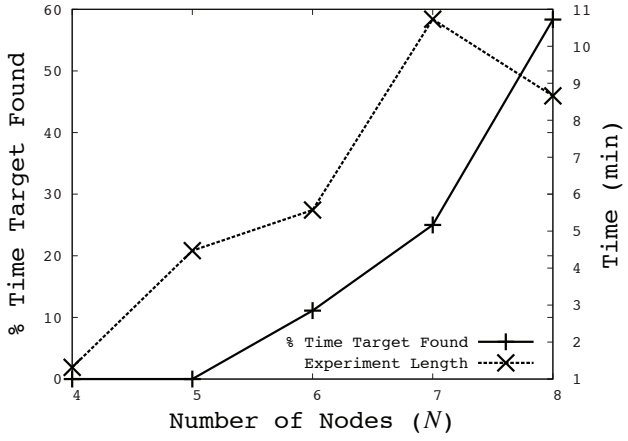
While the experimental space was moderately large and subject to both wireless interference from competing networks, as well as broadcast obstacles, our nodes could still broadcast a good proportion of its length. To conserve power and evaluate our algorithm in a more transmission-limited environment, we dialed down the broadcast power to the minimum level supported in software ($0.25dBm$) and did not restrict the shielding of client nodes (e.g., if they were behind doors, in far corners, or on the ground).

Our experiments tested networks of size $N = \{4, 5, 6, 7, 8\}$ using $k = 2$. Each data point averages 10 trials.

To choose this value of k we followed the logic laid out in Sec. IV-D. Given Roomba speed ($0.5m/s$), SCAN assessment cycle every $3s$, mean broadcast radius $20m$, and mean number of cycles for disconnection $\tau = 100$: we have $p = 3(s) * 0.5(m/s)/40(m) = 0.088$ and $\tau p^{k+1} = 100 * 0.088^3$ which would imply a tolerable partition likelihood of around 5% for $k = 2$. In fact, we found partition frequencies noticeably lower than this in our experiments.



(a) % trials coverage achieved/time taken.



(b) % time target found/time taken.

Fig. 8. Trial performance by network size

B. Experimental Results

In our tests SCAN maintained full network connectivity over 99% of the time. As can be seen from Fig. 8(a), this high degree of connectivity maintenance did come at the cost of constraining area coverage. In this figure, the left y-axis measures the percentage of trials in which coverage was achieved, the right y-axis the time in minutes, and the x-axis the number of nodes. Networks of four nodes were unable to ever fully cover all client simultaneously. Yet nodes were still able to move far enough that every client was covered for at least some significant proportion of the experiment as seen in Fig. V-B which plots the percentage of the time a given node was covered (y-axis) against the value of k used (x-axis). We can also see that the client nodes were quite heterogeneous with respect to coverage: certain clients were always covered, while others were often quite difficult to cover. Intuitively, this makes sense given the area's complexity.

For $N > 5$, we see a significant performance improvement. In this space a network of six nodes appears sufficient to provide simultaneous coverage to all nodes, although it takes over 2 minutes to do so. We see a continuing decrease in the time taken until all nodes are covered as N increases, along with an increase in success rate. The decrease in performance for $N = 7$ is most likely due to high variance resultant from the statistically smaller number of trials run. Thus, SCAN coupled with the most rudimentary of mobility mechanisms enabled a small number of mobile nodes to self-organize a configuration capable of covering to all clients.

Target-detection proved significantly more challenging. In part this was due to the difficulty an essentially randomly moving node would have in reaching a specific location in a circuitous environment with many small obstacles (e.g., waste baskets). But strikingly in only one out of 50 trials did the nodes run out of time before they either all froze or the target was reached by at least one node. The main constraint appeared to be that smaller networks simply lacked the number of nodes needed to maintain robust network connectivity as they continued spreading out towards the target area (for $N = 4$ even one link breaking was enough to freeze the network). Fig. 8(b) plots the percentage of trials (left y-axis) in which the target was reached, and the time in minutes for the experiment (right y-axis) against the number of nodes (x-axis). Unsurprisingly, the larger the number of nodes the more likely the target was to be detected. The experiment run time itself was dominated by the tendency of smaller networks to completely freeze (as explained above), while larger sets of nodes could continue moving for substantially longer - thereby increasing the likelihood of some node reaching the target. Only for the eight node experiment does the trial time decrease, as here the overall mobility of the network is so high that some node will quickly find the target. We suspect that for this environment, eight nodes is close to the network phase transition point discussed in Sec. VI-B.

C. Additional Results

We also conducted a preliminary assessment of our system on several outdoor areas which revealed two very interesting things. The first was that even using our basic hardware setup with broadcast power set to the minimum allowable, our testbed was still able to spread over a significant area as can be seen in Fig. 9. The resultant frozen configuration for the six nodes spanned an area over $100m$ north-south and $70m$ east-west. The second was the degree to which the environment effected wireless communication range. Only $300m$ north on an adjacent plaza we ran the same experiment, but the average distance between nodes was less than 33% that of the location we used for our experiments. We believe these differences are in large part due to the differing levels of radio frequency interference in these two locations (a weekend test showed greater distance spread between nodes at the northern location). Video of our experiment is available at [25].

VI. Simulation and Analysis

To further develop our understanding of SCAN and its scalability, we turn to simulation and analysis. In this section we seek to answer three essential questions regarding the behavior of networks running SCAN:

- Under what conditions will all nodes in a network running SCAN ultimately freeze?
- How well does a network running SCAN, when frozen, cover an area as a function of the probability of retaining connectivity?
- What is a good value of k to use?

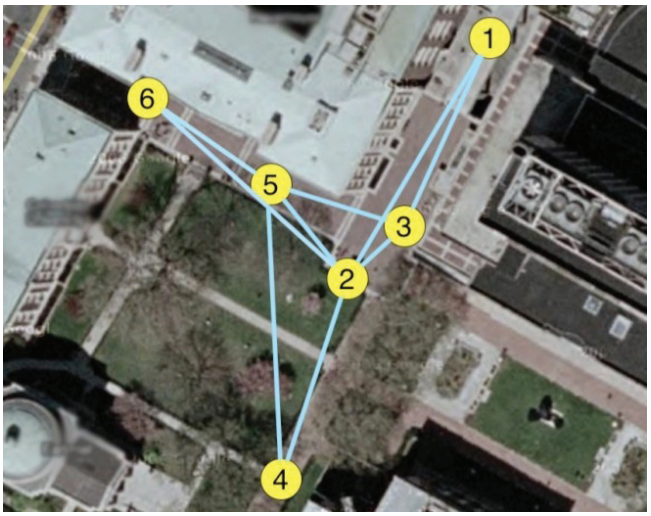


Fig. 9. Aerial view of frozen configuration, 6 node testbed.

