Hierarchical Routing over Dynamic Wireless Networks

Dominique Tschopp, Suhas Diggavi, and Matthias Grossglauser
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
Email: first.last@epfl.ch

Abstract

The topology of a mobile wireless network changes over time. Maintaining routes between all nodes requires the continuous transmission of control information, which consumes precious power and bandwidth resources. Many routing protocols have been developed, trading off control overhead and route quality. In this paper, we ask whether there exist low-overhead schemes that produce low-stretch routes, even in large networks where all the nodes are mobile. We present a scheme that maintains a hierarchical structure within which constant-stretch routes can be efficiently computed between every pair of nodes. The scheme rebuilds each level of the hierarchy periodically, at a rate that decreases exponentially with the level of the hierarchy. We prove that this scheme achieves constant stretch under a mild smoothness condition on the nodal mobility processes. Furthermore, we prove tight bounds for the network-wide control overhead under the additional assumption of the connectivity graph forming a doubling metric space. Specifically, we show that for a connectivity model combining the random geometric graph with obstacles, constant-stretch routes can be maintained with a total overhead of $n \log^2 n$ bits of control information per time unit.

Index Terms

distributed routing algorithms; wireless networks; geometric random graphs; competitive analysis; mobility

I. Introduction

Because of node mobility, the connectivity of (mobile) wireless networks can change over time. In order to route information in such networks, one needs to maintain routing paths as the network topology evolves. We would like these routes to be efficient and adaptive to the topology changes. Clearly, the maintenance of these routes incurs control traffic, which consumes bandwidth and power. Finally, we would like routing to be distributed so that no central routing entity is needed. The central question we address in this paper is the design of routing algorithms for dynamic wireless networks, for which we can provably quantify the routing efficiency and the control traffic overhead. In our performance guarantees, we focus on dynamically changing connectivity graphs that arise in wireless networks, but the overall algorithmic ideas are applicable to any continuously connected dynamically changing network.

Our measure of efficiency of the routes is in the form of a guarantee that the route is within a (small) constant factor, called *stretch*, of the shortest path length, with high probability over the connectivity graph. We assume inherently that the round-trip time (RTT) of a packet from source to destination is much smaller than the time-scale of topology change, so that a packet traverses a fixed graph. Our algorithm finds the route to a destination, which is identified through a fixed identity (name); it does not require the destination's current location[1].

In order to analyze the performance of these algorithms for mobile wireless networks, we need to model the dynamics of the connectivity graph. This is captured by modeling both the spatial distribution and the mobility processes of nodes. We analyze the routing and overhead performance of our algorithms under several widely studied models for node placements and mobility. For example, our results apply to models such as the geometric random graph model with connectivity radius growing as $\sqrt{\log n}$ with network size n; the fully connected regime of the dense or extended wireless network with signal-to-interference-plus-noise ratio (SINR) threshold connectivity; and some examples of networks with obstacles and non-homogeneous node distribution. As explained in Section II-A, we assume that all the nodes have access to a synchronized clock, and that transmissions of messages between directly connected nodes are free of transmission errors and congestion. These assumptions allow us to focus on the dynamic maintenance of the routing structure as the main question addressed in this work.

The dynamics of the network generate a sequence of wireless connectivity graphs. The mobility process and channel model impose constraints on the evolution of the connectivity graph with time, which we exploit explicitly in the design of our algorithm. Intuitively, topology changes due to mobility cannot cause very distant nodes to be connected after a very short period of time, or vice versa. This is captured by the notion of *smooth* mobility changes, which is essentially a limit on how quickly the connectivity graph distance can change (see Section II for precise definition).

Our main results in this paper are the following. (i) For smooth geometric sequence of connectivity graphs, we develop a routing strategy based on a hierarchical set of beacons with scoped flooding. We also maintain cluster membership for these beacons in a lazy manner adapted to the mobility model and doubling dimension. (ii) We develop a worst-case analysis of the routing algorithm in terms of total routing overhead and route quality (stretch). We show that we can maintain constant stretch routes with an average network-wide traffic overhead of $O(n \log^2 n)$ bits per time unit. A load-balanced version of the algorithm, which reduces the peak control load at high-level beacon nodes, requires $O(\log^3 n)$ bits per node per time unit. Through numerics we show that the theoretically obtained worst-case constants are conservative.

A. Related Work

Routing in wireless networks has been a rich area of enquiry over the past decade or more. The two main paradigms for routing have been geographic routing and topology based routing. Geographic routing (see for instance [KK00] and references therein) exploits the inherent geometry of wireless networks, and bases routing decisions directly on the Euclidean coordinates of nodes. Their performance depends on how well the Euclidean coordinate system captures the actual connectivity graph, and these approaches can therefore fail in the presence of

[1]This is in contrast to most analyses of geographic routing performance, where it is assumed that the location of the destination is already available. In practice, this is achieved through a *location service* that records the current location of every node. The location service itself incurs overhead, of course, both for *updates* (a node moves and udpates its location with the location service) and for *lookups* (a node looks up the current location of a destination). The protocol described in this paper finds routes to destinations identified directly by *name*, and should therefore be compared to a combined location service plus geo-routing protocol.

node or channel inhomogeneity (like in Figure 2 in Section II). This has led to a line of work designing fallback schemes to recover from dead-ends, usually by introducing some additional state in nodes and data messages. Another important, but often overlooked, issue with geo-routing is that geographical positions of the nodes need to be stored and continuously updated in a distributed database in the network, to allow sources of messages to determine the current position of the destination. This database is called a *location service* (see for instance [LJDC$^+$00]) and must be regularly updated so that source nodes can query it. Location services typically rely on some a-priori knowledge of the geographical boundaries of the network. This is necessary because these approaches typically establish a correspondence (for example, through a hash function) between a node identifier and one or several geographical locations where location information about that node is maintained. An important feature of our work is that we consider the total overhead incurred by the update and lookup operations of the location service, and the overhead of the routing algorithm itself.

Topology based routing schemes (see [PR97] and [JMB01]) do not utilize the underlying geometry of wireless connectivity graphs, but instead compute routes based directly on that graph. To reduce overhead, most of these schemes only establish routes *on demand* through a route discovery operation, rather than continuously maintaining a route between every pair of destinations; in this respect, they differ significantly from their counterparts for the wired Internet (such as OSPF, IS-IS, and RIP). Recently established routes are cached in order to allow their reuse by future messages. In distance-vector based approaches (e.g., [PR97], this cached state resides in the intermediate nodes that are part of a route, whereas in source-routing approaches (e.g., [JMB01]), the cached state resides in the source of a route. Despite such optimizations, topology-based approaches suffer from the large overhead of frequent route discovery operations in large and dynamic networks. This issue was, in fact, the reason why geo-routing approaches have reached prominence.

Two schemes that utilize the underlying geometry of graphs in *static* wireless network algorithms are the works presented in [RRP$^+$03] and the beacon vector routing (BVR) introduced in [FRZ$^+$05]. Both these schemes are heuristics which build a virtual coordinate system over which routing takes place. They were shown to work well through numerics. However, they utilize an external addressing scheme to make a correspondence between addresses and names. Other protocols based on creating hierarchical levels based on creating clusters of geometrically reducing radius have been studied in [GGC00], [HP01], [YC05], [Jac06]. Most of these schemes have been developed for *static* networks. A notable exception is [Jac06], which applies a link-state approach to dynamic networks, which is different from the approach developed in our work. From our understanding of this work, the overhead is significantly larger than our work. In [TDGW07], routing on dynamic networks using a virtual coordinate system was studied. For large scale dynamic wireless networks, these heuristics pointed to significant advantages to using some geometric properties for routing and addressing. These results motivated the questions studied in this paper.

There has been a vast amount of theoretical research on efficient routing schemes in wired (*i.e.,* static) networks (see for example [Gav01]). Most of this work has been focused on the trade-off of memory (routing table size) and routing stretch. There are two main variants of such routing schemes (i) *labeled* (or *addressed*) routing schemes, where the nodes can be assigned addresses so as to reflect topological information; (ii) *named* routing, where nodes have arbitrary names, and as part of the routing, the location (or address) of the destination needs to be obtained (similar to a location service). This examines the important question of how the node addresses need to be published in the network.

Routing in graphs with finite doubling dimension has been of recent interest (see [KRX06], and references therein). In particular [Tal04] showed that one could get constant stretch routing with small routing table sizes for doubling metric spaces; this relies on labeled routing, i.e., we are allowed to label the nodes as a function of the routing structure. This result was improved to make routing table sizes smaller in [CGMZ05]. The problem of named routing over graphs with small doubling dimension has been studied in [KRX06] and [AGGM06], and references therein.

To the best of our knowledge, there has been no prior work on *dynamic* graphs over doubling metric spaces and on control traffic overhead. It is worth pointing out that there is no direct correspondence between control traffic and memory. Bounds on memory do not take into account the amount of information which needs to be transported in the network in order to build and maintain routing tables.

In short, we believe that the work in this paper is distinct from the literature in the following aspects. The new formulation of the tradeoff between control overhead traffic versus performance (route stretch) for *dynamic* graphs over doubling metric spaces leads to several new technical questions. In particular, the need for maintenance of consistent routes and addressing leads to some of the novel technical contributions in our work, including algorithms for time-varying hierarchical beaconing, lazy membership updates for addressing, and the performance analysis of inhomogeneous dynamic networks.

II. Models and Definitions

A. Assumptions and Notation

A wireless network consists of a set of n nodes spread across a geographic area in the two-dimensional plane. We model the network region as the square area $[0, \sqrt{n}) \times [0, \sqrt{n})$. The n nodes move randomly in this area; we denote by $x^{(t)}(u)$ the position of node u at time t. The connectivity between two nodes is represented by an edge on the connectivity graph $\mathcal{G}_n^{(t)}$ if they can communicate *directly* over the wireless channel. The connectivity between two nodes depends on the distance between the two nodes (and could also depend on the presence of other nodes, see Section VII-B). We consider that when a node u transmits on the wireless channel, it broadcasts to all its neighbors in the connectivity graph $\mathcal{G}_n^{(t)}$. Consequently, one transmission of a packet is sufficient for all direct neighbors to receive that packet. To make the notation lighter, we will only add the dependence on time if it is necessary to avoid confusion.

To simplify the analysis, we assume a time-slotted system whose connectivity graph only evolves instantaneously between two time slots, but remains fixed during each slot. The length ΔT of a time slot is set to a constant short enough so that the evolution of connectivity in the time-slotted system is close to that of its continuous-time counterpart. For example, if the application at hand is a vehicular ad hoc network, where the communication radius is $100m$ and the distance between two cars is at most $160kph$, the duration of a contact would be at least $2sec$; a reasonable choice for ΔT might therefore be $1sec$ in this application. However, we assume that ΔT is much larger than the round trip time (RTT) through the network, *i.e.,* the time needed for a message to be sent wirelessly between any pair of nodes, and a response to be sent back. As will become clear later in the description of our algorithm, this assumption mostly concerns small control messages[2], which might have a size of a few bytes in typical applications. For example, to continue the scenario above, a vehicular network with $n = 10'000$ nodes would have a diameter on the order of $\sqrt{n} = 100$; it would take about $10ms$ to send a $100bit$ control message end-to-end for a channel rate of $10Mbps$. We incorporate this time-scale separation between changes in connectivity and communication latency into our model by assuming that changes in connectivity only occur every ΔT time units, and that communication can take place instantaneously within each such time slot. Furthermore, we assume that there is a random permutation π on the nodes, and that all nodes in the network know their rank in π. In Section VII we drop these assumptions and consider practical aspects of the implementation. Finally, we say that a result holds with high probability (w.h.p.) if it holds with probability at least $(1 - O(n^{-\rho}))$, for some constant $\rho > 0$. In Table I, we summarize the notations used in this paper.

$x^{(t)}(u)$	Position of node u at time t
$d^{(t)}(u, v)$	Shortest path distance from u to v at time t
ρ_n	Wireless communication radius
$\mathcal{G}(n, \rho_n)$	Random geometric graph
$\mathcal{B}_R^{(t)}(u)$	Ball of radius R around u
$\kappa(\tau, d)$	Smoothness constraint on the sequence of connectivity graphs

TABLE I
Table of notations

The distance $d^{(t)}(u, v)$ between nodes u and v is the shortest path distance between these nodes in $\mathcal{G}_n^{(t)}$. Note that $d(.,.)$ is a metric on $\mathcal{G}_n^{(t)}$, *i.e.,* the distance between a node and itself is zero, the distance function is symmetric and the triangle inequality applies. We now define a *ball* of radius R around a node u, which is simply the set of nodes within distance R of u. More formally, we can define it more generally for any metric space as follows:

Definition 1: A *Ball* $\mathcal{B}_R^{(t)}(u)$ around node u at time t in a metric space X is the set $\{v \in X | d^{(t)}(u, v) \leq R\}$.
In order to bound the control traffic overhead, we will recursively subdivide the connectivity graph into balls of exponentially decreasing diameter. It will be crucial for us to bound the number of balls of radius R necessary to

[2]Actual data payload can always be segmented to adjust the message size.

cover a ball of radius $2R$ around some node u. In other words, we want to find the smallest set of nodes v_i such that all nodes within $2R$ of u are also within R of some node v_i. The notion of doubling dimension of a metric space captures this idea.

Definition 2: The *doubling dimension* $\log \alpha$ of a metric space X is given by the smallest α such that any ball of radius $2R$ can be covered by at most α balls of radius R, for all $R \geq \min_{(u,v)} d(u, v)$, *i.e.,* $\forall u \in X \ \exists \ S_u \subseteq X, |S_u| \leq \alpha$ and

$$\mathcal{B}_{2R}^{(t)}(u) \subseteq \bigcup_{j \in S_u} \mathcal{B}_{R}^{(t)}(j)$$

Moreover, if α is a constant, we have the following definition:

Definition 3 (Doubling metric space): A metric space X is *doubling* if its doubling dimension is a constant.
A good way to illustrate and understand the concept of doubling dimension and doubling metric space is to look at the metric space defined by a set of points X in $\mathbb{R}^2$ with the Euclidean distance. A ball of radius $2R$ around a point x is simply a disk of radius $2R$ around this point. To cover this disk, we select a set of points such that all the surface is covered by the corresponding set of disks of radius R. Note that the number of disks required does not depend on R, and consequently this metric space is *doubling* (see Figure 1).

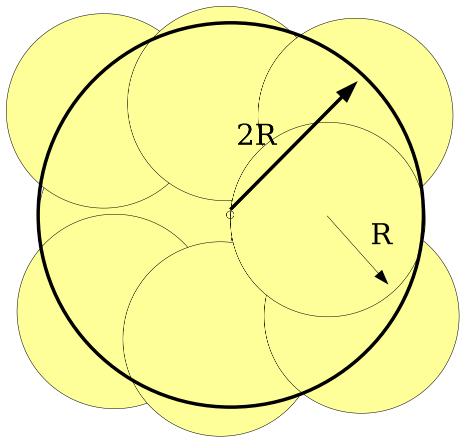


Fig. 1. The metric space defined by a set of points in $\mathbb{R}^2$ and the Euclidean distance is doubling. Indeed, we can cover a disk of radius $2R$ by a constant (8 in this case) number of disks of radius R, whatever the value of R.

In Section II-B, we describe the geometric random graph model as the canonical model we use to illustrate the ideas of the paper. We also give an example of a inhomogeneous network to which our results can be applied. In Section VII-B, we develop the model where connectivity is determined by the SINR, and we have uniform transmit power and full connectivity. We give the requirements for the mobility model to result in a *smooth* sequence of wireless network graphs in Section II-C. We state the underlying assumptions and give a table of notations in Section II-A.

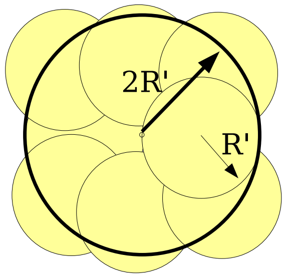
B. Sequences of Connectivity Graphs

The focus of this paper is on dynamic wireless networks, which are modeled as sequences of geometric random graphs and their variants with spatial inhomogeneities, defined below.

We denote the geometric random graph by $\mathcal{G}(n, \rho_n)$ and define its connectivity as follows.

Definition 4: A random geometric graph $\mathcal{G}(n, \rho_n)$ has an unweighted edge between nodes u and v if and only if $||x(u) - x(v)|| < \rho_n$, where $\{x(u)\}$ are chosen independently and uniformly in $[0, \sqrt{n}) \times [0, \sqrt{n})$.
In this paper, we are interested in fully connected geometric random graphs, and therefore focus on the case $\rho_n > \sqrt{\log n}$ [GK98]. One of the main properties of random geometric graphs that we exploit in this paper is that $\mathcal{G}(n, \rho_n)$ is a doubling metric space for the shortest-path distance metric. This is formally proved in Theorem 2 in Section IV.

We define a sequence of random graphs $\mathcal{G}^{(t)}(n, \rho_n)$ with unweighted edges between u and v at time t if and only if $||x^{(t)}(u) - x^{(t)}(v)|| < \rho_n$. Whether each graph in the sequence $\mathcal{G}^{(t)}(n, \rho_n)$ corresponds to a random geometric



graph as in Definition 4 depends on the mobility model for the nodes. We discuss this in more detail in Section II-C. For mobility models satisfying a mild regularity condition (see Section II-C), we show that the sequence of graphs from these mobility models result in a sequence of doubling metric spaces (see Theorems 2 and 3 in Section IV).

We also study variants of random geometric graphs that incorporate spatial inhomogeneities such as obstacles. In Figure 2, we illustrate a inhomogeneous random network where connectivity is not completely geometric as in Definition 4. An obstacle prevents communication between neighboring nodes, and therefore illustrates the complexities of wireless network connectivity. This example is revisited in Section VI, where we show that although this connectivity graph is more complicated than $\mathcal{G}(n, \rho_n)$, it is still doubling, and therefore the algorithms developed in this paper are applicable. This illustrates the advantage of our approach to network modeling.

