Hierarchical Routing over Dynamic Wireless Networks

Dominique Tschopp
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
dominique.tschopp@epfl.ch

Suhas Diggavi
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
suhas.diggavi@epfl.ch

Matthias Grossglauser
Internet Laboratory
Nokia Research Center
00180 Helsinki, Finland
matthias.grossglauser@nokia.com

ABSTRACT

In dynamic networks the topology evolves and routes are maintained by frequent updates, consuming throughput available for data transmission. We ask whether there exist low-overhead schemes for these networks, that produce routes that are within a small constant factor (stretch) of the optimal route-length. This is studied by using the underlying geometric properties of the connectivity graph in wireless networks. For a class of models for wireless network that fulfill some mild conditions on the connectivity and on mobility over the time of interest, we can design distributed routing algorithm that maintain the routes over a changing topology. This scheme needs only node identities and integrates location service along with routing, therefore accounting for the complete overhead. We analyze the worst-case (conservative) overhead and route-quality (stretch) performance of this algorithm for the aforementioned class of models. Our algorithm allows constant stretch routing with a network wide control traffic overhead of $O(n \log^2 n)$ bits per mobility time step (time-scale of topology change) translating to $O(\log^2 n)$ overhead per node (with high probability for wireless networks with such mobility model). We can reduce the maximum overhead per node by using a load-balancing technique at the cost of a slightly higher average overhead. Numerics show that these bounds are quite conservative.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing Protocol; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms, Design, Theory

Keywords

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

SIGMETRICS'08, June 2–6, 2008, Annapolis, Maryland, USA.

1. INTRODUCTION

Designing distributed routing algorithms that consume a minimal amount of network resources is a major challenge in wireless ad hoc networks. In dynamic networks, the topology can change over time, and routing tables must be updated frequently. Such updates incur control traffic, which consumes bandwidth and power. It is natural to ask whether there exist low-overhead schemes for dynamic wireless networks that could produce and maintain efficient routes. We consider dynamically changing connectivity graphs that arise in wireless networks. Our performance metric for the algorithms is the average signaling overhead incurred over a long time-scale when the topology changes continuously. We design a routing algorithm which can cope with such variations in topology. We maintain routes from any source to any destination node, for each instantiation[1] of the connectivity graph. We want to guarantee that the route is within a (small) constant factor, called *stretch* of the shortest path length. In order to route to a destination, we need only the identity of the destination and not any kind of address or geographical position. Therefore, in the wireless routing terminology, we have included the "location service" in the control signaling requirement, and therefore hope to characterize the complete overhead needed to maintain efficient routes.

In order to develop and analyze the routing algorithms we utilize the underlying geometric properties of the connectivity graphs which arise in wireless networks. This geometric property is captured by the *doubling dimension* of the connectivity graph. A graph induces a metric space by considering the shortest path distance between nodes as the metric distance. The doubling dimension of a metric space is the number of balls of radius R needed to cover a ball of radius $2R$. For example a Euclidean space has a low doubling dimension as will be illustrated in Section 2. A metric space having a low (constant independent of the cardinality of the metric space) doubling dimension is called "doubling". We show that several wireless network graphs (under conditions given in Section 2) are doubling and therefore enable the design and analysis of hierarchical routing strategies. In particular, it is not necessary to have uniformly distributed nodes with geometric connectivity for the doubling property to hold, as illustrated in Figure 2 in Section 2. Therefore, the doubling property has the potential to enable us to design and analyze algorithms for a general class of wireless

[1]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.

networks. Moreover, for a large class of mobility models, the sequence of graphs arising due to topology changes are all doubling (for specific wireless network models). Since there are only "local" connectivity changes due to mobility, there is a smooth transition between these doubling graphs. We can utilize the locality of topology changes to develop lazy updates methods to reduce signaling overhead.

We show that several important wireless network models produce connectivity graphs that are doubling. In particular, we show that the geometric random graph[2] 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; some examples of networks with obstacles and non-homogeneous node distribution. We define a sequence of wireless connectivity graphs to be *smooth* if each of the graphs is doubling and the shortest path distance between two nodes in the graph changes smoothly (defined in Section 2).