While the latter two questions have fairly straightforward motivation, the first question bears additional discussion. This question is a significant one since if our nodes are confined to a very small space relative to their broadcast capabilities, then the local network topology around at least one node will always remain sufficiently connected to allow for continued movement. In some scenarios this could prove highly inconvenient (one would not want a civilian self-deploying network moving continually underfoot), while in others it might prove helpful (the same behavior in a military setting might help nodes avoid being targeted by the enemy). Either way, this phenomenon is worth understanding.

We investigate asymptotic freezing via analysis, which we then confirm by simulation in an obstacle-free environment. Further, through simulation we find that SCAN outperforms ND by a significant but not overwhelming margin in obstacle-free environments. However, when we introduce walls and use a realistic physical-layer wireless model in our simulation (an environment matching SCAN's design concerns), we see that SCAN continues to perform well for the same range of k values used in the obstacle-free environment, while ND's performance decreases drastically.

A. Simulation Platform and Assumptions

All the simulations discussed in this section were conducted on NetLogo 4.0.2 [26], a combined Logo-like language and simulation platform. Netlogo is ideal for modeling a distributed protocol whose behavior influences and is influenced by the topology of the dynamically evolving network upon which it is running. Unless otherwise noted, all plot points average 100 trials.

1) Obstacle-Free Environment: In our obstacle-free simulation, nodes within mean broadcast range r of one another were considered connected. To provide additional realism, we varied the range of broadcast stochastically. Unless otherwise mentioned, in this simulation two neighbors are connected if the distance between them is less than or equal to a normal random variable with mean $r = 1$ and $\sigma = 5\%$. Each potential pair of neighbors had its own independently chosen random variable, and these random variables were regenerated at a rate equal to the frequency of the movement decision made by the nodes We ran these experiments for $N = \{25, 50, 100, 200\}$.

2) Indoor Environment: Our indoor simulation mode deals with an environment partially subdivided by walls of differing thickness. For modeling the signal propagation in that environment, we used the COST 231 Multi-Wall Model (MWM) [27, Ch. 4]. Among empirical models, this is one of the most sophisticated ones and it is applicable in the $2.45GHz$ band. The MWM model predicts path loss as being equal to free space loss plus losses introduced by the walls and floors penetrated by the direct path between the transmitter and the receiver. Since we consider a single floor, the loss (in dB) is given by:

$$L = L_{fs} + L_c + k_{w1}L_{w1} + k_{w2}L_{w2} \tag{1}$$

where L_c is constant loss (we assume $L_c = 0$), k_{w1} is the number of light walls, $L_{w1} = 3.4dB$ is the loss due to a light wall, k_{w2} is the number of heavy walls, $L_{w1} = 6.9dB$ is the loss due to a heavy wall, and L_{fs} is the free space loss given by

$$L_{fs} = 32.4 + 20\log(r/1000) + 20\log(f) \tag{2}$$

where r is the distance between the transmitter and receiver (in meters) and f is the frequency in MHz (2450 in our case).

As much as possible, we aimed for our simulation to parallel our indoor experiments, using the same set of barriers shown in Fig. 6 for our indoor simulations. We set the transmit power of a node to $0dBm$ $(1mW)$. We assume that the receiver sensitivity is $-82dBm$, which is a reasonable value for ICE 802.11g devices and we assume a fast fading margin of $16dB$. Hence, we require that the propagation loss will satisfy $0 - (-82) - L > 16$ (or simply $L < 66$) in order for two nodes to be within transmission range. In an environment without walls this translates to allowed distance of approximately $17m$. In a multi-wall environment the distance varies with the locations of the different nodes.

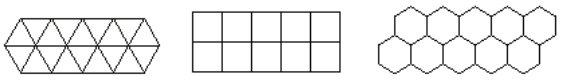
We note that the simulation model ignores collisions between SCAN messages simultaneously sent by other nodes. In our actual implementation, messages were sent on the order of seconds making such collisions very unlikely.

Given the higher computational complexity of these experiments, our simulations used a smaller number of nodes $(N = \{8, 16, 32, 64\})$.

3) Link Failure: In either simulation environment, the main factor in whether two nodes are connected arises from nodes' mobility. Our analysis does not consider stochasticity, relative movement being the only factor in determining connectivity. This assumption is made only to ease the analysis and is not required for SCAN to function correctly, as our testbed experiments in Sec. V demonstrate.

B. The Freeze Phase-Transition

If our nodes are confined to a very small space relative to their transmission radii, then they will never freeze. As the size of the space is increased, and nodes are able to spread further apart, the likelihood of all nodes eventually freezing increases. As the size of the space grows to ∞, we eventually will reach a point where, with probability 1, all nodes will freeze at some point in time. Once frozen,



(a) Triangles (b) Squares (c) Hexagons

Fig. 10. The only regular Tessellations.

nodes cannot obtain new neighbors, and thus network remains permanently in a frozen configuration. Here, we investigate this phase-transition. Given k and N, what is the *ratio* of the size of the space to the node transmission radius, beneath which the network is forever moving, and above which the network always reaches a freezing configuration? We will first build a model to predict this point and then verify our model's accuracy via simulation.

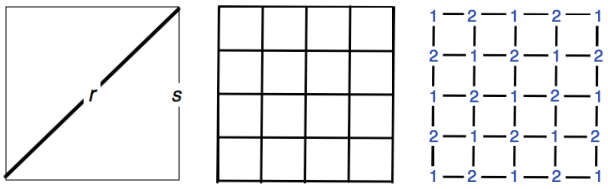
1) Minimum Bounding Area: To determine the inflection point, we begin by considering for a fixed broadcast radius and number of nodes how tightly those nodes could possibly be packed and still freeze. By closely bounding how tightly these nodes might be packed, we can then reverse this relationship and come to an approximation of how many nodes might be packed in a given area and still freeze.

Our model is composed of two components: a regular spatial pattern in which nodes can be laid out in a frozen configuration and the minimum scaling of this spatial pattern below which additional links will form. If we choose our model appropriately, the minimum area needed by this model will approximate the minimum area needed for such a system to freeze. Our main task is to identify a regular spatial pattern that is more dense than almost any frozen configuration we can expect to encounter and then to deliver a closed form equation for that pattern's size as a function of k, N, and r.

We begin by considering the simplest case $k = 1$ and a simple topology, a perfectly square space of area A. In order for a network to freeze, we must find some spatial configuration of the nodes such that (a) no node has any neighbors in common with any other neighbor, (b) the nodes are configured as densely as possible, and (c) the configuration is regular enough to analyze easily. The final criterion leads us to explore *regular tessellations*. A *tessellation* is created when a shape is repeated over and over again covering a plane without any gaps or overlaps. A *regular tessellation* is simply a tessellation composed of *regular polygons* - polygons for which all sides are the same length s. As it turns out, our search is relatively simple since there are only three regular polygons which tessellate in the euclidean plane: the triangle, square, and hexagon [28] as shown in Fig. 10.