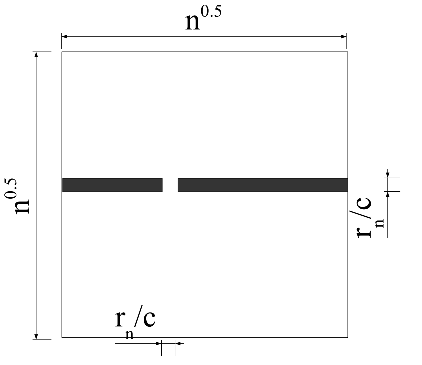


Fig. 2. n nodes are distributed uniformly at random on a square area of side $\sqrt{n}$. A wall of width ρ_n/c is added, which only has a small hole in the middle. Again, we assume $\rho_n > \sqrt{\log n}$. Nodes cannot communicate through the wall. Finally, we remove the nodes below the wall, which leads to an inhomogeneous node distribution.

C. Uniform Speed-Limited (USL) Mobility

We assume that nodes are mobile and move according to the uniform speed-limited (USL) model, a fairly general mobility model defined next. The USL model essentially embodies two conditions: (i) the node distribution at every time step is uniform over the network domain (which may contain obstacles, as explained above), and (ii) the distance a node can travel over a time step is bounded. We restrict ourselves to the case in which the maximum speed is not dependent on n. In practice, of course, such an assumption is realistic since the maximum speed of the nodes will not increase when new nodes join the network.

Definition 5: A collection of n nodes satisfies the uniform speed-limited (USL) mobility model if the following two conditions are satisfied:

(i) At every time t, the distribution of nodes over the network domain is uniform;
(ii) For every node u and time t, the distance traveled in the next time step is bounded, i.e., $||x^{(t+1)}(u) - x^{(t)}(u)|| < S$.

The USL mobility model is quite general. For example, it includes the following cases: (i) The nodes perform independent random walks with bounded one-step displacement. The random walks can be biased, and the displacement distribution does not need to be homogeneous over the node population. We have to assume that the nodes operate in the stationary regime. (ii) The nodes follow the random waypoint model (RWP)[3]. The system has to be in the stationary regime. (iii) The generalized random direction models from [SMS06], which interpolate between the random walk and the random waypoint cases, through a control parameter that can be viewed as the "locality" of the mobility process. (iv) We can also allow for models where nodes do not move independently. As an illustrative example, assume we uniformly place nodes on the square; the nodes then move in lockstep according to any speed-limited mobility process, maintaining their relative positions to each other. Observe that the uniform distribution is maintained for all time steps (note that we move on a torus), and that the speed-limited property is true by definition.

[3]The uniform distribution requires a torus in this model [Bet03].

We see that the USL class of mobility models is fairly general, and includes many of the models that have been proposed in the literature. For simplicity, we consider that time is discrete. In other words, we look at a snapshot of the network every ΔT seconds. At every time step, the connectivity between nodes can change. Hence, we will work with a sequence of connectivity graphs. In order to design a routing algorithm with a low control traffic overhead, we need to understand how fast the graph distances between nodes can evolve over time. In particular, consider two nodes u and v at distance $d = d^t(u, v)$ at time t. We want to bound the multiplicative factor κ by which this distance can change in τ time steps. Formally, we define $\kappa(\tau, d)$ as follows:

Definition 6: We say that a communication network is $\kappa(\tau, d)$-smooth if the shortest path distance between any two nodes u an v at shortest path distance d cannot change by more than a factor $\kappa(\tau, d)$ in τ time steps, *i.e.,* $\forall u, v$, we have:

$$\max\left\{\frac{d^{(t)}(u,v)}{d^{(t+\tau)}(u,v)}, \frac{d^{(t+\tau)}(u,v)}{d^{(t)}(u,v)}\right\} \leq \kappa(\tau, d^{(t)}(u,v))$$

For most of the results of our paper, we need only a simpler characterization of mobility, which is defined as follows.

Definition 7: We say that the network is κ*-smooth* if there exists an integer constant $\nu \geq 1$ such that $\kappa(\nu d, d) \leq \kappa$ for all $d \geq 1$.

Informally, in a κ-smooth network, distances are allowed to grow at the same maximum speed at all scales. In the sequel, we will bound κ and ν for our model. For example, this smoothness property holds for a general class of *random trip* mobility models studied in [BV05], whose stationary distribution is shown to be uniform on the network domain $[0, a) \times [0, a)$ and ergodic, and satisfies the USL constraint of Definition 5.

III. Dynamic Hierarchical Embedding and Routing

We now develop the distributed data structure and associated routing algorithm, and discuss their key properties. We defer to Section V a rigorous performance analysis, in terms of control traffic overhead and routing stretch, for a general class of dynamic networks.

The main concept underlying our approach is to maintain a hierarchical decomposition of the network such that a constant-stretch route exists between any pair of nodes. This structure is routable, in the sense that an efficient route can be determined through local decisions within the hierarchy. Essentially, the source of the message probes increasingly coarse levels of the hierarchy until it identifies a region in the decomposition that contains the destination; the message is then passed down from this level to increasingly finer levels to deliver the message to the destination. The key idea is in the dynamic maintenance of the hierarchy as the network evolves: the finer levels of the hierarchy are updated frequently, while the coarser levels are allowed to age before they are refreshed, in order to reduce control traffic. Between two updates at a given level in the hierarchy, the stretch increases with time because of node mobility. The update frequency of a level therefore trades off the control overhead to rebuild this level with the increased stretch due to outdated routes.

We make a few simplifying assumptions to give a compact description of the data structure and algorithm. We assume time t is slotted, and that each slot has two separate phases, a *maintenance* phase and a *forwarding* phase. Changes in network topology due to mobility only happen between time slots. We also assume that all the nodes have access to a synchronized clock, and that transmissions of messages between directly connected nodes are free of transmission errors and congestion. These assumptions allow us to focus on the dynamic maintenance of the routing structure as the main issue; standard techniques (congestion control, ARQ, etc.) can be incorporated into the protocol to make it robust to more realistic channels as well as to mobility and traffic patterns.

During the maintenance phase, nodes exchange control information for the maintenance of the routing data structure. The actual data messages are carried during the forwarding phase. There are three types of control information: beaconing, registration, and probe messages. A beaconing message is sent by a beacon node and flooded to every node within a certain flooding radius around the beacon. Its purpose is to establish a reverse path from these nodes to the beacon, to be used in the forwarding phase. Every beacon is associated with a level in the hierarchy. Registration messages allow a newly elected beacon to collect the set of nodes that are members of its cluster. Probe messages are used during the forwarding of a message, to ask beacons whether a particular destination is in their membership.

In the sequel, we refer to a beacon at level i as an i-beacon, and a timeslot during which beacons up to level i are refreshed as an i-timeslot.

A. Beaconing and Registration Algorithm

We first list the data structures and variables maintained by every node, as well as the types and fields of messages that are exchanged in the two phases of the algorithm. The central data structure in every node is the routing table, as shown in Table II, in which each node records all the routes (next hops) towards the beacons from the top level down to its own level (note that the entries for the level immediately below the own level of a beacon record its children in the hierarchy):

Beacon node identifier	distance [hops]	level	next hop
$O(\log n)$	$O(\log \Delta)$	$O(\log \Delta)$	$O(\log n)$

TABLE II

ROUTING TABLE ROUTETABLE: FIELDS AND THEIR LENGTHS.

Other variables maintained by each node u are:

(i) π_u: this node's position in a permutation π of all n nodes, which controls the order in which the hierarchy is built;

(ii) myBeaconLevel: current beaconing level of the node; this can change in every time step due to changes in topology; note that every node is always considered a beacon, at least trivially at level 0;

(iii) refreshLevel: the highest level that is getting refreshed in the current timeslot (i.e., refreshLevel $= i$ in an i-timeslot); the routing and membership tables up to this level are cleared, new beacons for all levels up to and including refreshLevel are elected, and these new beacons collect their membership;

(iv) members[i]: the list of nodes that have registered with this beacon in the last i-slot;

There are three types of control messages: beaconing, registration, and probing. Each type of message has a fixed set of fields, each of which is either constant-sized (e.g., a flag) or of size at most $\log n$ (node identifier or level). More specifically, the fields and their function are listed in Table III. The beaconing messages are flooded around each beacon in such a way that each node within the flooding radius broadcasts at most one copy of this message. The purpose of these messages is to establish reverse routes towards each beacon at every level. The registration messages are used for nodes to announce their membership to their beacons. This only happens at every i-slot for level i. Registration messages are also sent in every time slot by a level $(i-1)$-beacon to its level i-beacons in order to maintain the downstream routes from every beacon to its children. Finally, the probing messages are used in the upstream forwarding phase for the source to check for membership amongst the possible beacons.

Field	*length*	*function*
id	$O(1)$	a unique message identifier for duplicate detection
type	$O(1)$	gives the message type (data, beaconing, registration, probe)
src	$O(\log n)$	source (originator) of the message
dst	$O(\log n)$	destination of the message; for flooding messages, this field has no function
dist	$O(\log \Delta)$	number of hops traveled so far
level	$O(\log \Delta)$	indicates the level, or an interval of levels (for beaconing, registration, and probe messages)
next-hop	$O(\log n)$	next hop on path

TABLE III

MESSAGE FIELDS, THEIR LENGTHS AND FUNCTIONS.

We now describe the maintenance phase of every time slot in more detail. Essentially, this phase updates the hierarchical data structure to account for changes in the network topology since the preceding time step. This involves two functions: beaconing and registration. First, at the beginning of every time step, the nodes invoke Algorithm 1 to send a beaconing message at their level. Furthermore, every i-beacon is up for reelection in a j-slot if $j > i$. The beacon election proceeds according to a permutation π of all the nodes in the network; each node awaits its turn in the permutation, and then elects itself as beacon at the highest level at which it does not yet belong to any beacon preceding it in[4] π. Beaconing happens at every time step, and amounts to each beacon at every level broadcasting a message within a certain graph radius (Algorithms 1 and 2). The main function of beaconing is to

[4]This is similar to the procedure proposed in [Tal04].

ensure that every node at every level knows its beacons and has an active route towards these beacons. As a result, each node knows at least one beacon at every level of the hierarchy[5].

Registration is a much more costly operation, which is triggered at exponentially decreasing frequencies going from low (fine) to high (coarse) levels. It amounts to a beacon collecting its *membership*, i.e., the set of nodes that are close to this beacon. Recall that we refer to a timeslot when membership registrations occur up to level i an i-timeslot. During a i-timeslot, each node receiving a level i beacon update returns a registration message to this beacon (in other timeslots, the node merely remembers the route to the beacon, but remains silent.) This membership update is described in Algorithm 3, with respect to how the beacon updates its memberships.

The main difficulty in maintaining the hierarchical embedding without completely rebuilding the structure in every time step lies in the registration update schedule. In an i-timeslot, each node registers with a level-i beacon if it lies within a *cover radius* of r_i hops of this beacon. However, as the network evolves after the registration timeslot, this distance can increase or decrease, due to node mobility. For this reason, each beaconing message by an i-beacon is broadcast over a distance of $f_i > r_i$ hops. This *flooding radius* f_i is chosen such that any node within distance r_i of an i-beacon during an i-timeslot is within f_i of this beacon in any other timeslot, despite the movement by both the node and the beacon after the registration. Clearly, the relationship between f_i and r_i depends on the mobility model and the registration schedule; in Section V we explore their interplay for mobility processes defined in Section II-C, Definitions 5 and 7.

Let the *cover radius* at level i, for $i = 1, ..., \log \Delta$ ($\Delta = O(n)$ being the diameter of the network), be defined as $r_i = 2^i$ and the *flooding radius* at level i be defined as $f_i = \kappa(r_{i+1} + r_i)$, where κ is a constant chosen such that $\kappa(\nu d, d) \leq \kappa$, $\forall d$, as in Definition 7. In order for the routing algorithm to work properly, it is crucial that beacons at level i are within the flooding radius of the beacons at level $i + 1$, if they have common nodes inside their cover radii (see Fig. 3). This is why we define the flooding radius above. Note that if the network is static, we can set κ to 1. Else, the value of κ depends on how mobile the nodes are (see Definition 7).

Note that due to the κ-smooth mobility model of Definition 7, the distances $d^{(t)}(u, v)$ between two nodes u and v cannot change by a factor κ in less than νd time steps. In particular, if a node u is at distance 2^i from a beacon v at the time it becomes a member of its cluster, then we have $d^{(t+\nu 2^i)}(u, v) \leq \kappa 2^i$. Hence, we update the memberships at level i only every $\nu 2^i$ time steps (see Figure 4). This leads to a routing scheme in which the distances can be distorted by at most a constant factor, to be calculated in the sequel. Additionally, in a dynamic environment, routes can break. To reestablish them, we let the beacons at all levels flood at every time step. Levels at which no membership updates take place simply use the floods of the beacons to update their routes toward theses beacons. This will ensure that a route always exists for all pairs of nodes.

In Figure 5, we give a simple example with three levels. The beaconing algorithm is presented in Algorithms 1-4. It is important to note that the routes are updated at every time step and consequently routing towards a beacon always succeeds. Further, when the membership at a given level i is updated, all the memberships at the levels $j < i$ will also be updated, after canceling all prior memberships at these levels.

B. Forwarding Algorithm

The forwarding algorithm described in Algorithm 4 has two phases, an upstream and a downstream phase. In the upstream phase, a source node u with a message for a target node v searches for v by first probing all the level 1 beacons it knows of (i.e., contained in its routing table). If all the answers are negative, node u probes all level 2 beacons it knows of, and so on until it receives a positive answer from at least one beacon. The message is sent to this beacon, which starts the downstream phase. In the downstream phase, an i-beacon holding the message asks all of its $(i - 1)$-beacons (i.e., $(i - 1)$-beacons that are registered with this i-beacon) whether the destination is in their membership set. We establish in Section V that this must be the case for at least one lower-level beacon.

The i-beacon forwards the message to one of the $(i - 1)$-beacons having answered positively, and the process repeats recursively until the message is delivered to the destination (cf. Figure 6).

[5]We emphasize that the beaconing protocol described here assumes the *synchronous* forwarding of the flooding message, so that the first copy of the message received by a node arrives on the shortest path from the beacon to that node. In more challenging environments, where a copy of a beaconing message might reach a node on a longer path before a second (delayed) copy arrives on the shortest path, care needs to be taken to eliminate duplicate messages and update the reverse paths appropriately. This is outside the scope of this paper, and we refer the reader to [PC12] for an approach to robust and efficient flooding.

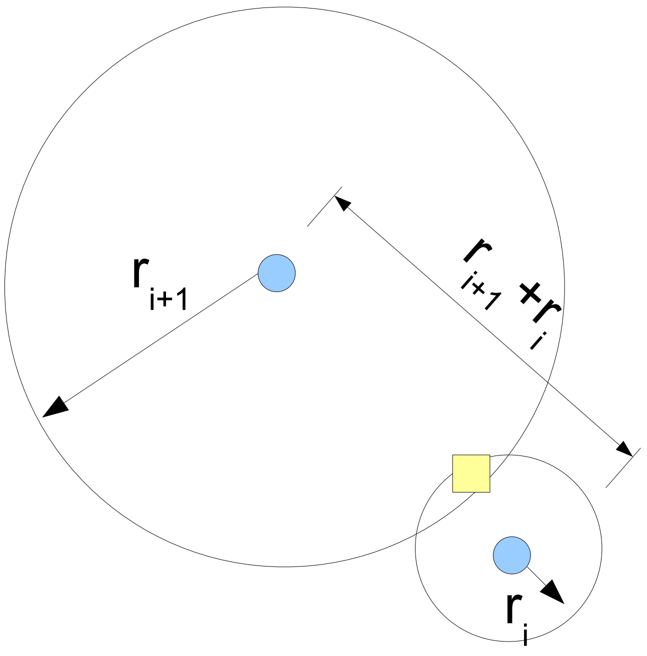


Fig. 3. The flooding radius is chosen in such a way that beacons at level i hear the floods of beacons at level $i+1$. In static networks, this is $r_i + r_{i+1}$.