Since we consider $k = 1$, triangles cannot be used, as any neighbor v of a given node u is also neighbors with the third node w on any triangle built upon edge u, v. Of the two remaining options, the hexagon allows for a tighter packing. However, it is moderately more difficult to work with than the square, upon which, as we will see, a very reasonable approximation of the phase-transition can be built.

Examining Fig. 11(a) we see that the maximum size for s in a square, the length of whose diagonal is $> r$ is $r/\sqrt{2}$ since $r^2 = 2s^2$. We can then place one node on each of the grid points on a $\lceil\sqrt{N}\rceil X \lceil\sqrt{N}\rceil$ grid as seen in Fig. 11(b). Such a grid will take up an area of $A = (r^2/2)(\lceil\sqrt{N} - 1\rceil)^2$ and the



(a) Square w/ radius r (b) 25 node 5x5 grid (c) 37 node 5x5 grid

Fig. 11. Geometry of our bounding area model.

ratio of one node's broadcast to the N-node bounding area is

$$\pi r^2/A = 2\pi(\lceil\sqrt{N}\rceil - 1)^2 \quad (3)$$

Extending this model to cover larger k is not overly difficult. For $k = 2$ instead of placing a single node at each grid intersection, we place two nodes at every other intersection as seen in Fig. 11(c). In this way, each node shares precisely one node with any of its neighbors, whether it is alone on its grid point or sharing it. Then, for a given N we only need $\lceil\sqrt{2N/3}\rceil - 1$ grid lines on each side, requiring an area of $(r^2/2)(\lceil\sqrt{2N/3}\rceil - 1)^2$. $k = 3$ is even easier requiring us to put two nodes at each grid intersection. We can extend this strategy to arbitrary $k \geq 1$ obtaining:

$$\pi r^2/A = 2\pi/(\lceil\sqrt{2N/(k+1)}\rceil - 1)^2 \quad (4)$$

which describes the ratio of an individual node's broadcast area to the total area. As will now be seen, our model's prediction tightly bounds the behavior seen in simulation.

2) Ratio of Broadcast Area to Bounding Area in Simulation: Our simulation results in this section were obtained through a binary search for the largest ratio of individual node broadcast area to bounding area that would result in a frozen configuration. The precise phase-transition point cannot be determined via simulation as 'failure to ever freeze' is an asymptotic property. Instead, we estimate this value by measuring the average convergence times from our experiments in the unbounded space and allowed our system to run in excess of 10 times the maximum convergence times taken there.

Each combination of N and k received 10 trials, each over the course of up to 5000 time-steps. To obtain a clearer correspondence with our model, in this trial alone, we did not stochastically vary the connection lengths. At the end of each trial that did not result in a freezing configuration, the ratio was decreased by half its current value for the subsequent trial. Conversely, whenever a trial ended in a freezing configuration the ratio would be increased by half. The first trial began with a ratio set slightly above the point where freezing might be expected to occur (determined by a set of preliminary experiments).

The results of our exploration for $k = \{1, 4\}$ are shown in Fig. VI-B2, as are the model predictions from (4) (other k values bound similarly but were omitted for graphical clarity). In this figure the y-axis measures the ratio of an individual node's broadcast area to the total bounding area while the x-axis measures the number of nodes N, each plot point representing the highest such ratio found at which a network of N froze.

The behavior of the phase-transition point for all k can be characterized roughly as "for every doubling in the number

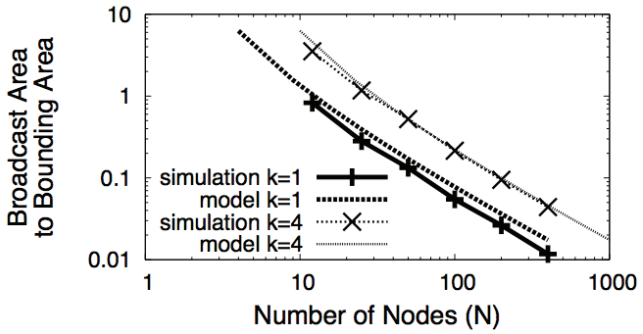


Fig. 12. Freezing phase transition.

of nodes, the ratio between node broadcast area and bounding area decreases by slightly less than half". Moreover, increasing values of k appear to lie at a relatively constant log-scale distance above one another. This is unsurprising as at higher levels of k a given number of nodes with a given broadcast area can fit into a smaller bounding space while still being able to reach a freezing state.

C. Performance in Obstacle-Free Environments

Here, we explore properties of the frozen configuration when SCAN and ND are applied in an unbounded space, where the configuration is guaranteed to eventually freeze. In our simulation, nodes are deployed at a gateway from which they proceed to spread across a 2-dimensional plane. As these nodes spread across the plane, they extend the area which the network attached to the station covers. However, if nodes becomes cut-off from the gateway through a network partition event, the entire area over which only these nodes broadcast ceases to be covered. Here we measure *coverage area* as the union of the area over which nodes currently connected to the gateway can broadcast. Consequently, coverage implicitly takes into account global connectivity, insofar as that connectivity benefits the coverage goal of a self-deploying wireless network.

1) Bounding the Maximum Coverage for $k = 1$: We begin by computing an analytic upper bound on the coverage-ratio. For the sake of tractability, we will assume that the transmission range of all nodes have identical broadcast discs bounded by a fixed radius r [2], and validate the results with simulation where we allow broadcast distance to vary stochastically.

We bound the maximum area a network running SCAN can cover for $k = 1$ (and consequently for all $k \geq 1$, although the bound becomes progressively less tight as k increases). We first note it is possible to bound this area trivially as $N\pi r^2$ where N is the number of nodes in the network. However, we can give a much tighter bound for $k = 1$ by examining the minimally overlapping line topology shown in Fig. 13. To calculate this area we first must determine the overlap between two nodes at distance r from one another.

Consider some neighbor v of u. v will be positioned at some distance s from u. This distance determines the shaded area of

[2]Although it has been recently shown that in some cases other models are required in order to capture issues such as collision and wireless interference [29], [30], the model still provides a reasonable abstraction. Extending the results to general SINR-based constraints is a subject for further research.

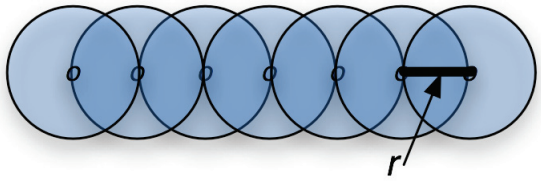
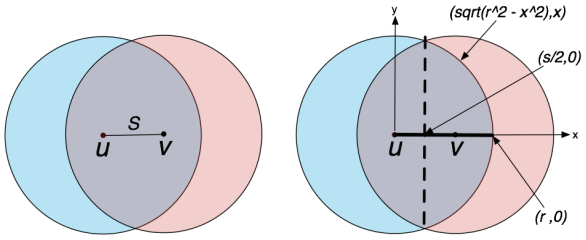


Fig. 13. The connected topology covering maximal area for $k = 1$.



(a) v at distance d from u (b) calculating overlap

Fig. 14. Overlap calculations.

overlap as seen below in Figs. 14(a) and 14(b). We calculate this shaded area S by noting that in general it is four times the size of segment circumscribed by u's perimeter and the dashed line and the x-axis. Since both circles have identical radii r and are centered at u and v respectively, by symmetry the dashed line lies halfway between them. This implies that $s/2 \leq x \leq r$ and $0 \leq y \leq \sqrt{r^2 - x^2}$ in the area of interest.

$$\begin{aligned} S &= 4\int_{s/2}^{r}\int_{0}^{\sqrt{r^2-x^2}} dydx \\ &= 4\int_{s/2}^{r}\sqrt{r^2-x^2}dx \\ &= 4\left[\frac{x}{2}\sqrt{r^2-x^2}+\frac{r^2}{2}\sin^{-1}\left(\frac{x}{r}\right)\right]_{s/2}^{r} \end{aligned}$$

Since in this case $s = r$ we have:

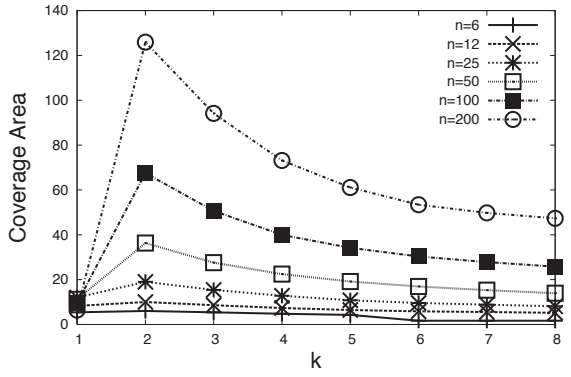
$$S = \frac{2\pi}{3}r^2 - \frac{\sqrt{3}}{2}r^2$$

Finally, we calculate the total area covered in a straight line configuration is

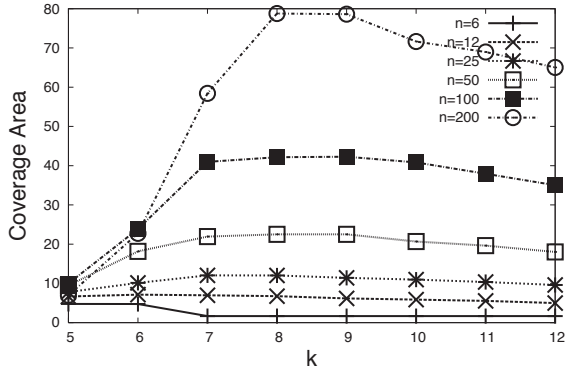
$$A = \pi r^2 + (N-1)(\pi r^2 - S) = \left[\pi + (N-1)\left(\frac{\pi}{3}+\frac{\sqrt{3}}{2}\right)\right]r^2 \quad (5)$$

□

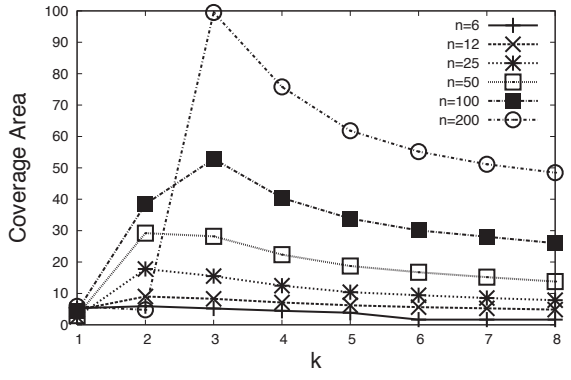
2) Coverage in Simulation for $k \geq 1$: Figs. 15(a) and 15(c) plot the size of SCAN's coverage area (y-axis) as a function of k (x-axis) for $\sigma = 0.05, 0.2$. The respective curves depict differing numbers of nodes in the network, with each node's communication range averaging a unit distance. The coverage area increases in proportion to the size of N, and is a decreasing convex function with the size of k, where nodes are required to maintain larger collections of neighbor sets. Additionally, for $k = 1$ the benefit of increased mobility in providing greater coverage is more than offset by the decrease in connectivity. For $k > 2$, the frequency of any network



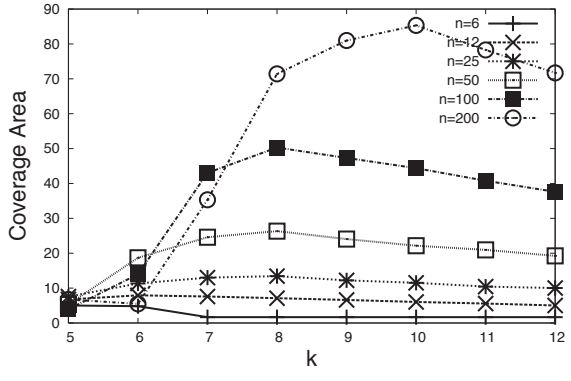
(a) SCAN, $\sigma = 0.05$



(b) ND, $\sigma = 0.05$



(c) SCAN, $\sigma = 0.2$



(d) ND, $\sigma = 0.2$

Fig. 15. Normalized coverage area (NCA).

partition does decrease but proves increasingly costly from a coverage standpoint.

Figs. 15(b) and 15(c) provide comparable plots for ND. The same trends discussed for SCAN are apparent, although optimally parameterized SCAN covers in excess of 50% greater area than optimally parameterized ND.

Here, we can see that increasing link variability from $\sigma = 0.05$ to $\sigma = 0.2$ shifts optimal parameterization of both SCAN and ND by one. This seems sensible, as with less reliable links, the best parameterization will need to be a bit more conservative in estimating when movement can be safely made without contributing to a network partition event.

3) Frequency of Network Partition: Figs. 16(a) and 16(c) plot the fraction of time that the SCAN graph is disconnected (y-axis) as k is again varied on the x-axis with the respective curves plotting differing values of N. Here, we see that the rate of disconnection is relatively unaffected by the size of N, but drops dramatically as a function of k. Note that even for values of $k = 2$ and $k = 3$, a significant number of disconnections occur as connections become less stable. This is due to the variability in the size of the transmission radius: nodes will form neighbor relationships when the radius is large, and then suddenly lose them, even when frozen, when the radius is small. By increasing k, nodes have a sufficiently large set of neighbors to offset the stochastic disconnections.

Figs. 16(b) and 16(d) show the same plots and trends for ND. In striking comparison to SCAN, ND's convergence towards a zero-partition frequency is both much more gradual, and incomplete. ND has a non-zero partition rate, even for $k = 12$.