Algorithm 1: clock-tick (t) (t: current time)

```
1 begin
2     refreshLevel = $\arg\max_j \{t \bmod \nu 2^j = 0\}$ /* select new beacons for levels 1,...,refreshLevel */
3     if myBeaconLevel $\leq$ refreshLevel then
          /* new beacon election at my level; higher-level beacons do not change */
4         delete routeTable and member entries for levels 1...refreshLevel
5         wait ($\pi_{self}$) /* wait for beacon messages from other nodes */
6         myBeaconLevel = highest level for which routeTable empty
7     beaconMsg = new msg (type=beacon, src=self, dst=flood, dist=0, level=myBeaconLevel)
8     send (beaconMsg)
9 end
```

Algorithm 2: receive-beacon (msg, from) (msg: flooded beacon message; from: sending node)

```
1  begin
2      if msg.id not seen before OR msg.dist< routeTable[msg.src].dist then
3          lowestLevel = $\lceil \log_2$ msg.dist$\rceil$ /* determine lowest level where this node belongs to the beacon
               sending this message                                                                   */
4          beaconLevel = msg.level /* level of the beacon = highest level where this node belongs to this beacon
               */
5          msg.dist++
6          if msg.dist < $f_{\texttt{beaconLevel}}$ then
7              send (msg) /* continue flooding the beaconing message                                   */
               /* within flooding radius: remember return path to beacon in routing table              */
8              for level = lowestLevel,...,beaconLevel do
9                  add (msg.src, msg.dist, level, from) to routeTable
10             if myBeaconLevel= beaconLevel-1 then
                   /* this beacon is from my parent: an $(i-1)$-beacon $u$ sends register msg to each $i$-beacon $v$ within
                      distance $f_i$ in every time slot, to maintain the reverse path $v \to u$              */
11                 msg = new msg(type=membership, src=self, dst=msg.src, dist=0, level=beaconLevel,
                   next-hop=from routeTable)
12                 send (msg)
13             if msg.dist < $r_i$ then
                   /* within membership radius: return membership msg in $i$-slot for every level in
                      [lowestLevel,...,beaconLevel] for which level$\leq i$                            */
14                 msg = new msg(type=membership, src=self, dst=beacon, dist=0,
                   level=[lowestLevel,min{beaconLevel, refreshLevel}], next-hop=from
                   routeTable)
15                 send (msg)
16 end
```

Algorithm 3: receive-membership (msg, from) (msg: membership message; from: sending node)

```
1  begin
2      if I am the target beacon (msg.dst = self) then
3          add msg.src  to member[msg.level] /* add source of registration msg to members for this level   */
4          if msg.level = myBeaconLevel-1 then
5              add (msg.src, msg.dist, msg.level, from) to routeTable
6      else
           /* msg not destined to me, forward towards msg.dst and remember route (this establishes forward routes from
              $i$-beacons to $(i-1)$-beacons                                                            */
7          add (msg.src, msg.dist, msg.level, from) to routeTable
8          msg.dist++
9          msg.nextHop= look up path to beacon msg.dst from routeTable
10         send (msg)
11 end
```

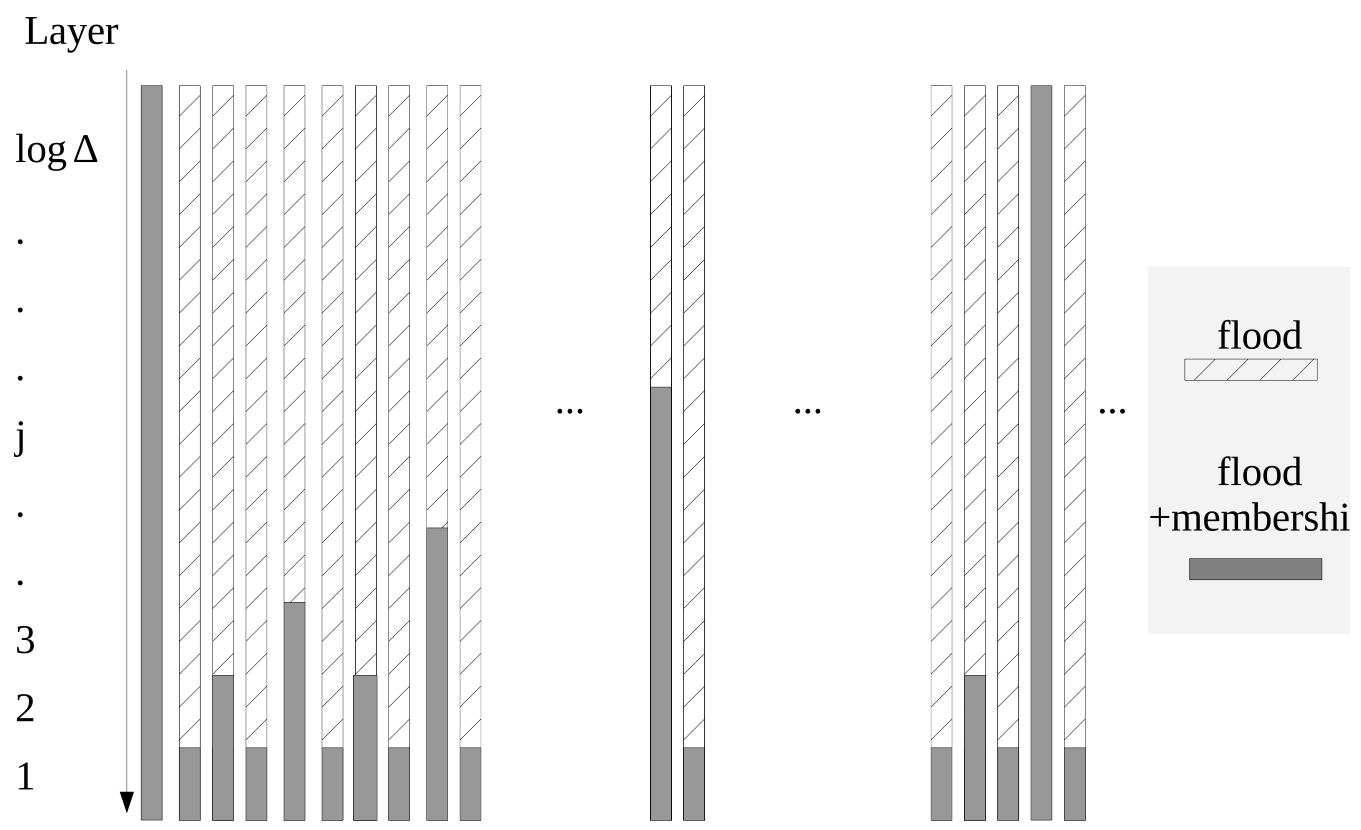
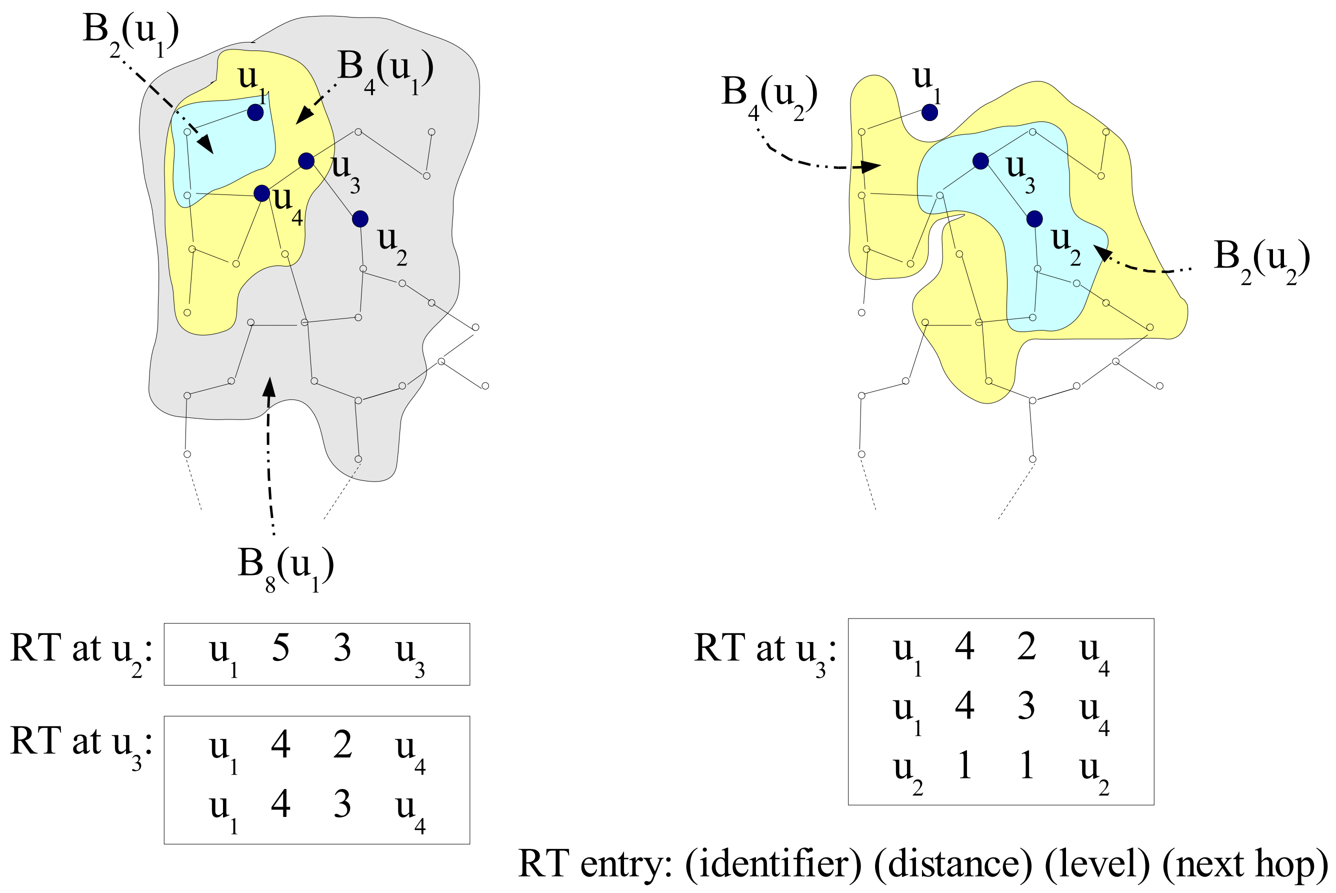


Fig. 4. The memberships up to level i are updated every $\nu 2^i$ time steps. At the levels above i, the beacons and their memberships do not change.

Algorithm 4: forward (`msg`) (`msg`: data packet)

```
1  begin
2      if msg.dst= self then
3          succeeded, deliver pkt
4      else
5          if msg.src= self then
               /* I am the source; execute upstream probing until positive response     */
6              probe all levels in increasing order $1, \ldots, \Delta$
7              send probe packets to parents at each level, wait for responses
8              forward msg to (one of) the positive responders
9          else
               /* I am an intermediate beacon in the downstream phase                   */
10             probe all level-(myBeaconLevel $-1$) beacons
11             send probe packets to each child beacon, wait for responses
12             forward msg to (one of) the positive responders
13 end
```



RT at u_2:

u_1	5	3	u_3

RT at u_3:

u_1	4	2	u_4
u_1	4	3	u_4

RT at u_3:

u_1	4	2	u_4
u_1	4	3	u_4
u_2	1	1	u_2

RT entry: (identifier) (distance) (level) (next hop)

Fig. 5. The example starts with empty routing tables. First, on the left, node u_1 floods at level 3. We focus on nodes u_2 and u_3. Node u_2 is within 8 hops from u_1 but further away than 4 hops. Consequently, it can only have an entry for node u_1 at level 3. At the same time, node u_3 can add an entry for node u_1 at the levels 2 and 3, since it is at distance 4 of u_1. Next, on the right, u_2's turn to flood comes (right after u_1's turn). This node is already covered at level 3. Consequently, it will flood at level 2. The node u_3 could potentially add an entry for this node at levels 1 and 2. However, it is already covered at level 2 and so adds only an entry for level 1. We do not show the entries beacons add for themselves.

C. Sketch of Analysis

The algorithm as described in Sections III-A and III-B applies to any sequence of κ-smooth connected graphs. We first informally argue that the algorithm maintains correct routes. The more formal proof of its correctness and calculation of the routing stretch and overhead is deferred to Section V, where the specific smooth and doubling structure of random geometric graphs is used to demonstrate efficiency in routing stretch and control overhead. Here, we briefly sketch this analysis, and highlight how the assumptions we make lead to low stretch and to low overhead.

Suppose that node u wants to communicate to node v. Recall that a node maintains a route to every beacon it has heard from in the last time step (see Algorithm 2), and also registers with all these beacons. The forwarding algorithm starts with the node u initiating a request to discover node v, sequentially to the beacons it knows of, according to its level (nearest first). Every beacon, upon receiving this request, checks with the beacons it knows of at the level immediately below itself. This succeeds because routes are maintained between any pair of two beacons that are in adjacent levels in the hierarchy and within the flooding radius of the higher of the two. Furthermore, we can show that the lazy membership collection process ensures that the downstream phase of the forwarding algorithm always succeeds, provided no node can "escape" beyond the flooding radius before the next refresh, i.e., no pair (u, v) can be at a distance $\leq r_i$ at refresh, and at $> f_i$ after less than 2^i slots.

We establish in the next section that the smoothness and doubling assumptions ensure bounded route stretch and low control overhead of roughly $O(\log^2 n)$ per time step and node. It is instructive to note that low route stretch can be established based only on the smoothness assumption. Essentially, the smoothness assumption allows us to

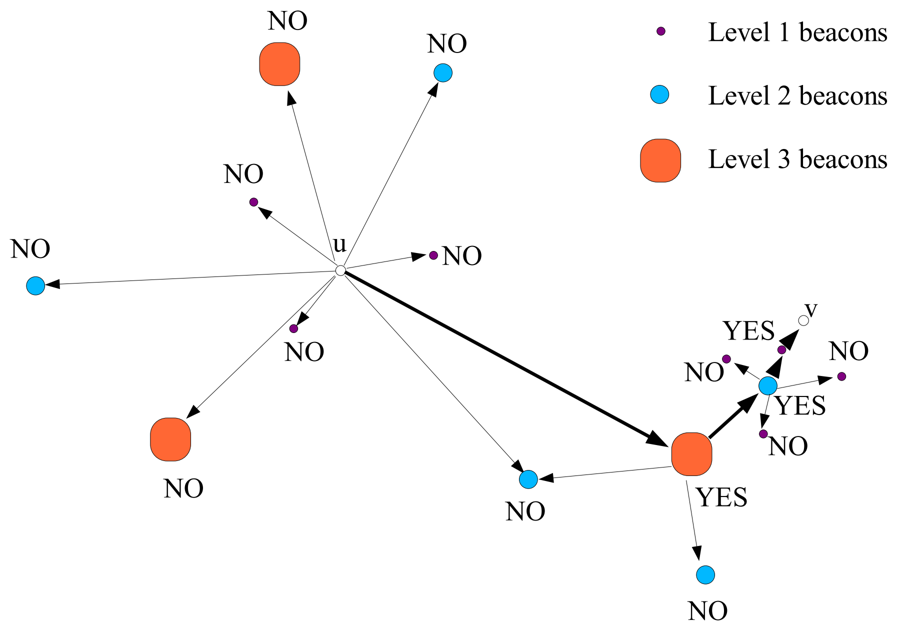


Fig. 6. Node u has a packet for node v. It searches in its routing table for all beacons it knows of at level 1 and sends them a probe packet containing v's identifier. None of the beacons at level 1 has an entry for this node and consequently they all answer negatively to node u. Next, node u repeats the same procedure with all the beacons it knows of at level 2. Again, all beacons answer negatively. On the third level, now, a beacon has an entry for node v. This beacon will probe all the beacons it knows of at level 2, while the other beacons at level three will answer negatively to u. At least one beacon at level 2 must have an entry for v. This beacon again probes all the beacons it knows of at level 1 among which one must have an entry for v itself. Meanwhile, the other beacons reply negatively as they do not have any entry for v.

bound the difference between distances between two nodes and distances between beacons that these two nodes belong two. The smoothness assumption is required because of the lazy refresh schedule: as beacon membership is in general based on an outdated topology, smoothness ensures that the path from a source to a destination through the hierarchy is not too far off the shortest path.

We need the additional assumption of a doubling connectivity graph to bound the control overhead to build and maintain the hierarchy. Specifically, without this assumption, we would not be able to make statements about the number of beacons a node may belong to; in a doubling topology, we can bound this number at every level of the hierarchy. This ensures that the beacon flooding done at every time step is not too costly, and that membership and probing messages are not too numerous.

Finally, we demonstrate below that the smoothness and doubling assumption is fairly mild, by establishing that uniform (random) node placement plus the USL mobility assumptions yield sequences of graphs that satisfy both. More specifically, we study generalizations of the random geometric graph, where we allow the network domain to have topological inhomogeneities such as holes and obstacles. These have proven to be major complications for geo-routing protocols. We show that this family of topologies is smooth and doubling, and therefore handled efficiently by our routing algorithm.

IV. Models of Dynamic Wireless Networks

In this section, we prove properties of the network models presented in Section II that are necessary to analyze the performance of our algorithm. We focus our attention on the geometric random graph $\mathcal{G}(n,\rho_n)$, but all the arguments can be extended to the SINR full connectivity model with TDMA scheduling, discussed in Section VII-B. We also generalize many of the results to variants of geometric random graphs, to include inhomogeneities, obstacles etc., in Section VI. For technical reasons, for $\mathcal{G}(n,\rho_n)$, we assume that the communication radius ρ_n is such that $\rho_n = \sqrt{(1+\epsilon)\log n} > \log^{1/2} n$, where $\epsilon > 0$.

For uniform speed-limited (USL) mobility models discussed in Section II-C, the node locations $\{x^{(t)}(u)\}$ have a distribution that is uniform over $[0,\sqrt{n}) \times [0,\sqrt{n})$ at each time. Therefore, we now discuss the property of a sequence of geometric random graphs, $\mathcal{G}^{(t)}(n, \rho_n)$ under USL mobility. We subdivide the network area on which the nodes live into small squares of side $\frac{\rho_n}{c}$, where c is a constant chosen such that nodes in neighboring squares are connected and that an integer number of squares fit into the network area.

We arbitrarily set $c = \sqrt{5}$. We number the small squares from 1 to $m = \frac{n}{(\rho_n/c)^2} = \frac{nc^2}{(1+\epsilon)\log n}$ and denote by E_i the event that small square i does not contain any node, in a sequence of length n^ρ time steps, for some constant ρ. In the next theorem, we show that when nodes move according to USL mobility model, all small squares are populated w.h.p.

Theorem 1: There exists a constant ρ such that for a network divided into small squares of side $\frac{\rho_n}{c}$ (with $\rho_n > \sqrt{\log n}$), at every time step in a sequence of length n^ρ, every small square contains at least one node w.h.p.

Proof: Consider a sequence of length $Z = n^\rho$. Denote by $E_i^{(j)}$ the event the small square i is empty at time j. Let $m = \frac{n}{\rho_n^2}$. We can compute:

$$\begin{aligned}
\mathcal{P}\left[\bigcup_{j=1}^{Z}\bigcup_{i=1}^{m} E_i^{(j)}\right] &\leq Z\sum_{i=1}^{m}\mathcal{P}\left[E_i^{(j)}\right] \\
&= Z\sum_{i=1}^{m}(1-\frac{1}{m})^n \\
&\leq Z\sum_{i=1}^{m} e^{-\frac{n}{m}} \\
&= Z\frac{nc^2}{(1+\epsilon)\log n}e^{-\frac{nc^2(1+\epsilon)\log n}{n}} \\
&\leq Z\frac{nc^2}{(1+\epsilon)\log n}\frac{1}{n^{(1+\epsilon)}} \\
&= Z\frac{c^2}{(1+\epsilon)n^\epsilon\log n} \\
&= o(n^{\rho-\epsilon}).
\end{aligned}$$

We can now choose ρ such that $\epsilon - \rho > 0$ and the result follows. ∎

It is immediate that a in single instance of the connectivity graph (*i.e.,* time step), every small square is populated w.h.p.

Corollary 1: With probability at least $(1 - O(n^{-\epsilon}))$, there is no empty small square in a sequence of length 1.

We are now ready to show that at every time step in a sequence of n^ρ connectivity graphs, the connectivity graph is doubling w.h.p. Since we have a USL mobility model, any graph $\mathcal{G}^{(t)}(n, \rho_n)$ is statistically identical to $\mathcal{G}(n, \rho_n)$.

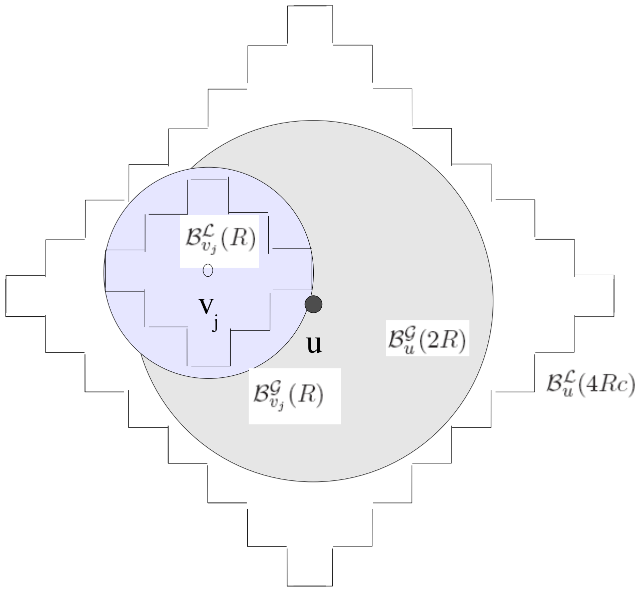
Theorem 2: $\mathcal{G}(n, \sqrt{(1+\epsilon)\log n})$ are doubling w.h.p.

Proof: By Lemma 1, all small squares contain at least one node w.h.p. Consequently, neighboring squares (vertically and horizontally) have at least one communication link. Denote by $\mathcal{L}(m, r)$ the grid having the small squares as vertices, and with edges between vertical and horizontal neighbors. Consider a ball $B_{upper} = \mathcal{B}_{2R}^{\mathcal{G}}(u)$ centered around some node u. Clearly, all nodes in B_{upper} must be contained in a square which is part of $\mathcal{B}_{4Rc}^{\mathcal{L}}(u)$, *i.e.,* $B_{upper} \subseteq \mathcal{B}_{4Rc}^{\mathcal{L}}(u)$. This follows from the fact that no node in B_{upper} can be further away from u than $2Rr$ in Euclidean distance, and that the grid is fully connected w.h.p. Similarly, one can see that $\mathcal{B}_{R}^{\mathcal{L}}(u) \subseteq B_{lower} = \mathcal{B}_{R}^{\mathcal{G}}(u)$. This is a consequence of the fact that $\mathcal{L}$ is a subgraph of $\mathcal{G}$, *i.e.,* two nodes in small squares R hops a part in $\mathcal{L}$ cannot be more than R hops apart in $\mathcal{G}$ (see Fig. 7). For an appropriately chosen constant α, we have:

$$B_{upper} \subseteq \mathcal{B}_{4Rc}^{\mathcal{L}}(u) \subseteq \bigcup_{j=1}^{\alpha}\mathcal{B}_{R}^{\mathcal{L}}(v_j) \subseteq \bigcup_{j=1}^{\alpha}\mathcal{B}_{R}^{\mathcal{G}}(v_j) \tag{1}$$

and $\mathcal{G}(n, \sqrt{(1+\epsilon)\log n})$ is *doubling*. ∎

Note that it is possible to build a deterministic geometric graph for which this property does not hold (for a counter-example, see appendix). Further, one can show that $\mathcal{G}(n, \rho_n)$ are *not* doubling w.h.p when $\rho_n < \sqrt{\log n}$. We prove this result in the appendix. At this point, we would like to emphasize that even though we analyze networks in which the nodes are uniformly distributed on a square area, the doubling property is a much more powerful tool. Indeed, our results and algorithms depend only on the doubling constant. Consequently, the algorithms and the bounds can be applied to any other type of network or node configuration which lead to a smooth and doubling connectivity graph. For instance, one can consider the network shown in Figure 2, described in Section II-B. It can easily be shown by using a technique similar to the one used in Theorem 2 that this network is doubling. While we can seamlessly apply our routing algorithm to such a network, a geographic routing algorithm would fail or require a high control traffic overhead to get out of dead-ends. This is because nodes would get stuck against the wall when routing packets from the lower to the upper part of the network. In the next subsection we prove a set of sufficient conditions for a wireless networks to have a constant doubling dimension.



$\mathcal{B}^{\mathcal{L}}_{v_j}(R)$
v_j
u
$\mathcal{B}^{\mathcal{G}}_{u}(2R)$
$\mathcal{B}^{\mathcal{G}}_{v_j}(R)$
$\mathcal{B}^{\mathcal{L}}_{u}(4Rc)$

Fig. 7. Proof of Theorem 2.

A. Sequences of Communication Graphs

In this subsection we study the behavior of a sequence of communication graphs *without* any obstacles. We show that a sequence of $\mathcal{G}^{(t)}(n, \rho_n)$ of length n^ρ, for some constant ρ, with the USL mobility model is κ-smooth. As already seen in Theorem 2, such a sequence of graphs is doubling at every time instant.

Theorem 3: A sequence of $\mathcal{G}^{(t)}(n, \rho_n)$ of length $\leq n^\rho$, where nodes move according to the USL mobility model with maximum constant speed S is w.h.p.

$$\max\left\{\frac{\rho_n d^{(t)}}{\frac{\rho_n d^{(t)}}{\sqrt{5}\sqrt{2}} - 2\sqrt{5}\sqrt{2}\tau S}, \sqrt{5}\sqrt{2}(1 + \frac{2\tau S\sqrt{5}\sqrt{2}}{\rho_n d^{(t)}})\right\} - \text{smooth}$$

Proof: Consider two nodes u and v at Euclidean distance $q_2^{(t)} = ||x_u - x_v||_2$ at time t. Let $q_1^{(t)} = ||x_u - x_v||_1 = \sum_{m=1}^{2} |x_m(u) - x_m(v)|$. Further, denote by $d^{(t)} = d^{(t)}(u, v)$ their shortest path distance at time t. One can see that $\frac{q_2^{(t)}}{\rho_n} \leq d^{(t)} \leq \frac{\sqrt{5}\sqrt{2}q_2^{(t)}}{\rho_n}$. Indeed, the shortest possible path will follow a straight line between u and v. The length of this line is $q_2^{(t)}$ and one hop can be of length at most ρ_n. In the worst case, the shortest path from u to v will follow the shortest path in the grid formed by the small squares of side $\frac{\rho_n}{c} = \frac{\rho_n}{\sqrt{5}}$, which exists w.h.p. Recall that we can only guarantee horizontal and vertical connectivity between small squares. The number of small squares in this path will be at most $\frac{\sqrt{5}q_1^{(t)}}{\rho_n}$. One can easily show that $q_1^{(t)} \leq \sqrt{2}q_2^{(t)}$. Let $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ sx_1 \end{pmatrix} = (x_u - x_v)$. We have

$$\begin{aligned} q_2^{(t)} &= \sqrt{x_1^2 + x_2^2} = x_1\sqrt{1+s^2} \\ &= (1+s)x_1\tfrac{\sqrt{1+s^2}}{1+s} = q_1^{(t)}\tfrac{\sqrt{1+s^2}}{1+s}. \end{aligned}$$

Since, we have $\frac{q_2^{(t)}}{q_1^{(t)}} = \frac{\sqrt{1+s^2}}{1+s}$, the term is maximized when $s = 1$. In Figure 8, we illustrate this point. Similarly, at time $t + \tau$, the shortest path distance will be bounded by $\frac{q_2^{(t+\tau)}}{\rho_n} \leq d^{(t+\tau)} \leq \frac{\sqrt{5}\sqrt{2}q_2^{(t+\tau)}}{\rho_n}$. However, we know that the Euclidean distance can change by at most $2\tau S$ in τ time steps[6]. Consequently,

$$\frac{q_2^{(t)} - 2\tau S}{\rho_n} \leq d^{(t+\tau)} \leq \frac{\sqrt{5}\sqrt{2}(q_2^{(t)} + 2\tau S)}{\rho_n} \tag{2}$$

[6]One can show that this remains true even if the nodes are reflected on the borders of the network.

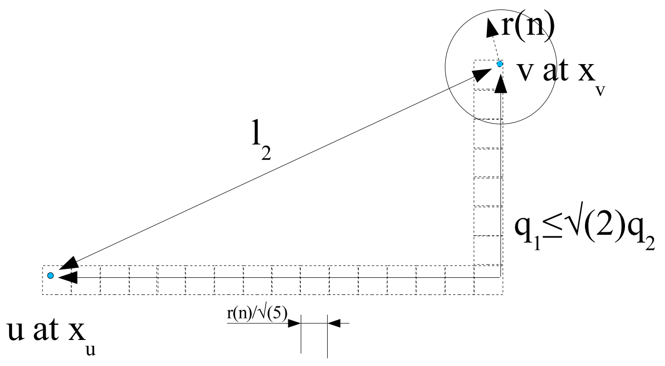


Fig. 8. Upper and lower bounds for the shortest path.

We can now bound the multiplicative stretch as follows: Hence,

$$\begin{aligned}&\max\left\{(\tfrac{1}{\sqrt{5}\sqrt{2}}-\tfrac{2\tau S}{\sqrt{5}\sqrt{2}q_2^{(t)}})^{-1},\sqrt{5}\sqrt{2}(1+\tfrac{2\tau S)}{q_2^{(t)}})\right\}\\&=\max\left\{\sqrt{5}\sqrt{2}\tfrac{q_2^{(t)}}{q_2^{(t)}-2\tau S},\sqrt{5}\sqrt{2}(1+\tfrac{2\tau S}{q_2^{(t)}})\right\}\\&=\max\left\{\frac{\rho_n d^{(t)}}{\frac{\rho_n d^{(t)}}{\sqrt{5}\sqrt{2}}-2\sqrt{5}\sqrt{2}\tau S},\sqrt{5}\sqrt{2}(1+\tfrac{2\tau S\sqrt{5}\sqrt{2}}{\rho_n d^{(t)}})\right\}=\kappa(\tau,d)\end{aligned}$$

■

The time it takes to multiply the shortest path distance between two nodes at distance d is proportional to d. Note that the larger the communication radius ρ_n, the smaller κ. Hence, the distance grows at most linearly with time. In particular, we have:

Corollary 2: There exist constants ν and κ defined in the proof such that a sequence of n^ρ connectivity graphs, under the USL mobility model with maximum constant speed S, is κ-smooth w.h.p.

Proof: By Theorem 3, we know that the sequence is

$$\max\left\{\frac{\rho_n d^{(t)}}{\frac{\rho_n d^{(t)}}{\sqrt{5}\sqrt{2}}-2\sqrt{5}\sqrt{2}\tau S},\sqrt{5}\sqrt{2}\left(1+\frac{2\tau S\sqrt{5}\sqrt{2}}{\rho_n d^{(t)}}\right)\right\}\text{-smooth}$$

w.h.p. Note that both terms decrease as a function of the communication radius ρ_n. Hence, we can set $\rho_n = 1$ without decreasing $\kappa(\tau, d)$. Similarly, both terms are decreasing in the distance d. We can therefore also set $d = 1$, which is the smallest possible distance in an unweighted graph. Consequently, if we set $\tau = \nu d = \nu$, we can now write

$$\kappa(\tau,d)\le\max\left\{\frac{1}{\frac{1}{\sqrt{5}\sqrt{2}}-2\sqrt{5}\sqrt{2}\nu S},\sqrt{5}\sqrt{2}(1+2\nu S\sqrt{5}\sqrt{2})\right\}$$

which is constant for ν constant. ■

V. Performance Analysis

In this section, we analyze the performance of our algorithm both in terms of control traffic and of route stretch. We do this for a sequence of smooth and doubling connectivity graphs, and will use $\mathcal{G}^{(t)}(n,\rho_n)$ with USL mobility for illustration. Note however that the results hold for any sequence of smooth and doubling graphs.

The bounds derived in this section hold w.h.p. uniformly for a sequence of length n^ρ of $(\log\alpha)$-doubling connectivity graphs. In particular, the results also apply to generalizations of the connectivity models given in Section VI.

In the sequel, α, κ and ν are the constants derived in Section IV. Let $\Delta = O(\sqrt(\frac{n}{\log(n)}))$ denote the diameter of the network. To bound the control traffic necessary for beaconing, we rely on the $(\log\alpha)$-doubling property of the metric space to show that a node can only hear a constant number of beacons at every level. We first show that a ball of radius $2R$ around any node u can only contain a constant number of balls (clusters) of radius R when we select the centers of these balls in an arbitrary order, and ensure that two centers cannot be closer than R. We later use this result to show that a node can hear at most a constant number of beacons at every level.

Theorem 4 (Random Cover): Let $\mathcal{B}_{2R}^X(u)$ be a ball of radius $2R$ centered at u in a graph metric (X, d) with doubling constant α. Then, there exist at most $k \leq \alpha^2$ nodes v_i, $(i = 1, 2, .., k)$ such that $\mathcal{B}_{2R}^X(u) \subseteq \bigcup_i^k \mathcal{B}_R^X(v_i)$ and $\min_{(i,j)} d(v_i, v_k) > R$.

Proof: By definition of a $(\log \alpha)$-doubling metric space, there must exist a cover of a ball of radius $2R$ consisting of at most α balls of radius R. Recursively, there must also exist an $\frac{R}{2}$-cover consisting of α^2 points. One can select at most one center v_i in each ball of radius $R/2$, as any other point inside this ball is within R of v_i. Hence, one can select at most α^2 such centers. ■

Corollary 3: Let B be a ball of radius $R > R'$ in an $(\log \alpha)$-doubling metric space (X, d). Then, one can select at most $k \leq (\frac{R}{R'})^{2\log(\alpha)}$ nodes v_i, $(i = 1, 2, .., k)$ such that $\mathcal{B}_R^X(u) \subseteq \bigcup_i^k \mathcal{B}_{R'}^X(v_i)$ and $\min_{(i,j)} d(v_i, v_j) > R'$. In particular, if $R = \eta R'$ for some constant R, then k is at most a constant $(\eta)^{2\log(\alpha)}$ independent of n.

Proof: Let $R = 2^i R'$. Hence, R' is doubled $\log \frac{R}{R'}$ times to obtain R. By Theorem 4, B can be covered by $\alpha^{2\log \frac{R}{R'}} = (\frac{R}{R'})^{2\log(\alpha)}$ balls of radius R'. ■

Here, one can think of the radius R of the large balls as the flooding radius, and of the radius R' of the small balls as the cover radius. Indeed, we use this result to show that a node u can hear the floods of all beacons within a given radius R. Moreover, this ball of radius R can contain at most $(R/R')^{2\log(\alpha)}$ beacons, since beacons must be at least R' apart.

A. Control Traffic

Theorem 5: The average control traffic overhead per time step for beaconing is at most $O(n \log^2 n)$ bits.

Proof: We analyze the control traffic at level i. Recall that a beacon at level i floods a distance $f_i = \kappa(2^{i+1} + 2^i)$ at every time step. During an i-slot, an i-beacon *cannot* be within $r_i = 2^i$ of another i-beacon (because the one later in π would not have been elected to level i). Level i is updated every $\nu 2^i$ time steps. Consider a node u. By Corollary 2, no nodes that are further away than κf_i hops at the time the memberships are updated at level i could move within f_i of u in less than $\nu 2^i$ time steps. However, that is before this level is updated again. Consequently, the number of beacons whose flood can reach u at any given time step is at most the number of i-beacons in a ball of radius κf_i at the time the membership is updated. In turn, node u will broadcast[7] the flood packets of at most that many beacons for this level i. By Corollary 3, this number is a constant[8] given by $(\frac{\kappa f_i}{2^i})^{2\log(\alpha)} = (3\kappa^2)^{2\log \alpha}$. Given that there are $O(\log n)$ levels, that there are n nodes and that a flood packet has size $O(\log n)$ bits, the average control traffic overhead per time step for beaconing is at most $O(n \log^2 n)$ bits. ■

We now compute the control traffic overhead necessary for nodes to update their memberships with beacons. Recall that level i and all levels below are updated every $\nu 2^i$ times steps and that a node can only be a member of one cluster at every level. Furthermore, a node only becomes a member of a cluster if it is within 2^i of the corresponding beacon.

Theorem 6 (Membership Update Overhead): The average control traffic overhead per time step to update memberships without load-balancing is at most

$$\frac{n \log \Delta \log n}{\nu} = O(n \log^2 n)$$

bits.

Proof: Consider a sequence of T time steps. The memberships will be updated up to level i every $\nu 2^i$ time steps, so $\frac{T}{\nu 2^i}$ times in a sequence of length T. At the time of the update, a node can be at distance at most 2^i from a beacon at level i. Consequently, the overhead in bits generated by a node in a sequence of T time steps is upper bounded by $\sum_{i=1}^{\log \Delta} \frac{T}{\nu 2^i} 2^i \log n = \frac{\log \Delta}{\nu} \log n$. ■

[7]Recall that when a node broadcasts a packet it is received by all direct neighbors in the connectivity graph. Consequently, there is one packet transmission per i-beacon.

[8]In the load-balanced scheme discussed in Section VII-A, this constant is $(5\kappa^2)^{2\log \alpha}$.