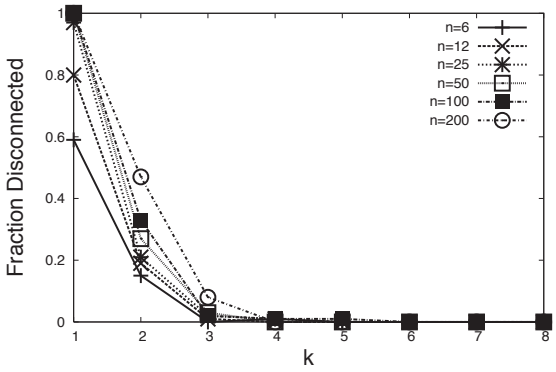
Increasing the link variability σ has relatively little impact, primarily affecting SCAN at $k = 2$ and having a slightly more noticable, but still minor, impact on the stability of ND.

4) Movement of Nodes: We now examine the impact on SCAN varying values of k on nodal movement for $N = 200$ considering first the more stable broadcast scenario.

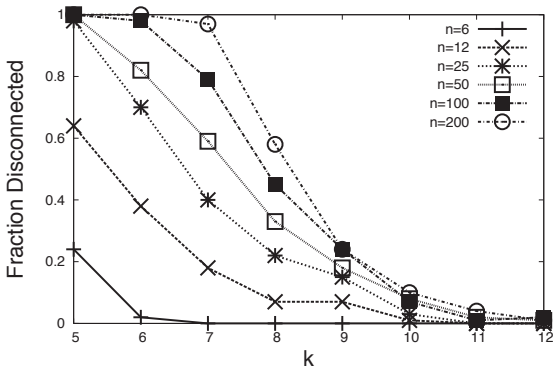
Fig. 17(a) shows the fraction of nodes moving (y-axis) plotted against time (x-axis). As is expected, eventually all networks freeze. There are two interesting trends. The first is a significant steepening of the curves as k increases. As k grows larger, not only does the network freeze more quickly, but the vast majority of the nodes halt their movement within a relatively short time window. The second interesting trend is that all distributions have relatively thin tails. There tends to be a long period before the network freezes during which only very few of the network's nodes are moving. This trend is most pronounced for small k.

When we consider the PDF (y-axis) of the fraction of nodes moving (x-axis) in Fig. 17(b) we can see the above trends clearly. For $k \geq 2$ the PDF spikes for very large numbers of nodes and very small numbers of nodes. Most nodes stop during a relatively short period of time, but the last few nodes take a significantly longer time to halt. For $k = 1$ the PDF actually peaks significantly before this point. We do not currently have an explanation for this behavior.

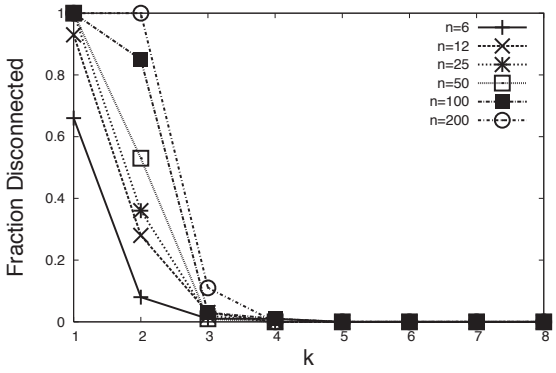
5) Target Detection: To assess the performance of SCAN for a target detection mission, we ran a simulation experiment in which nodes detected target when they were within $\frac{1}{10}^{th}$ of their mean broadcast radius. To gain an understanding of the interplay between connectivity and exploration targets were



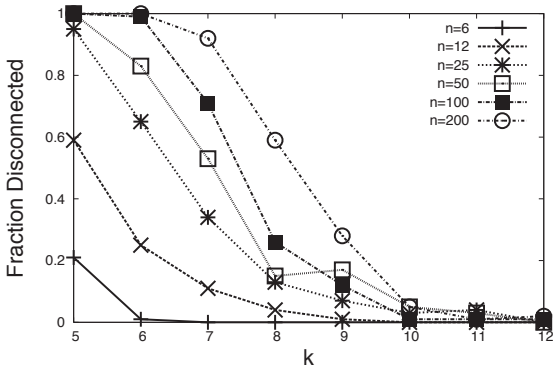
(a) SCAN, $\sigma = 0.05$



(b) ND, $\sigma = 0.05$

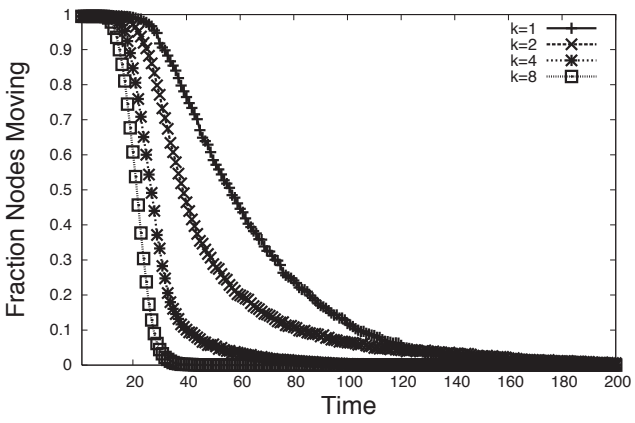


(c) SCAN, $\sigma = 0.2$

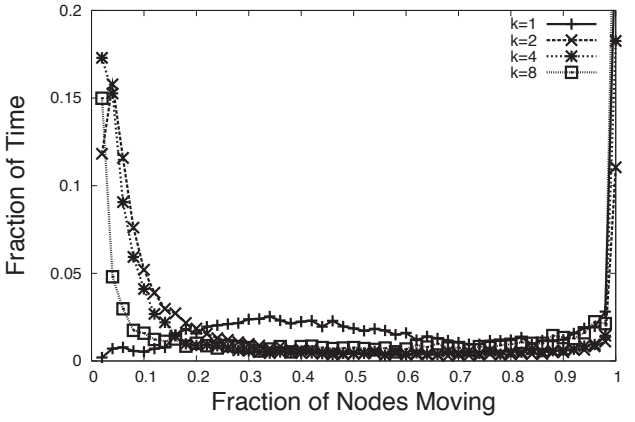


(d) ND, $\sigma = 0.2$

Fig. 16. Disconnection frequency.



(a) % nodes moving vs. time.



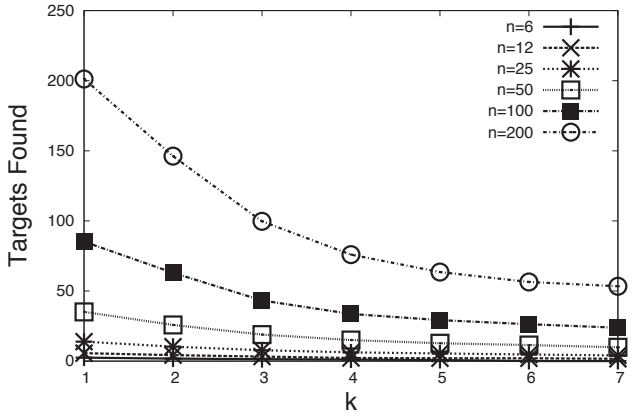
(b) Node movement PDF.