B. Route Stretch

In this section we will show that the route found with the forwarding algorithm is only a constant factor longer than the shortest path. Additionally we show that the probing overhead takes a negligible fraction of the cost to forward the message itself.

Theorem 7 (Routing Stretch): The worst case multiplicative routing stretch is $O(1)$.

Proof: Consider that we want to route from a node u to a node v, and that we had $2^k \le d(u,v) \le 2^{k+1}$, the last time level k was updated before the route search takes place. Let us denote by $b_i(v)$ the beacon to which node v had registered the last time level $i \le k$ was updated before the route search takes place. Clearly, we have $d(u, b_k(v)) \le \kappa(2^{k+1} + 2^k)$, and $d(b_i(v), b_{i-1}(v)) \le \kappa(2^i + 2^{i-1})$. This is true since the membership of node v at level i must have been updated at most $\nu 2^i$ time steps before the routing takes place, and that at the time level i gets updated, we have $d(b_i(v), b_{i-1}(v))) \le d(b_i(v), v) + d(v, b_{i-1}(v))$ by triangle inequality. Note that $d(b_i(v), b_{i-1}(v)) \le f_{i-1}$ and that $d(u, b_k(v)) \le f_k$. Hence, a route must exist between u and v and the length $r^{(t+\tau)}(u,v)$ of the route at time t is at most:

$$\begin{aligned} r^{(t)}(u,v) &\le \textstyle\sum_{i=1}^{k} f_i = \kappa \sum_{i=1}^{k} (2^{i+1} + 2^i) \\ &= 3\kappa \textstyle\sum_{i=1}^{k} 2^i = 3\kappa \frac{2^{k+1}-1}{2-1} \le 6\kappa d^{(t)}(u,v) \end{aligned}$$

In the worst case, nodes u and v have moved closer together (by a factor κ) while the beacons have moved further apart. Indeed, we have $d^{(t+\tau)}(u,v) \le \kappa d^{(t)}(u,v)$ for $\tau \le \nu 2^k$ as our network is κ-smooth. Note that if we waited longer that $\nu 2^k$, memberships would be updated again at level k and we could find another beacon at distance 2^k at most from v at level k. Hence, the worst case stretch is

$$\frac{r^{(t+\nu 2^k)}(u,v)}{d^{(t+\nu 2^k)}} \le 6\kappa^2 = O(1). \tag{3}$$

■

Every node can only hear floods from a constant number ($\mu = (3\kappa^2)^{2\log\alpha}$, see Theorem 5) of beacons at every level. Recall that the source will first probe all beacons at level 1, then all beacons at level 2, and so on. The procedure is repeated up to level k at which the source u will send a packet to $b_k(v)$. Note that the distance from u to this beacon can be at most $\kappa 2^{k+1} + 2^k = f_k$, and so it must hear its floods. In turn, when routing down the hierarchy, beacon $b_j(v)$ will probe at most a constant number ($(3\kappa^2)^{2\log\alpha}$) of beacons at level $j-1$. Finally, the distance between a node u and a beacon at level i can be at most f_i, and a probe packet will traverse at most $2f_i$ packets when a beacon at level i is probed (back and forth). This means that for discovery of the location of the destination, we need a probe overhead of at most $\mu 6\kappa d(u,v)$ packet transmissions. Therefore, this is a negligible part of the forwarding cost assuming the size of the message is large relative to $6\mu\kappa$.

VI. Performance in Inhomogeneous Topologies

In this section, we show that under certain conditions, the presence of topological holes (obstacles) in the network does not invalidate the doubling property, and only increases cost by a constant factor. In particular, we are interested in how we can alter the topology of a fully connected and dense network by removing nodes while still preserving the doubling property. In the second part of this section, we generalize this idea to arbitrary metric spaces. Consider a $\mathcal{G}(n, \rho_n)$ with $\rho_n > \sqrt{\log n}$, such that full connectivity is guaranteed. The network area is divided into squares of side $\frac{\rho_n}{c}$, where c is chosen such that nodes in horizontally and vertically adjacent squares are guaranteed to be within communication range. We now arbitrarily select squares and remove all nodes they contain. We call the resulting graph $\mathcal{G}_n$. Call $\mathcal{L}_n$ the full grid where the squares are vertices and $\mathcal{H}_n$ the corresponding grid in $\mathcal{G}_n$, *i.e.,* the thinned out grid obtained by selecting only non-empty squares in $\mathcal{G}_n$. In $\mathcal{L}_n$, we add an edge between horizontally and vertically adjacent squares (see Fig. 9). In $\mathcal{H}_n$, we first add an edge between horizontally and vertically squares containing at least one node. Then, for every pair of squares containing nodes that can communicate directly, we add an edge of weight corresponding to the distance between those two squares in $\mathcal{L}_n$. We add the edges from the shortest to the longest one, and only if no path of the same length already exists in $\mathcal{H}_n$. We can now define a topological hole as follows:

Definition 8 (Topological Hole): A set of vertically and horizontally adjacent empty squares in the graph $\mathcal{H}_n$ is called a *hole* if adding a (virtual) vertex in all of the squares in that set modifies the distance between at least two vertices in $\mathcal{H}_n$.

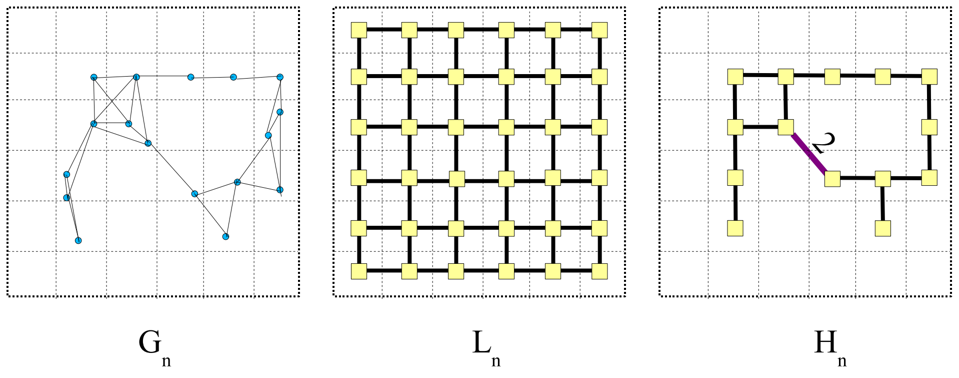


Fig. 9. Graphs $\mathcal{G}_n$, $\mathcal{L}_n$ and $\mathcal{H}_n$. The network area is divided into squares of side $\frac{\rho_n}{c}$ such that nodes in horizontally and vertically adjacent squares are guaranteed to be within communication range.

Let us denote by $\mathcal{V}_k$ the k^{th} hole ($k = 1, 2, 3, \ldots$). We define the perimeter $p(\mathcal{V}_k)$ of $\mathcal{V}_k$ as 2 times the maximum distance between any two vertices on the border of the hole, *i.e,* in squares adjacent to the empty squares defining the hole. Note that for all u, v, we have $d^{\mathcal{H}_n}(s_u, s_v) \geq d^{\mathcal{G}_n}(u, v)$.

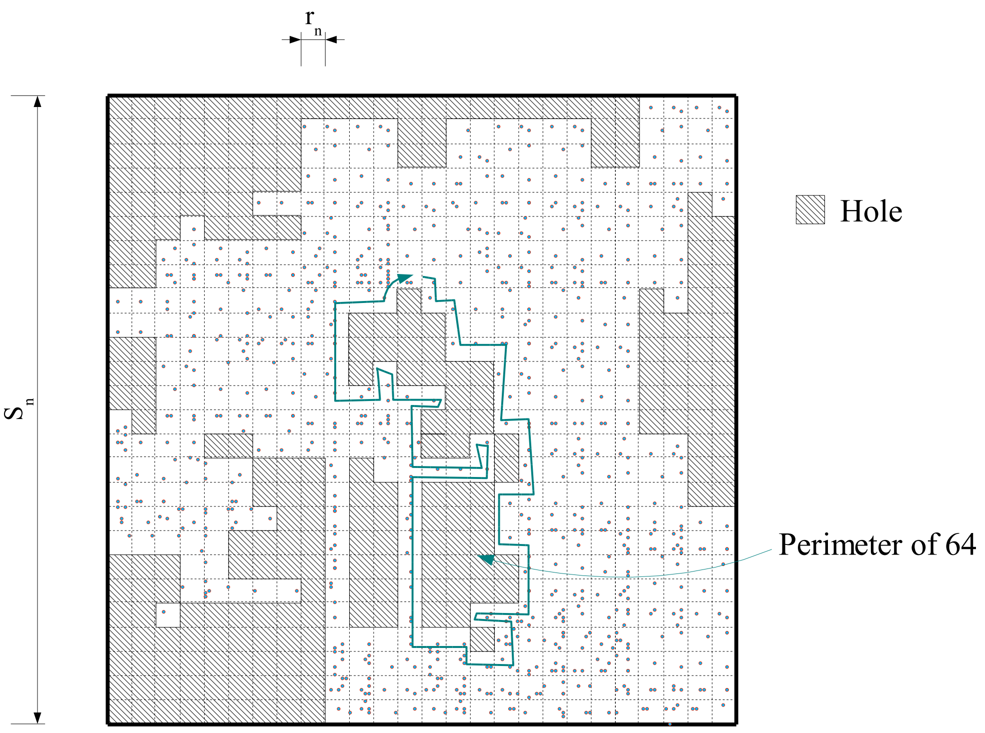


Fig. 10. A network with holes.

Theorem 8: Let $P = \max_k p(\mathcal{V}_k)$. Then, the doubling dimension is upper bounded by $O(P^2)$.

Proof: Consider a ball $\mathcal{B}_{2R}^{\mathcal{G}_n}(u)$ centered at u in $\mathcal{G}_n$. First, observe that $\mathcal{B}_{2R}^{\mathcal{G}_n}(u) \subseteq Box_{2Rc}^{\mathcal{L}_n}(u)$, where $Box_{2Rc}^{\mathcal{L}_n}(u)$ is the box centered at the square containing u in $\mathcal{L}_n$ which contains all nodes at "maximum norm" $2Rc$ (*i.e.,* l_∞-norm) from this square. In other words, all nodes within $2R$ hops from u in $\mathcal{G}_n$ must be in a square contained in this box. We will now cover this box with smaller boxes $Box_{\max\left\{1, \left\lceil \frac{R}{4\gamma} \right\}\right\rceil}^{\mathcal{L}_n}(s_{v_i})$. We need

$\left\lceil \frac{16R^2c^2 4\gamma^2}{R^2} \right\rceil = 64c^2\gamma^2$ such boxes at most. Consider the same small boxes in $\mathcal{H}_n$. Pick one non-empty square s_{v_i} in each such small boxes. Note that the maximum hop distances between two squares in such a small box in $\mathcal{L}_n$ is at most $\frac{R}{\gamma}$. For each of these hops, we might have to make a detour of at most P steps. Consequently, the same two squares could be at distance at most $\frac{PR}{\gamma}$ in $\mathcal{H}_n$. Observe now that for any two nodes u and w contained in squares s_u and s_w respectively, we have $d^{\mathcal{H}_n}(s_u, s_w) \geq d^{\mathcal{G}_n}(u, w)$. For each square s_{v_i}, we pick one node v_i contained in this square. Hence, for all nodes w contained in this small box, we have $d^{\mathcal{G}_n}(v_i, w) \leq d^{\mathcal{H}_n}(s_{v_i}, s_w) \leq \frac{PR}{\gamma}$. By setting $\gamma = P$, we obtain the claim. ■

We can extend this result to the case where the network can be divided into convex sets. We define a convex set in $\mathcal{G}_n$ with slack as follows:

Definition 9: Let Ψ be a set of nodes in $\mathcal{G}_n$. Let ${\mathcal{H}_n}^{(\Psi)}$ be the squares in $\mathcal{H}_n$ containing at least one node in Ψ. We say that the set Ψ is *convex* if $\forall u, v \in \Psi$, $d^{{\mathcal{H}_n}^{(\Psi)}}(s_u, s_v) = d^{\mathcal{L}_n}(s_u, s_v)$, where s_u and s_v are the squares containing u and v, *i.e.,* there must be at least one shortest path inside the convex set. We say that the set Ψ is *convex with slack P* if $d^{{\mathcal{H}_n}^{(\Psi)}}(s_u, s_v) \leq P d^{\mathcal{L}_n}(s_u, s_v)$.

We can now state the following theorem

Theorem 9: Let $\zeta_1, \zeta_2, ..., \zeta_q$ be a partition of the network into q convex sets with slack $P_1, P_2, ..., P_q$ respectively. Let $per_1, per_2, ..., per_q$ denote the perimeter of the convex sets. The doubling dimension is then upper bounded by

$$4 \max_{u,\rho} \sum_{\zeta_i : \zeta_i \cap Box_{\rho}^{\mathcal{L}_n}(s_u) \neq \emptyset} \left(\left\lceil \frac{per_i}{\frac{\rho}{P_i}} \right\rceil \right)^2 . \tag{4}$$

Proof: In the proof of Theorem 8, we have shown that any ball of radius $2R$ around some node u is contained in a box $Box_{\rho}^{\mathcal{L}_n}(s_u)$, where $\rho = 2Rc$. We can cover each convex set ζ_i intersecting this box with at most $4(\left\lceil \frac{per_i}{\frac{\rho}{P_i}} \right\rceil)^2$ small boxes of radius $R/4P_i$, as shown in Theorem 8[9]. A slack of P implies that by selecting one node in each of the small boxes, all nodes in the convex set are within R hops of this node in $\mathcal{G}_n$. If the box $Box_{\rho}^{\mathcal{L}_n}(s_u)$ is partitioned into several convex sets, selecting $4(\left\lceil \frac{per_i}{\frac{\rho}{P_i}} \right\rceil)^2$ nodes in each convex set ζ_i intersecting this box will in turn guarantee that all nodes are covered. ■

In practice, this result implies that if we are given a decomposition of the network into convex sets, we can bound the overall doubling dimension given the doubling dimension of each set separately. Further, this result implies that networks that consist of a small number of convex areas, which can each contain arbitrarily many small holes, have a low complexity in terms of doubling dimension. We will now relate the "shapes" of a topological hole to the doubling dimension. In particular, we will show that one can relate the doubling dimension to the maximum number of connected components in any square subarea.

Theorem 10: For any $\gamma \geq 2$, the doubling dimension $\log \alpha$ is such that

$$\alpha \leq 4\gamma^2 c^2 \max_{Box_{R/\gamma}^{\mathcal{L}_n}(u)} \left\{ \text{number of convex disconnected components with slack } \gamma \text{ in } Box_{R/\gamma}^{\mathcal{L}_n}(u) \right\}$$

Proof: In the proof of Theorem 8, we have shown that any ball of radius $2R$ around some node u is contained in a box $Box_{\rho}^{\mathcal{L}_n}(s_u)$, where $\rho = 2Rc$. In turn, we showed that by dividing this box into smaller boxes of side R/γ, and by selecting one node in each box, we could cover the larger ball of radius $2R$. Now, in each small box of side R/γ, the presence of holes might create several disconnected components. However, we know that inside each such component, we can cover any convex subset with slack γ with one nodes. The result follows. ■

This last result gives us a characterization of the alterations we can make to a fully connected $\mathcal{G}(n, \rho_n)$ network, while only affecting the doubling dimension by a constant factor. In particular, we can remove nodes as long as we do not create too many convex and disconnected components in any square subarea. Note that we can still remove arbitrarily many nodes as long as we only create small holes. Theorems 8, 9, and 10 imply that topologies such as the one shown in Fig. 11 have a constant doubling dimension. The results stated above are special cases of the more general result detailed in the sequel. Indeed, we can relate the doubling dimension in a metric space to the doubling dimension in another metric space if we know the distortion of the embedding that maps the points in one

[9]A convex area of perimeter q can always be included in a square area of side q.

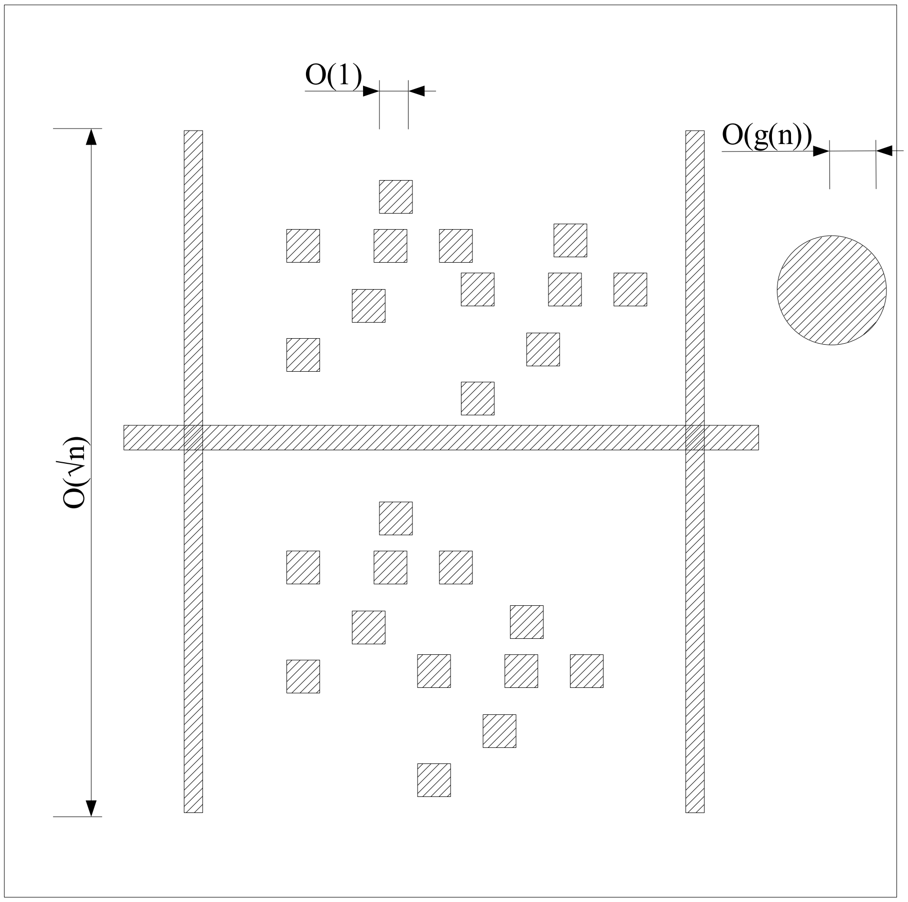


Fig. 11. A network with topological holes and a constant doubling dimension. The size of the large holes grow with n, but the network can be divided into a constant number of areas, each being convex with slack $O(1)$, *i.e.,* each of the convex areas contains only obstacles with a constant perimeter or that can only increase the distance between nodes by a constant factor. Note that even though the doubling dimension is low, greedy geographic forwarding of packets would fail as packets would get stuck in dead-ends against the holes. Squares containing no nodes are hatched.

metric space to the points in the other metric space. The example above is a special case of that setup where we map the nodes of a graph to points in Euclidean space. Consider two metric spaces (X, d) and (X', d'), where d and d' are distance functions which define a metric on the sets of point X and X'. We could for instance consider the two metric spaces $(X, ||.||)$ and $(\mathcal{H}, d(.,.))$, *i.e.,* the points in the plane with the Euclidean distance and the nodes in the graph with the shortest path distance. A *metric embedding* is a bijective function $\phi : X \to X'$ which associates to a point in one metric space a point in another metric space.