Fig. 17. Node movement $N = 200$, $STD = 5\%$.

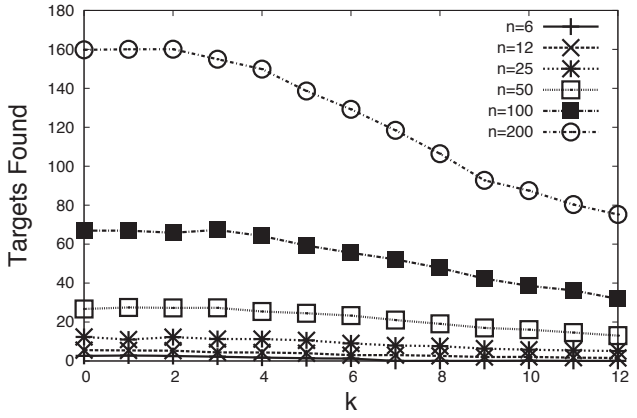
only registered as detected by the system, if the detecting node was connected to the base station through some path in the network at time of detection.

Targets were located uniformly throughout the space with a density of 2 per square mean broadcast radius. Performance was examined for the 5% STD broadcast case.

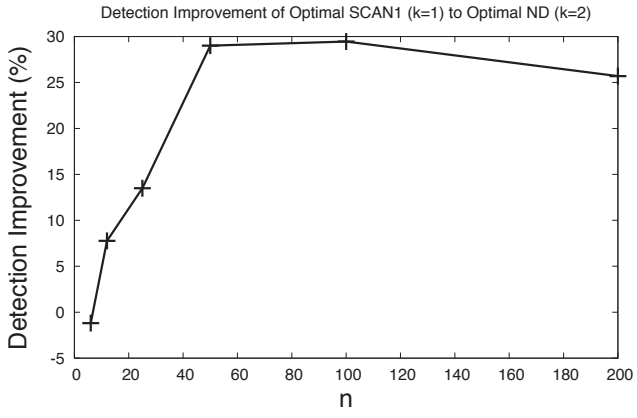
Fig. 18(a) shows the performance of SCAN at the target detection task plotting the number of targets detected (y-axis) against k (x-axis). For all curves, the number of targets found decreases monotonically with k. From this we might conclude that connectivity is of little value in the target detection task, even though the version we investigate requires some level of connectivity to detect targets! However, this would be a mistaken conclusion as comparison to the corresponding Fig. 18(b) for ND shows. When no provisioning for connectivity is supplied ND we see significantly poorer performance than that of SCAN. This can be seen in the direct comparison of optimally parameterized variants of SCAN and ND shown in Fig. 18(c) (y-axis - percentage improvement of SCAN over ND, x-axis - network size). The conclusion to be gained is that intelligent maintenance of some minimal level of connectivity is substantially helpful for the target detection task (above and beyond any other potentially advantage provided by connectivity maintenance) although overmuch and/or unintelligent connectivity provisioning will decrease performance.



(a) # targets detected vs. k for SCAN.

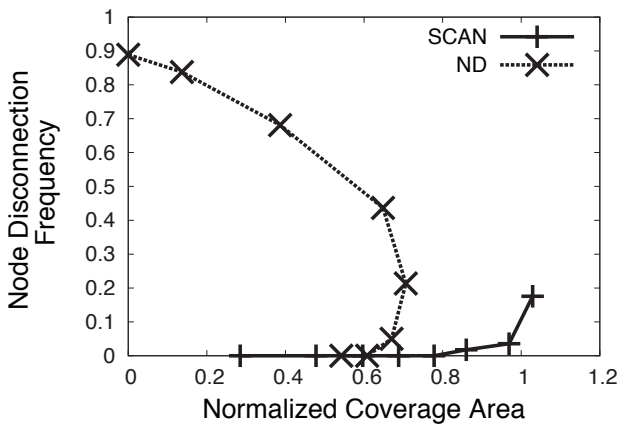


(b) # targets detected vs. k for SCAN.

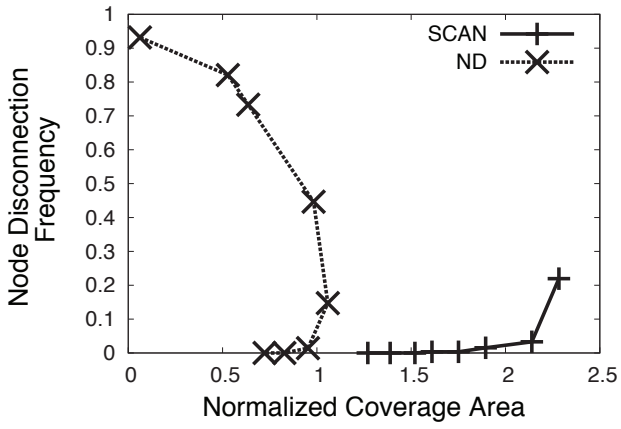


(c) % improvement of SCAN over ND.

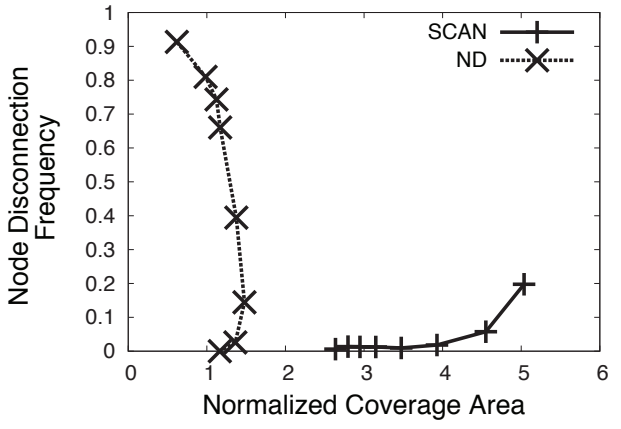
Fig. 18. Target detection.



(a) 8 nodes.



(b) 16 nodes.



(c) 32 nodes.

Fig. 19. SCAN vs. ND indoors.

D. Performance in Indoor Environments

In this set of simulation experiments we examined how SCAN and ND performed in a more complex environment, filled with walls that were obstacles to both wireless signal propagation and also to node movement. For SCAN, we found the relationship between k and the coverage area to be substantially similar to those of the obstacle-free environment, albeit with slightly higher rates of partition for a given k. However, things are very different for the parameterization of ND. In an indoor environment the presence of walls affects not only connectivity but also movement. Indoors, corridors and other obstacles increase the likelihood that a cluster of nodes begins moving in the same direction. When this occurs ND's behavior becomes pathological. Particularly if the number of nodes in the cluster is greater than k, then all nodes in the cluster will be able to move away from the rest of the network without ND's freezing criterion being triggered. Thus k must be made very high in order to maintain connectivity.

Partially as a consequence of this, partially because SCAN's functioning is little affected by obstacles, the difference in performance between them is significantly greater than in our obstacle-free environment. In Figs. VI-D we can see a comparison of the performance of both SCAN and ND for a systems of 8, 16, and 32 nodes. The x-axis of each figure plots the *normalized coverage area* (the total coverage area, divided the by broadcast area of a single node) and the y-axis plots the frequency of disconnections. Several aspects of these plots are noteworthy. The first trend seen is that as network size increases, SCAN outperforms ND by an increasing margin. As the number of nodes with which SCAN has to work increases, it will be able to cover an increasingly large area. However, ND which is very sensitive to obstacles cannot make much use of additional nodes, which stay trapped in a relatively small area. For this same reason, as the number of nodes increases, the variability of ND's resultant coverage area shrinks. Irrespective of the value k used, nodes running ND will be quickly stopped as obstacles server connectivity links below the movement threshold. This can be seen in the steady progression of ND from greatly bowed to much more mildly so.

Since each of these graphs displays a similar structure we will discuss Fig. 19(c). In this figure, even at its worst parameterization ($k = 1$), SCAN provides strongly bounded connectivity of 80%, while poorly parameterized ND provides almost no assurance of connectivity. Moreover, for a given minimum level of required connectivity, SCAN far outperforms ND, generally covering between 2 and 3 times greater area for the network sizes studied in this simulation experiment. Finally, in this plot, one can see that ND may produce the same area coverage for different node disconnection frequencies. When a low k is chosen, ND does a poor job at maintaining connectivity and no nodes are attached to the base, resulting in low coverage despite nodes traveling relatively far, while a high k causes the network to freeze before it has covered much area.

E. Correlation between Simulation and Experiments

The experimental design for our simulation experiments was necessarily different from the hardware experiments conducted on our testbed. This was both because of differences in goal (our simulation was designed to explore asymptotic proprieties and extend modeling, while our hardware experiments examined the function of SCAN and its suitability for use in a real-world setting) The natural constraints of our experimental space dictated the design of testbed experiments, our experimental space limited the number of mobile nodes and density of monitoring nodes we could reasonably we could reasonably deploy. Conversely our simulation environment could only provide a rough approximation of the vagaries of wireless broadcast (e.g., multi-path, fading, interference from competing systems) but was well suited towards exploring SCAN's behavior at large network scales.

Consequently, we might expect that SCAN's behavior on our hardware testbed would show little in common with it's behavior in our simulator (and analytic models). However, when we compared how the number of nodes moving evolved with time as SCAN was run in identically parameterized versions of both our simulation and experiment $N = 6, k = 2$, we found a substantial degree of correlation. This can be seen in Fig. 20 which compares the number of nodes moving (y-axis) for both the testbed and simulation against time (x-axis). [3] The degree to which movement patterns correlate between our simulation and testbed experiments indicate that the general trends of the results produced in each experimental domain bear significant applicability to the other.

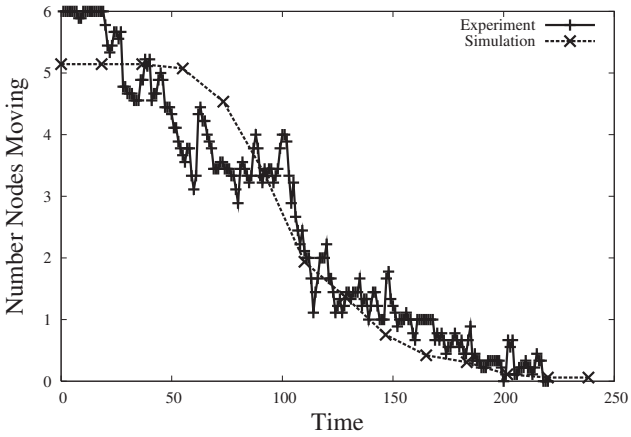


Fig. 20. Number of nodes moving vs. time for $N = 6$.

F. Summary of Results

In this section, we identified the freezing phase transition point for networks running SCAN. Above this point the network will freeze and below this point the network will not. We find that our analytic approximation is a good fit for our simulated results. We then focused our attention on obstacle-free environments. Using simulation, we found SCAN superior, covering an area 1.5 times larger than ND. Finally, we examined SCAN's behavior under more realistic simulation conditions, finding that its behavior vis-a-vis optimal k remained substantially the same, but that its performance advantage over ND increased five-fold over that found in obstacle-free environments.

VII. Conclusion and Future Work

In this paper, we have presented SCAN, a fully-distributed, low-overhead, tunable protocol for maintaining the physical layer connectivity of a mobile wireless network. SCAN enables nodes to move about the space as they desire, freezing *only* when risk of further movement endangers the network's connectivity. By relying on local connectivity information instead of localization data combined with predictive wireless models, SCAN is able to handle noise and obstacles present in realistic settings. SCAN is *extensible*, allowing for the inclusion of additional information (e.g., RSSI), and *expects little* from the environment or hardware, making it an ideal connectivity maintenance mechanism for challenged hardware/environments, or to serve as a backup mechanism for higher-performance techniques with stronger environmental/hardware requirements. When tested on hardware in a

[3]We compare with the higher variability simulation link model with link length $\sigma = 20\%$ of mean length and appropriate time rescaling.

challenging indoor environment, we found SCAN allowed for significant area coverage while robustly maintaining full network connectivity over 99% of the time.

REFERENCES

[1] "DARPA LANdroids project." [Online]. Available: http://www.darpa.mil/ipto/programs/ld/ld.asp

[2] J. Fink and V. Kumar, "Online methods for radio signal mapping with mobile robots," in *Proc. IEEE ICRA*, May 2010.

[3] D. Spanos and R. Murray, "Motion planning with wireless network constraints," *Proc. IEEE ACC*, June 2005.

[4] M. Zavlanos and G. Pappas, "Distributed connectivity control of mobile networks," in *Proc. IEEE CDC*, Dec. 2007.

[5] J. Reich, V. Misra, D. Rubenstein, and G. Zussman, "Spreadable connected autonomic networks (SCAN)," Columbia University, Tech. Rep. CUCS-016-08, Mar. 2008.

[6] D. Spanos and R. Murray, "Robust connectivity of networked vehicles," in *Proc. IEEE CDC*, Dec. 2004.

[7] M. Zavlanos and G. Pappas, "Controlling connectivity of dynamic graphs," in *Proc. IEEE CDC-ECC*, 2005.

[8] ——, "Potential fields for maintaining connectivity of mobile networks," *IEEE Trans. Robotics*, vol. 23, no. 4, pp. 812–816, Aug. 2007.

[9] N. Bezzo and R. Fierro, "Tethering of mobile router networks," in *Proc. IEEE ACC*, June 2010, pp. 6828 –6833.

[10] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," in *Proc. IEEE ICRA*, May 2010.