Definition 10 (Distortion of an Embedding): A mapping $\phi : X \to X'$ where (X, d) and (X, d') are metric spaces, is said to have distortion at most D, or to be a *D-embedding*, where $D \geq 1$, if there is a $K \in (0, \infty)$ such that $\forall x, y \in X$,

$$Kd(x, y) \leq d'(\phi(x), \phi(y)) \leq KDd(x, y).$$

If X' is a normed space, we typically require $K = 1$ or $K = \frac{1}{D}$. An embedding has distortion D with slack ϵ if all but an ϵ fraction of node pairs have distortion D under ϕ. Additionally, one can loosen this definition by allowing slack. The slack is said to be *uniform* if each node has distortion at most D to a $1 - \epsilon$ fraction of the other nodes. Finally, an embedding with distortion D and slack ϵ is *coarse* if for every node u the distortion is bounded to a node a distance greater than $r_\epsilon = \inf \left\{ r : |\mathcal{B}_r^X(u)| > \epsilon n \right\}$.

The doubling dimension of a metric space embedded into another metric space can be bounded as follows:

Theorem 11 (Bounding the Doubling Dimension): Consider a metric space $(\mathcal{H}, d)$ embedded in another metric space $(\mathcal{E}, d')$ by a function ϕ. Let the doubling dimension of $\mathcal{E}$ be β. Let the distortion of this embedding be D. Then, $\mathcal{H}$ has doubling dimension $\log \alpha$ with $\alpha = O((2D)^{\log \beta})$.

Proof: Choose any node $u \in \mathcal{H}$. If the above condition is fulfilled, the images of all nodes in $\mathcal{B}^{\mathcal{H}}_{2R}(u)$ can

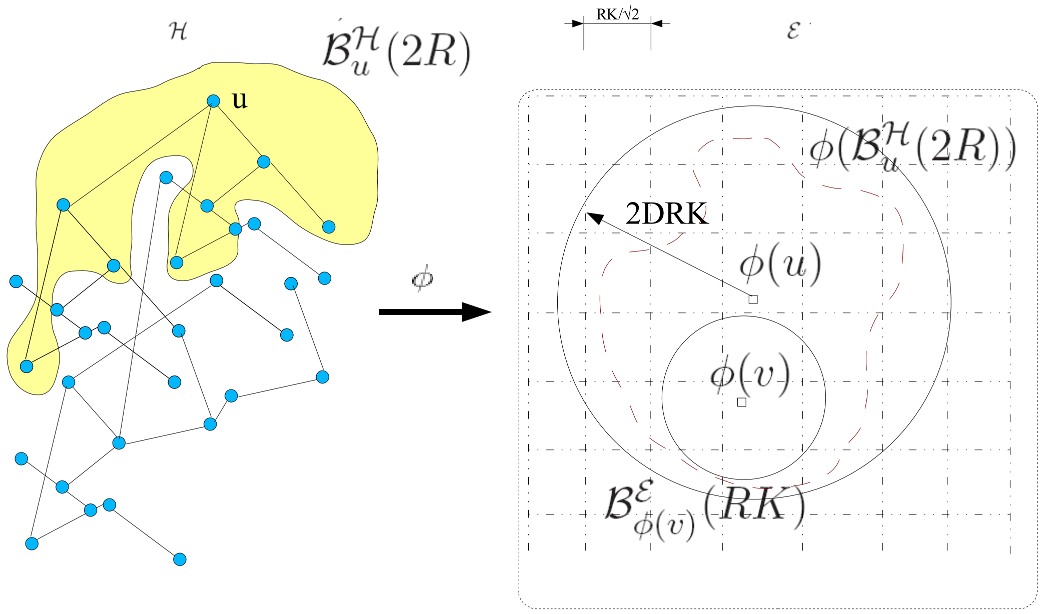

$\mathcal{H}$
$\mathcal{B}^{\mathcal{H}}_{u}(2R)$
u
ϕ
RK/√2
$\mathcal{E}$
$\phi(\mathcal{B}^{\mathcal{H}}_{u}(2R))$
2DRK
$\phi(u)$
$\phi(v)$
$\mathcal{B}^{\mathcal{E}}_{\phi(v)}(RK)$

Fig. 12. Proof of Theorem 11.

be at distance d' at most $2KDR$ from u at $\phi(u)$. Hence, $\phi(\mathcal{B}^{\mathcal{H}}_{2R}(u)) \subset \mathcal{B}^{\mathcal{E}}_{2KDR}(\phi(u))$. We will now try to cover $\phi(\mathcal{B}^{\mathcal{H}}_{2R}(u))$ by as few balls $\mathcal{B}^{\mathcal{E}}_{RK}(\phi(v))$ as possible (see Fig. 12, which illustrates this setup in the case when $\mathcal{H}$ is a graph and $\mathcal{E}$ the Euclidean space). To do so, let us cover $\mathcal{B}^{\mathcal{E}}_{2KDR}(\phi(u))$ by small balls of radius KR in $\mathcal{E}$. Covering $\mathcal{B}^{\mathcal{E}}_{2KDR}(\phi(u))$ will require at most $\beta^{\log 2D}$ balls of radius KR in $\mathcal{E}$, given that $\mathcal{E}$ has doubling dimension β. We know that $d(u, v) \leq d'(u, v)/K$, by Definition 10. Consequently, $\phi^{-1}(\mathcal{B}^{\mathcal{E}}_{RK}(\phi(v))) \subset \mathcal{B}^{\mathcal{H}}_{R}(v)$. We can conclude that $\mathcal{B}^{\mathcal{H}}_{2R}(u) \subset \bigcup_{j=1}^{\beta^{\log 2D}} \mathcal{B}^{\mathcal{H}}_{R}(v_j)$. ■

The presence of large obstacles in the network does not necessarily imply that the network is not doubling. In particular,

Theorem 12: Consider a metric space $\mathcal{E}$ with doubling dimension β. A metric space $\mathcal{H}$ that can be divided in k sets $S_1, S_2, ..., S_k$, such that each set embeds individually with distortion D_i into $\mathcal{E}$ has doubling dimension at most $\sum_{j=1}^{k} \beta^{2 \log 2D_j}$.

Proof: Consider any ball of radius $2R$ in $\mathcal{H}$, such that the nodes in the ball belong to at least two different sets (otherwise the theorem is clearly true). Note that the radius of each of these subsets can be at most $4R$. Consequently, we now that the part of the ball that belongs to S_i can be covered by at most $\beta^{2 \log 2D_i}$ (by applying Theorem 11 to cover a ball of radius $4R$ by balls of radius R). The theorem follows. ■

We can now broaden the class of communication networks that have low doubling dimension. In particular, if we can subdivide the communication graph into a constant number of subsets, such that each one embeds with constant distortion into the Euclidean plane, the whole network is doubling. Consequently, topologies such as the one shown in Figure 13 are doubling. In this example, we embed an unweighted graph into the Euclidean plane. Note that the minimal Euclidean distance between nodes should be $\rho\rho_n$ (for some constant ρ), such that $\rho\rho_n d(u, v) \leq ||x(u) - x(v)|| = O(1)\rho\rho_n$. If this equation is true for all pairs of nodes, then the distortion is $O(1)$. There is an issue when the nodes are neighbors in the communication graph, as the above rule implies that the Euclidean distance between such pairs of nodes should then be at least $O(\rho_n)$. However, we can ignore the distances below 2 as we will not cover balls of radius 1 (since we have a broadcast medium, the degree of a node does not impact the communication overhead). In such cases, it is obvious that geographic routing would fail, even though the inherent complexity of the network is low. Indeed, packets would get stuck against walls. Remarkably, our routing algorithm is oblivious to the topology and only depends on the doubling dimension. Hence, there is absolutely no need to detect or identify obstacles. The communication overhead will simply depend on the doubling dimension.

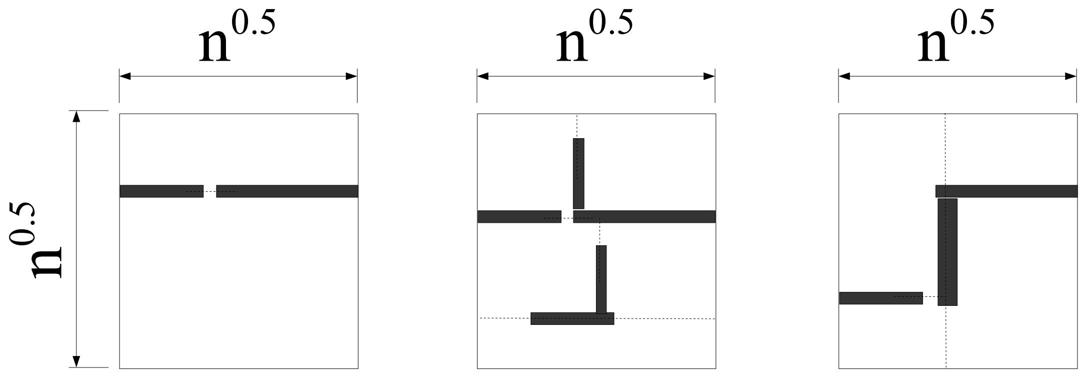


Fig. 13. A set of doubling network topologies. The network is dense, and made inhomogeneous by the walls, which do no allow transmissions to go through. Note that the walls stretch when n grows, such that the network wide distortion also grows with n. Dashed lines indicate the separation into sets.

VII. Extensions and Implementation Issues

A. *Load-Balancing the Routing Algorithm*

The approach above guarantees a low network-wide control traffic overhead. However, beacons at the highest levels might get temporarily overloaded by the membership packets of the nodes in their cluster when a membership update takes place. Over time, the overhead can be averaged out by randomizing the permutation π, but these nodes will be hot spots in the network for a short period of time. To work around this problem, memberships can be distributed in the cluster instead of stored at the beacon itself. First, we now set $f_i' = \kappa(2r_{i+1} + r_i)$. Additionally, whenever a beacon floods at level i, it includes its membership at level $i+1$ in the packet. This information is stored by all nodes that receive this flood packet. This will guarantee that all nodes that are members of a cluster at level i, know how to reach all beacons at level $i-1$ inside that cluster. A node that becomes a member of the cluster of beacon $b_i(u)$ at level i will now send its membership packet directly toward the beacon $\psi_{i-1}(u)$ at level $i-1$ inside this cluster with the identifier closest to u's. In turn, as soon as the packet reaches a node which is a member of $\psi_{i-1}(u)$'s cluster at level $i-1$, the membership packet is redirected toward the beacon $\psi_{i-2}(u)$ which is a member of $\psi_{i-1}(u)$'s cluster at level $i-1$ and has the identifier closest to node u's. The process is repeated until we reach a single node, which will store u's identifier on behalf of $b_i(u)$. Note that the membership can only be registered at a single location in the cluster reachable through a unique sequence of clusters. This remains true even when nodes move. Indeed, the nodes in the cluster of $b_i(u)$ will only forward the packet to beacons at level $i-1$ which were in the same cluster at the time the membership for this level got updated. Of course, whenever level $j < i$ is updated, we do now not only need to send u's identifier toward its new beacon at that level. Additionally, the node that holds u's identifier at level j might not be reachable anymore through a path of clusters with identifiers closest to u's. Consequently, this node will need to forward u's identifier toward the beacon at level j with the identifier closest to u's. Again the process will be repeated recursively until a single node is reached. As we will see, the cost of avoiding hot spots is a factor $\log n$ in the total control traffic. Finally and most importantly, with this procedure beacons no longer get overloaded. Rather, the traffic is be distributed in its cluster.

The data forwarding process remains the same except that the source node will not probe the beacon itself, but rather search for the node in the beacon's cluster that should hold the destination's identifier. If this node holds the identifier, it will then probe the beacons one level below in the same way. Recall that the nodes which potentially hold u's membership can be reached at any given instant in time through a unique sequence of clusters. The procedure is repeated until the destination is reached.

Finally, we show that the average control traffic overhead with load-balancing is increased by at most a factor $\log n$.

Theorem 13 (Membership Update Overhead): The average control traffic overhead per time step to update memberships with load-balancing is at most

$$\frac{n \log^2 \Delta \log n}{\nu} = O(n \log^3 n)$$

bits.

Proof: Consider a sequence of T time steps. The memberships will be updated up to level i every $\nu 2^i$ time steps, so $\frac{T}{\nu 2^i}$ times in a sequence of length T. At the time of the update, a node can be at distance at most 2^{i+1} from a beacon at level $i-1$ inside its cluster at level i. Similarly, a node can be at distance at most 2^i from a beacon at level $i-2$ inside its cluster at level $i-1$. In the load balanced scheme, we have to count the overhead to go down the hierarchy of beacons. For a beacon at level i, this is at most $2^i \times 2$. Consequently, the overhead in bits generated by a node in a sequence of T time steps is upper bounded by $4\sum_{i=1}^{\log \Delta} \frac{T}{\nu 2^i} 2^i \log n = 4\frac{\log \Delta}{\nu} \log n$. However, node u is a member of a cluster at all $\log \Delta$ levels. Recall that the node that holds u's identifier must always be reachable through a path by choosing the beacon (cluster) with the identifier closest to u's. Hence, whenever level i gets updated, all $\log \Delta$ nodes that hold u's identity must follow the same procedure as u itself. We conclude that the overhead is upper bounded by $(\log \Delta) 4\frac{\log \Delta}{\nu} \log n$ bits. ■

B. Performance under SINR Full Connectivity

Since the wireless channel is a shared medium, the transmissions between nodes interfere with each other. However, the signal strength decays as a function of the distance traveled, and therefore we can define the SINR for transmission from node u to v as,

$$\text{SINR} = \frac{P_n ||x(u) - x(v)||^{-\beta}}{N_0 + \sum_{w \neq u,v} P ||x(w) - x(v)||^{-\beta}}, \tag{5}$$

where β is a distance loss (decay) parameter depending on the propagation environment, P_n is the common transmit power of the nodes and N_0 is the noise power. We can of course easily adapt this to have power control for the nodes. A transmission is successful if the SINR is above some constant threshold value ς. For static nodes, just as in the case of geometric random graph, we assume that the node locations $\{x(u)\}$ are chosen independently and uniformly in $[0, \sqrt{n}) \times [0, \sqrt{n})$. This model for wireless networks has been extensively studied in the literature (see [GK00], [KV02]). The authors base their analysis of the capacity of wireless networks on a TDMA scheme for the SINR connectivity model of (5). We argue here that the structure of the resulting connectivity graph is identical to that of $\mathcal{G}(n, \rho_n)$, for $\rho_n > \sqrt{\log n}$. Therefore, the results we prove for $\mathcal{G}(n, \rho_n)$, would also be applicable to such graphs. In practice, it is a non-trivial task to design a distributed scheduling protocol (MAC layer protocol) that mimics the behavior of this TDMA scheduler. However, these MAC layer implementation issues are far beyond the scope of this document (see for instance [MSZ06]). We only make the argument here that the connectivity graph resulting from such a TDMA scheme would yield the same behavior as a $\mathcal{G}(n, \rho_n)$.

We will subdivide the network into small squares of side $s_n = \frac{\rho_n}{c}$. We need to show that if two nodes u and v are in neighboring small squares (and so have the guarantee that they can communicate under the $\mathcal{G}(n, \rho_n)$ model as we will see in the sequel), then there exists a TDMA scheme that allows them to communicate under the SINR connectivity model of (5). If this is the case, then we can apply the same proof techniques for both models. We let the maximum transmission power grow in the same way as we did for the $\mathcal{G}(n, \rho_n)$ model[10], *i.e., $P_n \leq (N_o \varsigma \rho_n)^\beta$.* Additionally, we want to design a TDMA scheme such that the capacity of all links is at least $O(\frac{1}{\log n})$ $[bits/sec]$. It can be shown (see [RS98]) that every small square contains at most $O(\log n)$ nodes. Hence, we ask that the traffic can flow at constant rate independent of n between neighboring small squares, and that each node is treated

[10]Note that the $\mathcal{G}(n, \rho_n)$ model corresponds to the SNIR model without interference. Indeed, if we remove interference, two nodes can communicate whenever $\frac{P_n ||x(u) - x(v)||^{-\beta}}{N_0} > \varsigma$ for some threshold value ς. Hence, two nodes can communicate whenever $||x(u) - x(v)|| < (\frac{P_n}{N_o \varsigma})^{1/\beta}$. In particular, we let $P_n = (N_o \varsigma \rho_n)^\beta$.

equally. Note that this requirement is very similar to the scheme proposed in [GK00] in which one node per small square can transmit at constant rate to any neighboring square[11].

Theorem 14: There exists a TDMA scheme such that all nodes can communicate with any node located in a neighboring small square at a rate of $O(\frac{1}{\log n})$ [bits/sec]. Hence, the aggregate traffic can flow between neighboring small squares at a constant rate independent of n.

Proof: We take a coordinate system, and label each square with two integer coordinates. Then we take an integer k, and consider the subset of squares whose two coordinates are a multiple of k (see Figure 14). By translation, we can construct k^2 disjoint equivalent subsets. This allows us to build the following TDMA scheme: we define k^2 time slots, during which all nodes from a particular subset are allowed to transmit for the same duration of $O(\frac{1}{\log n})$ seconds. Each small square contains at least one and at most $O(\log n)$ nodes w.h.p. (see [RS98] and the proof of Theorem 1). We assume also that at most one node per square transmits at the same time, and that they all transmit with the same power P_n. Let us consider one particular square. We suppose that the

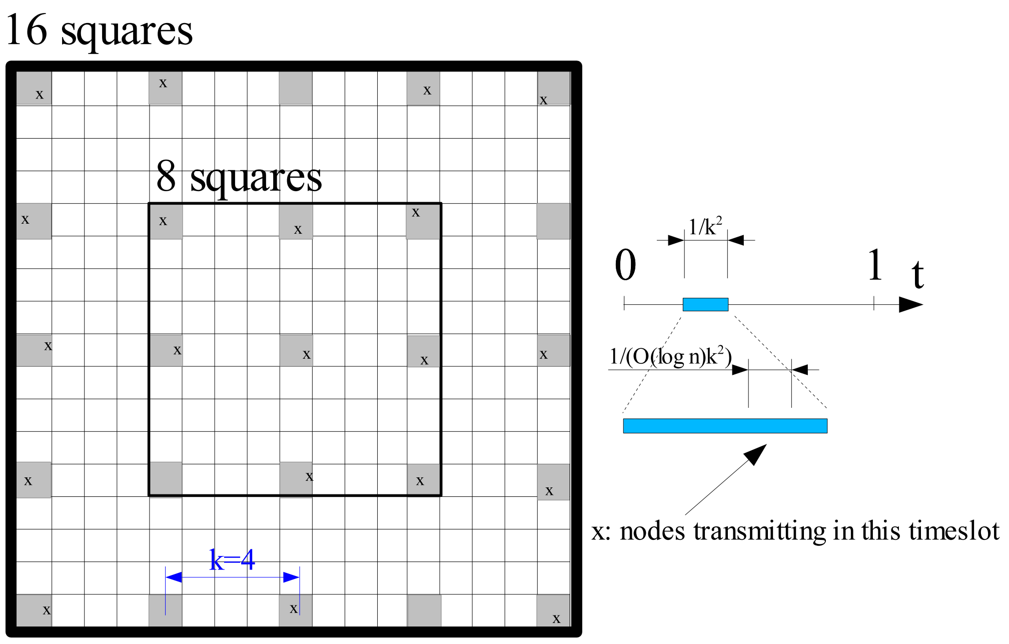


Fig. 14. Illustration of the TDMA scheduling scheme.

transmitter in this square transmits towards a destination located in a square at distance at most 1. We compute the signal-to-interference ratio at the receiver. First, we choose the number of time slots k^2 as follows: $k = 4$. To find an upper bound to the interference, we observe that with this choice, the transmitters in the 8 first closest squares are located at a distance at least 3 (in small squares) from the receiver (see left-hand side of Figure 14). This means that the Euclidean distance between the receiver and the 8 closest interferers is at least $2s_n$. The 16 next closest squares are at distance at least 7 (in small squares), and the Euclidean distance between the receiver and the 16 next interferers is therefore at least $6s_n$, and so on. The sum I of interference terms can be bounded as follows:

$$\begin{aligned} I &\leq \sum_{i=1}^{\infty} 8iP_n \left[2s_n(2i-1)\right]^{-\beta} \\ &= P_n \left[2s_n\right]^{-\beta} \sum_{i=1}^{\infty} 8i\left[(2i-1)\right]^{-\beta} \\ &= (N_o\varsigma\rho_n)^{\beta}\left[2\frac{\rho_n}{c}\right]^{-\beta} \sum_{i=1}^{\infty} 8i\left[(2i-1)\right]^{-\beta} \end{aligned}$$

This term clearly converges if $\beta > 2$. Now we want to bound from below the strength of the signal received from the transmitter. We observe first that the distance between the transmitter and the receiver is at most $\sqrt{2(s_n^2)} \leq 2s_n$.

[11]The throughput achieved by this scheme is $O(\frac{1}{\sqrt{n\log n}})$ $[bits/second/node]$ when n source destination pairs are chosen uniformly at random.

The strength S of the signal at the receiver can thus be bounded by

$$\begin{aligned} S &\geq P_n \min\left\{1, 2s_n^{-\beta}\right\} \\ &= O(1) \end{aligned}$$

Finally, we obtain the following bound on the SINR: $SINR \geq \frac{S}{N_o+I}$. As the above expression does not depend on n, the theorem is proven. ■

C. Implementation Issues

In Section V, we have computed worst case bounds whose constants may be conservative. In this section, we explore this aspect by looking at simulation results for the control traffic and stretch. Recall that we had computed that for each of the $O(\log n)$ levels, a node has to retransmit a packet of at most $(3\kappa^2)^{2\log\alpha}$ beacons. Even if we set the maximum speed as well as the parameter ν to 1, this is still $\sqrt{10}+20$, and consequently the constant in the bound on the overhead at least as high as $(3(\sqrt{10}+20)^2)^2 \approx 2.5 \cdot 10^6$! In Figure 15, however, we show that in practice this constant is approximately 30. This simulation was run with 50 up to 10000 nodes moving at a maximum speed of 1. One can observe that the experimental scaling behavior corresponds extremely well to the theoretical behavior. To stress this fact, we also plot $100\log n$ as a benchmark. Note that the overhead is expressed in number of packets rather than bits (a packet being of size $O(\log n)$).

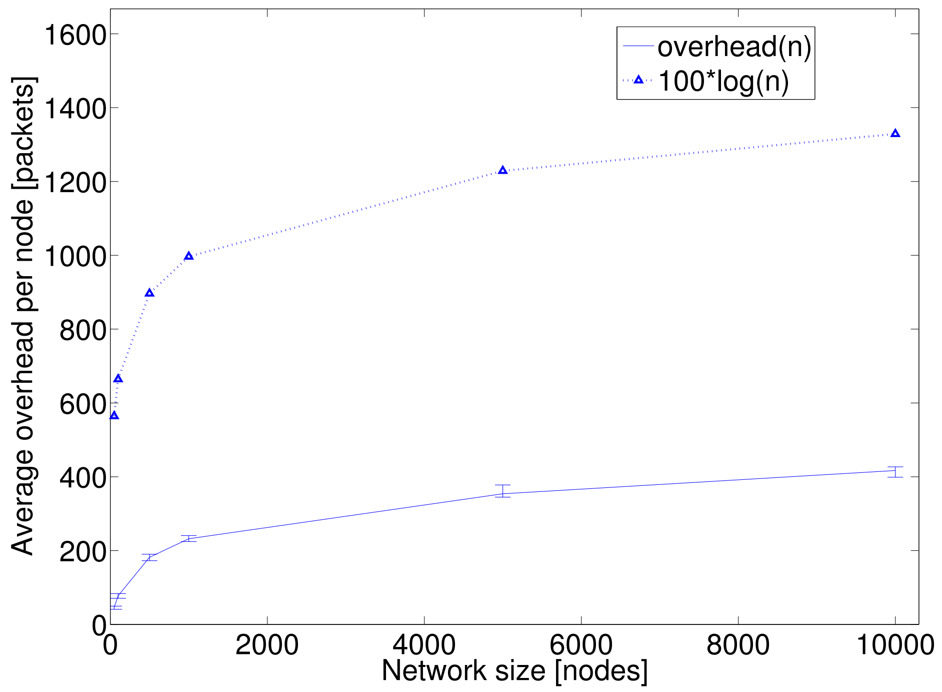


Fig. 15. Average control traffic overhead per node in packets as a function of the network size. Nodes move at a speed of maximum speed of 1. The confidence interval is given by the 95% and 5% percentiles. The size of a packet is $O(\log n)$ bits. We also plot $100\log n$ to show that our analytical predictions match the simulation results.

Similarly, in Figure 16 we show that for a network of 1000 nodes, the stretch is at most 1.5 for all node pairs. If we compute the maximum theoretical stretch, we can show that it is again considerably larger and hence a pessimistic bound. These small constants could make a practical implementation realistic.

We have made a certain number of assumptions in our models, which we now clarify.

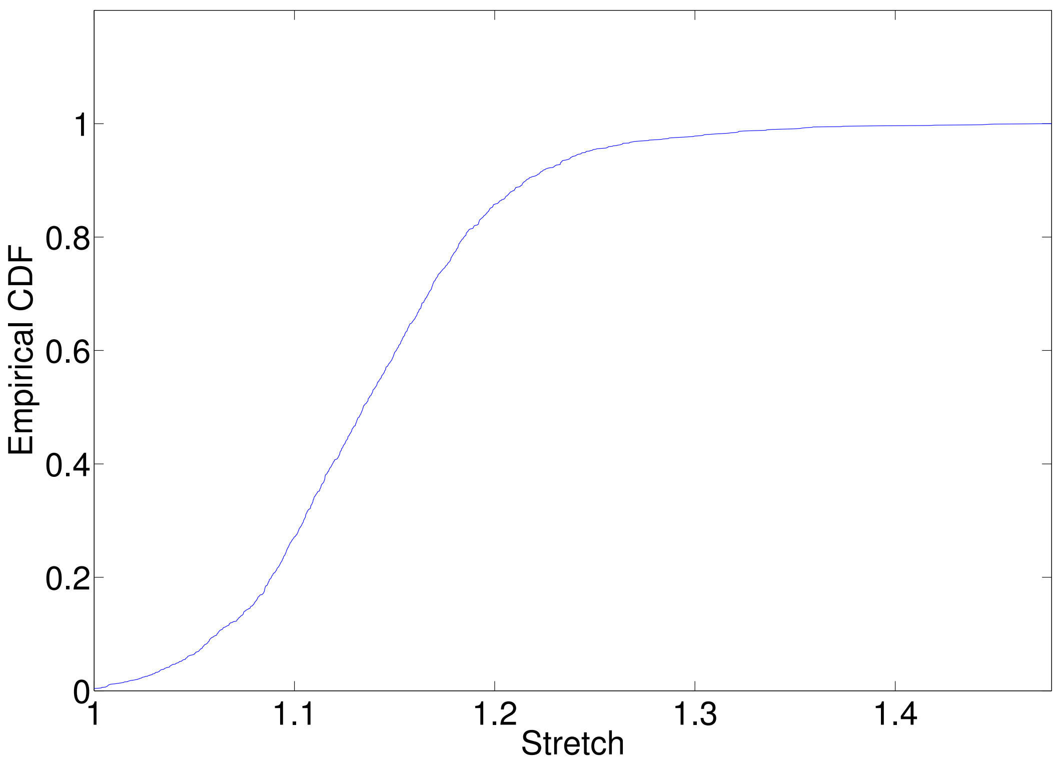


Fig. 16. Empirical cumulative distribution of stretch (route length/shortest path) for a network with 1000 nodes moving at a maximum speed of 1.

In practice, the random permutation π of the nodes, which determines the order in which the flooding occurs, causes an uneven average load on the nodes, because π_1 is always a highest-level beaconing, while the lower-order nodes are less likely to end up high in the hierarchy. This could be fixed by randomly drawing a new permutation π from time to time (or even to make π adaptive, e.g., to remaining battery capacity). This must occur only when the highest level of the hierarchy gets refreshed.

We have assumed that the network diameter Δ is known. In a practical implementation, where this parameter may not be known a-priori, Δ should be an upper bound of the real diameter. This is acceptable, as the control overhead only grows as the logarithm of Δ, and is therefore quite insensitive to overestimation.

Another important parameter is the time step ΔT. Our analysis is based on the time-scale separation assumption that the connectivity graph only changes between time steps, and that a message can be forwarded wirelessly from any node to any other node within ΔT. In practice, changes in network topology may happen at any time, and our protocol would restore full connectivity at the next time step after a change. As a consequence, ΔT should be set as a function of the rate of change due to mobility, and the specific requirements of the application. For example, in the example in Section II-A, ΔT might be set to a few $100ms$.

Finally, our performance results are derived for a time-slotted system, where the communication graph is assumed fixed within each time slot. In order to deal with a continuous-time system where the communication graph can change at any time instant, our model can be embedded in the following way. If the instantaneous speed of every node is limited to s, any two nodes that are within distance $\rho_n - 2s\Delta T$ at the beginning of a time step would remain within distance ρ_n within that time step. Therefore, to map the continuous-time system into our framework, we can conservatively ignore pairs of nodes for which this is not true, thereby ensuring that the graph "survives" the time slot. As ρ_n grows with n and $s\Delta T$ is a constant, and given the assumption of uniform node distribution in the USL model, we would thin out the graph by a vanishing factor. In this way, the performance of the continuous and the time-slotted systems are asymptotically the same.

In this work we assumed that the communication model is synchronous and error-free. There are several ways in which we can relax these assumptions and enhance the robustness and efficiency of our algorithms for real networks, where a synchronized clock and error-free channels may not be available. We outline possible approaches to this below, although a complete study of this is left to a future contribution. The possibilities include (i) We could maintain k separate instances of the routing structure and run some forward error correction code over these separate channels. While the control overhead would grow linearly with k, if routes are mostly disjoint, the probability of all routes failing at the same time would decrease exponentially. This would be a promising way to deal with low reliability of end-to-end routes within a single time interval. (ii) If a link failed before the interval is up, we could borrow from the extensive machinery developed in the context of on-demand and geographic mobile routing protocols, to locally repair this hole in the topology until the end of the interval. For example, we could run a constrained local flood to find a short path to route around a link that has died. While this would be difficult to incorporate into our theoretical analysis, we think this would be a very realistic way to harden the protocol in practice. (iii) Despite the above measures, we might still get occasional route failures in practice. Of course, end-to-end ARQ protocols would deal with these, as perfect end-to-end reliability is almost never achievable in reality, given the vagaries of wireless channels, clock drifts, topological holes, etc.

VIII. Conclusions

In this paper, we show that a large class of wireless network models belong to a larger class of networks, the *doubling* networks, in which efficient routing can be achieved. To design an efficient routing scheme, one can hierarchically decompose the network by relying on the doubling property to prove that the control traffic overhead and the stretch will remain low, even for dynamic doubling networks. This holds for a fairly broad class of uniform speed-limited (USL) mobility models. One advantage of the proposed routing algorithm is that it is robust, in that it works well in certain situations in which other existing algorithms cannot work well. This was illustrated in Section II-B for an example network with obstacles. We believe that many more such examples can be created where the use of the doubling rather than geographic properties would be crucial. To the best of our knowledge, our results are the first provable bounds for routing quality and costs for dynamic wireless networks. These techniques might give us insight into algorithm design for more sophisticated wireless network models.

Acknowledgments

The authors gratefully acknowledge many insightful comments from the anonymous reviewers, which helped to significantly improve this paper.

References

[AGGM06] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi, *Routing in networks with low doubling dimension*, ICDCS, 2006.
[Bet03] Christian Bettstetter, *The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks*, IEEE Trans. on Mobile Computing **2** (2003), no. 3.
[BV05] J.-Y. Le Boudec and M. Vojnovic, *Perfect simulation and stationarity of a class of mobility models*, INFOCOM, 2005.
[CGMZ05] T-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou, *On hierarchical routing in doubling metrics*, SODA, 2005.
[FRZ$^+$05] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee, David Culler, Scott Shenker, and Ion and Stoica, *Beacon-vector routing: Scalable point-to-point routing in wireless sensor networks*, NSDI, 2005.
[Gav01] C. Gavoille, *Routing in distributed networks*, ACM SIGACT News (2001), 36.
[GGC00] Pei Guangyu, M. Gerla, and Tsu-Wei Chen, *Fisheye state routing: a routing scheme for ad hoc wireless networks*, IEEE International Conference on Communications (ICC), 2000, pp. 70–74.
[GK98] P. Gupta and P. R. Kumar, *Critical power for asymptotic connectivity in wireless networks*, Stochastic Analysis, Control, Optimization and Applications (1998).
[GK00] P. Gupta and P. R. Kumar, *The capacity of wireless networks*, IEEE Transactions on Information Theory **46** (2000), no. 2, 388–404.
[HP01] Z. J. Haas and M R. Pearlman, *ZRP: a hybrid framework for routing in ad hoc networks*, Ad Hoc Networking, Addison-Wesley Longman, 2001, pp. 221–253.
[Jac06] P. Jacquet, *Control of mobile ad hoc networks*, IEEE Information Theory Workshop (ITW), 2006, pp. 97–101.
[JMB01] D. B. Johnson, D. A. Maltz, and J. Broch, *Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks*, Ad Hoc Networking (Addison-Wesley Longman Publishing Co., ed.), 2001, Book, p. 139172.
[KK00] Brad Karp and H. T. Kung, *Gpsr: Greedy perimeter stateless routing for wireless networks*, MOBICOM, 2000, p. 243.
[KRX06] Goran Konjevod, Andrea W. Richa, and Donglin Xia, *Optimal stretch name independent compact routing in doubling metrics*, PODC, 2006.
[KV02] S. R. Kulkarni and P. Viswanath, *A deterministic approach to throughput scaling in wireless networks*, ISIT, 2002, p. 351.

[LJDC$^+$00] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris, *Scalable location service for geographic ad hoc routing*, MOBICOM, 2000, p. 120.

[MSZ06] Eytan Modiano, Devavrat Shah, and Gil Zussman, *Maximizing throughput in wireless networks via gossip*, ACM SIGMETRIC'06, 2006, p. 26.