[11] Z. Yao and K. Gupta, "Backbone-based connectivity control for mobile networks," in *Proc. IEEE ICRA*, May 2009.

[12] C. Dixon and E. Frew, "Controlling the mobility of network nodes using decentralized extremum seeking," in *Proc. IEEE CDC*, 2006.

[13] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus, "From theory to practice: Distributed coverage control experiments with groups of robots," in *Proc. International Symposium on Experimental Robotics*, Athens, Greece, Jul. 2008.

[14] N. Michael, M. Zavlanos, V. Kumar, and G. Pappas, "Maintaining connectivity in mobile robot networks," in *Experimental Robotics*, 2009.

[15] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, "Maintaining network connectivity and performance in robot teams," *Field Robotics*, vol. 25, pp. 111–131, January 2008.

[16] M. Souryal, A. Wapf, and N. Moayeri, "Rapidly-deployable mesh network testbed," in *Proc. IEEE GLOBECOM*, 2009.

[17] N. Atay and B. Bayazit, "Mobile wireless sensor network connectivity repair with k-redundancy," in *Algorithmic Foundation of Robotics VIII*, ser. Springer Tracts in Advanced Robotics, G. Chirikjian, H. Choset, M. Morales, and T. Murphey, Eds. Springer Berlin / Heidelberg, 2009, vol. 57, pp. 35–49.

[18] N. Correll, J. Bachrach, D. Vickery, and D. Rus, "Ad-hoc wireless network coverage with networked robots that cannot localize," in *Proc. IEEE ICRA*, May 2009.

[19] J. Reich, V. Misra, and D. Rubenstein, "Roomba MADNeT: a Mobile Ad-hoc Delay Tolerant Network Testbed," *ACM MC2R*, Jan. 2008.

[20] F. Zeiger, N. Kraemer, and K. Schilling, "Commanding mobile robots via wireless ad-hoc networks: A comparison of four ad-hoc routing protocol implementations," in *Proc. IEEE ICRA*, May 2008.

[21] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proc. ACM SENSYS*, 2004.

[22] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," in *Proc. ACM MOBICOM*, Aug. 2000.

[23] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, and T. C. Chiueh, "MiNT-m: an autonomous mobile wireless experimentation platform," in *Proc. ACM MOBISYS*, 2006.

[24] I. Tsigkogiannis and et. all., "Dynamically configurable robotic sensor networks," in *Proc. ACM SENSYS*, 2005.

[25] [Online]. Available: http://www.cs.columbia.edu/~reich/research_connectivity.php

[26] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *Proc. ICCS*, 2004.

[27] D. Cichon and T. Kurner, "Euro-cost 231 final report," Tech. Rep., 1998.

[28] D. Chavey, "Tilings by regular polygons—ii: A catalog of tilings," *Computers and Mathematics with Applications*, no. 17, 1989.

[29] T. Moscibroda, R. Wattenhofer, and A. Zollinger, "Topology Control Meets SINR: The Scheduling Complexity of Arbitrary Topologies," in *Proc. ACM MOBIHOC*, May 2006.

[30] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan, "A general model of wireless interference," in *Proc. ACM MOBICOM*, Sept. 2007.

Joshua Reich is currently a Postdoctoral researcher in the Department of Computer Science at Princeton University. He received his B.A. in Mathematics (magna cum laude) from Columbia College, Columbia University in 2002. He received his M.S. in Computer Science in 2004 and his Ph.D. in Computer Science in 2011 from Columbia University's School of Engineering and Applied Science and Graduate School of Arts and Sciences, respectively. His research interests are in software defined networking. virtualization, cloud computing, green computing, and, more broadly, in the areas of networks and operating systems. He has interned with Microsoft Research (Redmond and Bangalore), Technicolor (Paris), and Sandia National Labs (Albuquerque and Livermore). He was selected as a 2011 NSF Computing Innovation Fellow by the CRA. He has been awarded a NSF GK-12 Graduate Teaching Fellowship, the Judge's Award at the SIGMOBILE student demo competition, back-to-back Extraordinary TA Awards, and was selected as an ARC Student Poster Finalist at SIGCOMM. Joshua is the inventor-of-record on several patents and his work on quick and scalable software reduplication in the cloud led to the founding of a startup, Silverlining Systems where he is a consulting co-founder.

Vishal Misra (S'98, M'99) is an Associate Professor in and the Vice Chair of the Computer Science Department at Columbia University. He has received an NSF CAREER Award, a DoE CAREER Award, a Google Research Award and IBM Faculty Awards. His research emphasis is on mathematical modeling of computer systems, bridging the gap between practice and analysis. His recent work includes the areas of cloud computing, peer to peer systems, Internet economics and efficient scheduling policies. He has served as the guest editor for the Journal of Performance Evaluation, chaired the ACM Sigmetrics conference and participated as member of program committees for conferences such as IEEE Infocom, ACM Sigmetrics, ACM SiGCOMM, IFIP Performance and IEEE ICNP. He is is currently on leave co-fouding a startup, Silver Lining Systems where he is President and CEO.

Dan Rubenstein is an Associate Professor in the Department of Computer Science at Columbia University. He received a B.S. degree in mathematics from M.I.T., an M.A. in math from UCLA, and a PhD in computer science from University of Massachusetts, Amherst. His research interests are in network technologies, applications, and performance analysis. He is an editor for IEEE/ACM Transactions on Networking, was program chair of IFIP Networking 2010 and ACM Sigmetrics 2011, and has received an NSF CAREER Award, IBM Faculty Award, the Best Student Paper award from the ACM SIGMETRICS 2000 conference, and Paper awards from the IEEE ICNP 2003 Conference, ACM CoNext 2008 Conference, and IEEE Communications 2011. He spent the 2011 year at Google, and is currently on leave co-fouding a startup, Silver Lining Systems where he is Chief Scientist.

Gil Zussman (S'02-M'05-SM'07) received the B.Sc. and B.A. degrees from the Technion in 1995 (both summa cum laude). He received the M.Sc. degree (summa cum laude) from Tel-Aviv University in 1999 and the Ph.D. degree in Electrical Engineering from the Technion in 2004. Between 2004 and 2007 he was a Postdoctoral Associate at MIT. He is currently an Assistant Professor of Electrical Engineering at Columbia University. His research interests are in the area of wireless networks. He has been an associate editor of IEEE Transactions on Wireless Communications and Ad Hoc Networks, and the TPC co-chair of IFIP Performance 2011. He is a co-recipient of the IFIP Networking 2002 Best Student Paper Award, the OPNETWORK 2002 and ACM SIGMETRICS 2006 Best Paper Awards, and the 2011 IEEE Communications Society Award for Advances in Communication. He was a member of a team that won first place in the 2009 Vodafone Foundation Wireless Innovation Project competition, and is a recipient of the Marie Curie Outgoing International Fellowship, the Fulbright Fellowship, the DTRA Young Investigator Award, and the NSF CAREER Award.