[PC12] C. Perkins and I. Chakeres, *Dynamic MANET On-demand (AODVv2) Routing (draft-ietf-manet-dymo-22)*, Internet Draft (work in progress) (2012).

[PR97] C. Perkins and E. M. Royer, *Ad-hoc on-demand distance vector routing*, MILCOM '97 panel on Ad Hoc Networks, 1997.

[RRP$^+$03] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, *Geographic routing without location information*, MOBICOM, 2003, p. 96.

[RS98] Martin Raab and Angelika Steger, *Balls into bins: A simple and tight analysis*, Lecture Notes in Computer Science **1518** (1998), 159.

[SMS06] G. Sharma, R. Mazumdar, and N. Shroff, *Delay and capacity trade-offs in mobile ad hoc networks: A global perspective*, INFOCOM, 2006, p. 1.

[Tal04] Kunal Talwar, *Bypassing the embedding: algorithms for low dimensional metrics*, STOC (Chicago, IL, USA), 2004, p. 281.

[TDGW07] D. Tschopp, S. Diggavi, M. Grossglauser, and J. Widmer, *Robust geo-routing on embeddings of dynamic wireless networks*, INFOCOM, 2007, p. 1730.

[YC05] J.Y. Yu and P.H.J. Chong, *A survey of clustering schemes for mobile ad hoc networks*, IEEE Communications Surveys & Tutorials **7** (2005), no. 1, 32–48.

Appendix

a) Unit Disk Graphs: Another common model used in studies on wireless networks are *Unit Disk Graphs* (UDG), which are the deterministic variants of the random geometric graphs. The randomness of the positions of the nodes is removed and they can be placed arbitrarily on a finite of infinite area. The channel model is completely deterministic as before and nodes are connected if their Euclidean distance is below a threshold distance r, called the communication radius. In mathematical terms, two nodes u and v with positions $x(u), x(v) \in [0, R]^2$ are connected if and only if $||x(u) - x(v)|| < r$. We will now show that there exist UDG which are not $\log \alpha$-doubling (see Section II for a definition of an $\log \alpha$-doubling metric).

Theorem 15: There exists an infinite UDG for which there is no constant that upper bounds the doubling dimension, *i.e.,* UDG are not doubling in general.

Proof: Consider the graph shown in Figure 17. To show that this graph is not $\log \alpha$-doubling, we must show that there exists no constant such that all balls of radius R can be covered a constant α number of balls of radius $R/2$, for all R. Consider the ball centered around u in the figure. One can see that there are $R/4 + 1$ "columns" which cross the middle row at a distance less than $R/2$ from u (that is, the intersection of the column and the row is less than $R/2$ hops away from u). The intersection of each of these columns with $B_u(R)$ is of length more than R (see hatched zones on Figure 17). Consequently, for each of these columns there is at least one node at distance more than $R/2$ from the middle row. To cover these nodes, we need to place at least one ball of radius $R/2$ on each of these columns. Hence, the doubling dimension is lower bounded by $R/4$ and tends to infinity as R goes to infinity. ■

One can notice that in the non-doubling UDG in the proof of Theorem 15 results from a careful construction. In the appendix, we show however that such a structure will occur with high probability when $\rho_n < \sqrt{\log n}$ in random geometric graphs.

b) Random Geometric Graphs with $\rho_n < \sqrt{\log n}$: We first consider the case in which the communication radius r is such that $\rho_n = r = (\log n)^{\frac{1}{2} - \frac{\theta}{2}} < \log^{1/2} n$ and $\theta \in \,]\,\zeta, 1\,]$. ζ is a constant such that $0 < \zeta < 1$.

Lemma 1: For any constant β, there exists constants $\gamma > 0$ and $b > 0$ such that a small square area of side γr with b nodes contains a subgraph of doubling dimension $\beta + 1$ with probability $p > 0$.

Proof: Consider the small square shown in Fig. 18 of side γr, where γ is a constant independent of n to be specified later. Subdivide the small square further into mini-squares of side r/c. Choose the constant c such that there exists a constant k satisfying $\sqrt{2}(k-2) > c \geq \sqrt{k^2+1}$. Under these conditions, two nodes in mini-squares separated by $(k-2)$ other mini-square will be connected, but not mini-squares $r(k-2)\sqrt{2}/c$ apart (see right hand side of Fig. 18). Consider now the graph on the left hand side of Fig. 18. Assume that each full (colored) mini-square contains exactly one node. We now focus on the ball $\mathcal{B}^{\mathcal{G}}_{2R}(u)$ and will lower bound the number of balls of radius R necessary to cover it. On the $\lfloor R/2 \rfloor$ first vertical branches from the left, the last node of the branch inside that ball (circled) must be covered by a ball of radius R centered on the same branch. This is clear since the length of the branch is larger than R. Consequently, the doubling dimension of this graph is at least $\lfloor R/2 \rfloor \geq \frac{R-1}{2}$. We want the doubling dimension to be larger than β, which can be easily achieved by choosing R

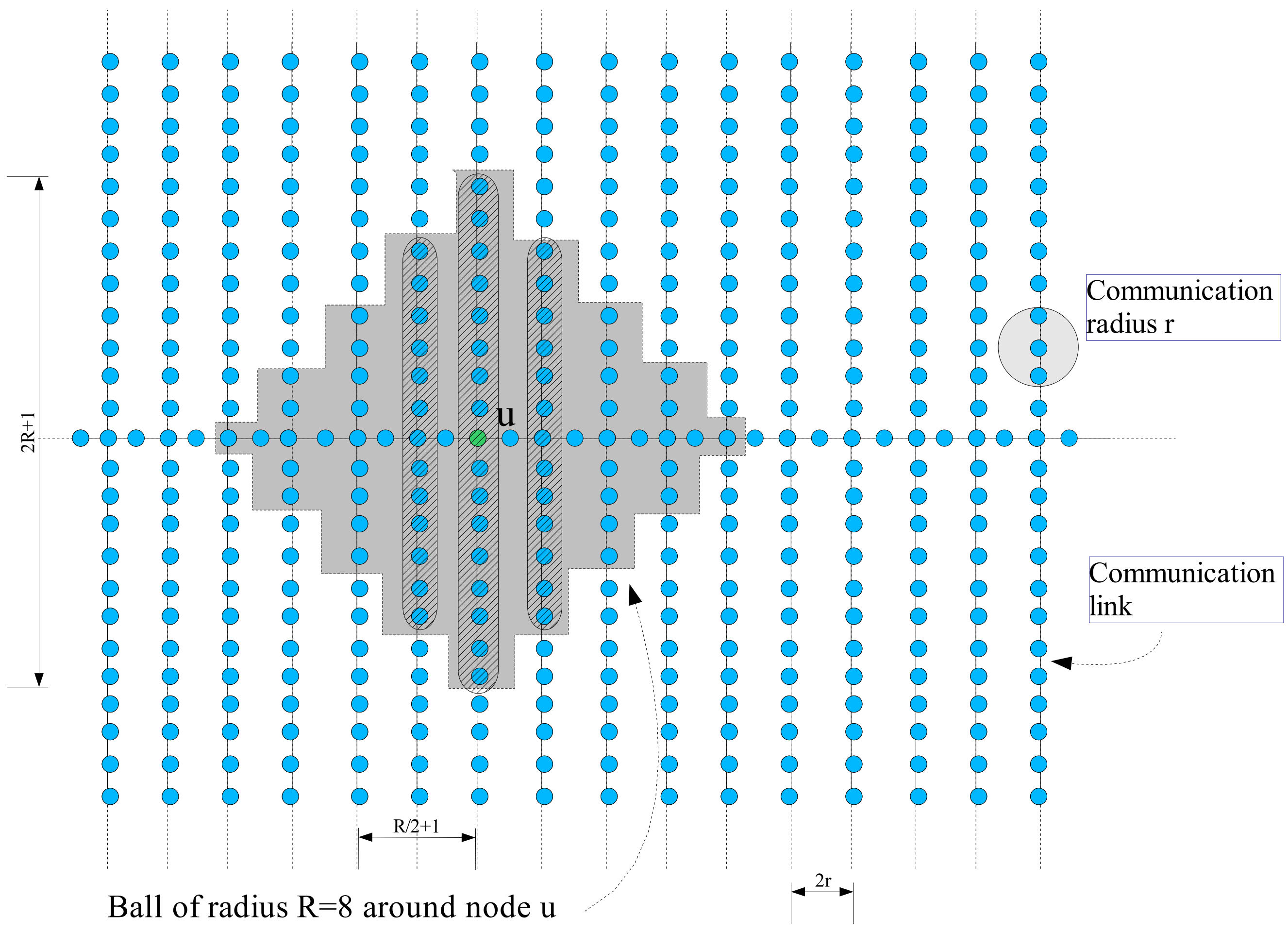


Fig. 17. An infinite UDG obtained by deleting all the nodes in every second column of a grid, except for the nodes on the the middle row. Consequently, "columns" are $2r$ apart.

such that $\frac{R-1}{2} > \beta$. Let $R = 2\beta+2 > 2\beta+1$. Further, we can now set $\gamma = (2R+5)(k-1)/c = (4\beta+9)(k-1)/c$ and $b = (2R+1)\lceil\frac{2R+1}{2}+1\rceil = (4\beta+5)\left\lceil\frac{4\beta+5}{2}+1\right\rceil$. This ensures that the doubling dimension is strictly larger than β.

It remains to be shown that when such a small square contains b nodes, the graph constructed above occurs with probability $p > 0$. The number m of mini-squares contained in a small square of side γr is $m = \frac{\gamma^2 r^2}{r^2/c^2} = \gamma^2 c^2$ which is constant. Each node can fall in any of the m squares with equal probability. Hence, all m^b configurations are equiprobable and $p = \frac{1}{m^b} > 0$. ■

We number the small squares from 1 to $m = \frac{n}{(\gamma r)^2} = \frac{n}{\gamma^2 \log^{1-\theta} n}$ and denote by X_i^b the indicator variable that takes value 1 when small square i contains exactly b nodes.

Lemma 2: There are at least $n^{1/2}$ squares containing b nodes with probability at least $(1 - O(\frac{1}{e^{n^{0.25}}}))$ for n sufficiently large

Proof:

$$\begin{aligned}
\mathcal{E}\left[X\right] &= \mathcal{E}\left[\sum_{i=1}^{m} X_i^b\right] \\
&= \sum_{i=1}^{m} \mathcal{P}\left[X_i^b\right] \\
&= \sum_{i=1}^{m} \binom{n}{b}(\tfrac{1}{m})^b(1-\tfrac{1}{m})^{n-b} \\
&\geq m(\tfrac{n}{b})^b(\tfrac{1}{m})^b(1-\tfrac{1}{m})^n \\
&\geq \tfrac{n}{b^b}(\gamma^2\log^{1-\theta} n)^{b-1}(1-\tfrac{1}{m})^{m\gamma^2\log^{1-\theta} n} \\
&\geq \tfrac{n}{b^b}(\gamma^2\log^{1-\theta} n)^{b-1}\frac{1}{2^{2\gamma^2\log_2^{1-\theta} n/\log_2^{1-\theta} e}} \\
&\geq O(n^{1-O(\frac{1}{\log^{\theta} n})}) \\
&\geq O(n^{\delta})
\end{aligned}$$

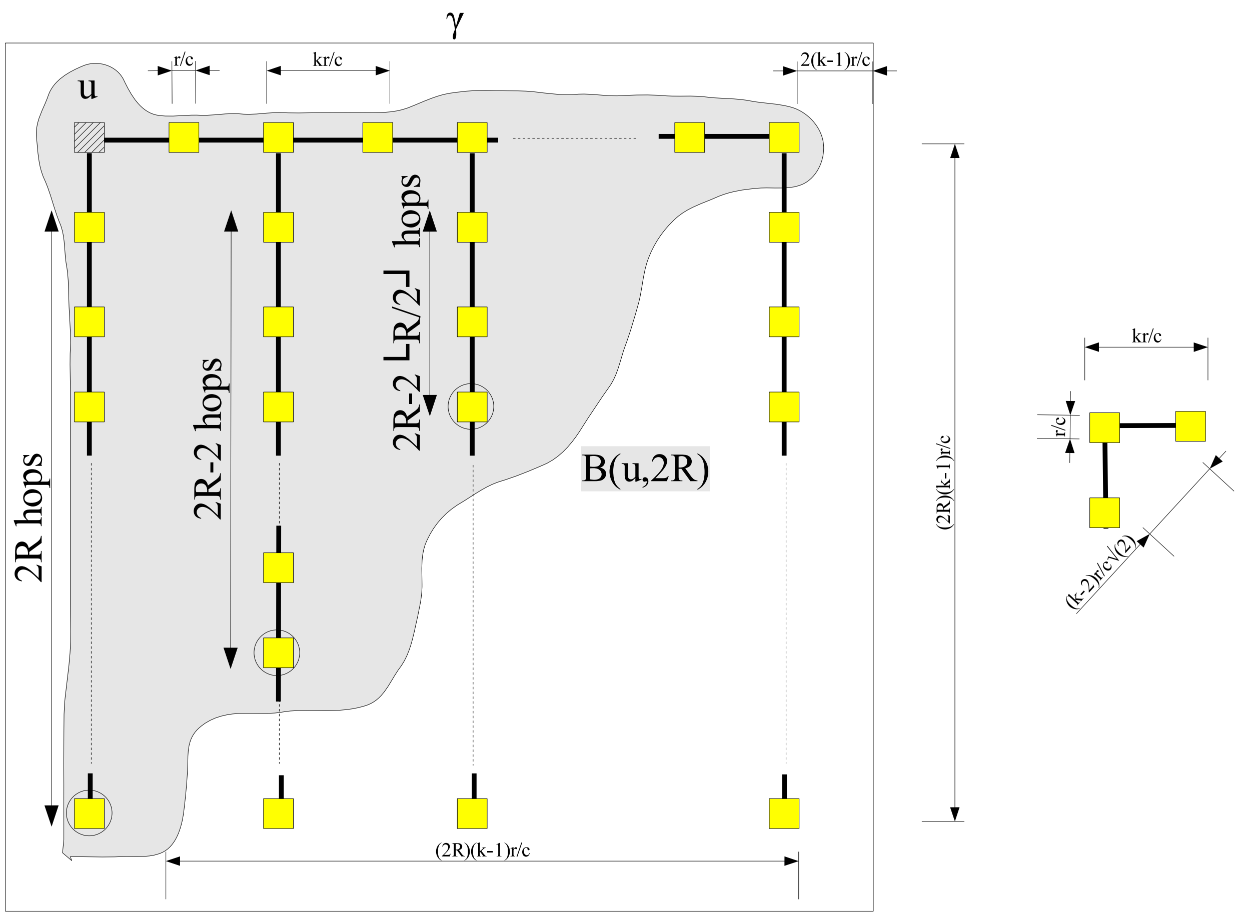


Fig. 18. Graph for the proof of Lemma 1.

where $\delta \geq \frac{7}{8}$ for n sufficiently large, since $\theta > \zeta$.

Let S_i be the random variable representing the small square into which the i^{th} node falls. Let F be the number of small squares containing exactly b nodes after all nodes have been placed. Then the sequence $Z_i = \mathcal{E}\left[F|S_1,...,S_i\right]$ is a Doob Martingale. One can show that $F = f(S_1, S_2, ..., S_n)$ satisfies the Lipschitz condition with bound 1. Indeed, changing the placement of the i^{th} ball can only modify the value of F by at most 1. We therefore obtain:

$$\mathcal{P}\left[|F - \mathcal{E}\left[F\right]| \geq n^{5/8}\right] \leq 2e^{-2n^{10/8-1}} = 2\frac{1}{e^{2n^{1/4}}}$$

by the Azuma-Hoeffding inequality. Consequently,

$$\begin{aligned}\mathcal{P}\left[F < n^{1/2}\right] &< \mathcal{P}\left[F < \underbrace{\mathcal{E}\left[F\right] - n^{5/8}}_{=n^{7/8}-n^{5/8}>n^{1/2}}\right] \\ &\leq 2\frac{1}{e^{2n^{1/4}}} \leq 2\frac{1}{e^{n^{1/4}}}\end{aligned}$$

and

$$\mathcal{P}\left[F \geq n^{1/2}\right] \geq (1 - 2\frac{1}{e^{n^{1/4}}})$$

■

It now remains to show that in this regime, $\mathcal{G}(n, r)$ are not doubling with high probability.

Theorem 16: $\mathcal{G}(n, (\log n)^{\frac{1}{2}-\frac{\theta}{2}})$, where $\theta \in \,]\,\zeta, 1\,[$ and ζ is a constant such that $0 < \zeta < 1$, are not *doubling* with high probability.

Proof: By Lemma 1, for any constant β, a small square area of side γr with b nodes contains a graph of doubling dimension $> \beta$ with probability $p > 0$. By Lemma 2, there are $n^{1/2}$ such small squares containing b nodes w.h.p. Let F denote the number of small squares containing exactly b nodes. Consequently, the probability that at least one of this squares contains a graph of doubling dimension $> \beta$ is given by:

$$\begin{aligned}
\mathcal{P}\left[\text{not doubling}\right] &= \textstyle\sum_{j=1}^{m} \mathcal{P}\left[\text{not doubling}|F=j\right]\mathcal{P}\left[F=j\right] \\
&\geq (1 - O(\tfrac{1}{e^{n^{0.25}}}))\textstyle\sum_{j=n^{1/2}}^{m}(1-(1-p)^{j}) \\
&\geq (1-(1-p)^{n^{1/2}})(1-O(\tfrac{1}{e^{n^{0.25}}})) \\
&\geq (1-(1-p)^{n^{1/2}})^{2} \\
&\geq (1-\tfrac{2}{x^{O(n)}})
\end{aligned}$$

where $x = (\frac{1}{1-p}) > 1$. Consequently, with probability at least $(1 - \frac{2}{x^{O(n)}})$, there exists no constant which bounds the doubling dimension of $\mathcal{G}(n, (\log n)^{\frac{1}{2}-\frac{\theta}{2}})$. ■