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 while having an average network-wide traffic overhead of $O(n \log^2 n)$ bits per mobility time step. The load-balanced algorithm would require $O(\log^3 n)$ bits per node, per mobility time. Through numerics we show that the theoretically obtained worst-case constants are conservative.

1.1 Related Work

Routing in wireless networks has been a rich area of inquiry. The two main paradigms for routing have been geographic routing and topology based routing. Geographic routing algorithms (see for instance [10] and references therein) exploits the geometry of wireless networks, and base routing decisions on the Euclidean coordinates of nodes. Their performance depends on how well the Euclidean coordinate system captures the connectivity graph, and they can therefore fail in the presence of node or channel inhomogeneity (like in Figure 2 in Section 2). An often overlooked issue is the overhead incured because the 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 [13]) 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 (*e.g.,* 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 in the overhead of the routing algorithm itself.

[2]To the best of our knowledge, it is the first such observation for geometric random graphs

Topology based routing schemes (see [14] and [9]) compute routes based directly on the communication graph. To reduce overhead, most of these schemes establish routes *on demand* through a reactively, rather than continuously maintaining a route between every pair of nodes. 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., [14], this cached state resides in the intermediate nodes that are part of a route, whereas in source-routing approaches (e.g., [9]), 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 networks algorithms are the works presented in [15] and the beacon vector routing (BVR) introduced in [4]. 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. In [19], 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 [5]). 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 [11], and references therein). In particular [17] showed that one could get constant stretch routing with small routing table sizes for doubling metric spaces, when we use labeled routing. This result was improved to make routing table sizes smaller in [3]. The problem of named routing over graphs with small doubling dimension has been studied in [11] and [1], and references therein. 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 sent around in the network in order to build routing tables. An illustration is the computation of the shortest path between two nodes u and v in a graph. While it is sufficient for every node on the path between these two nodes to have one entry for v (of roughly $\log n$ bits *i.e.,* the name of the next hop), computing that shortest path requires a breadth first search of the communication graph and leads to a control traffic overhead of $O(n \log n)$ bits.

To the best of our knowledge, there has been no prior work on *dynamic* graphs over doubling metric spaces and on

theoretical analysis of control traffic overhead. In particular, we do not know of any other formal results to which ours are directly comparable.

2. MODELS AND DEFINITIONS

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 and 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 2.2[3]. 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. 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 will now define a *ball* of radius R around a node u. It 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 $\mathcal{X}$ is the set $\left\{v \in \mathcal{X} | d^{(t)}(u, v) \leq R\right\}$.*

In order to bound the control traffic overhead, we will recursively subdivide the connectivity graph into balls. It will be crucial for us to bound the number of balls of radius R necessary to cover a ball of radius $2R$ around some node u. In other words, we want to find the smallest number 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[4].

DEFINITION 2. *The doubling dimension of a metric space $\mathcal{X}$ is 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., *$\forall u \in X \; \exists \; S_u \subseteq \mathcal{X}, |S_u| \leq \alpha$ 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 $\mathcal{X}$ is* doubling *if its doubling dimension is a constant independent of the cardinality of $\mathcal{X}$.*

[3]Note that we do not require unit disc connectivity *i.e.,* two nodes need not be connected if and only if they are within a given fixed distance. We show in subsection 3.1 that our approach is also applicable to inhomogeneous network topology where nearby nodes could potentially be disconnected

[4]This concept was originally introduced in [6]. How definition differs slightly from the original definition which allowed arbitrarily small balls

A good way to illustrate the concept of doubling dimension is to look at the metric space defined by a set of points $\mathcal{X}$ in $\mathbb{R}^2$ with the Euclidean distance. A ball of radius $2R$ around a point x will simply be a disc of radius $2R$ around this point. To cover this disc, we will select a set of points such that all the surface is covered by the corresponding set of discs of radius R. Note that the number of discs required is independent of R and $|\mathcal{X}|$. Hence, this metric space is *doubling* (see Figure 1). In Section 2.1, we describe the geometric random

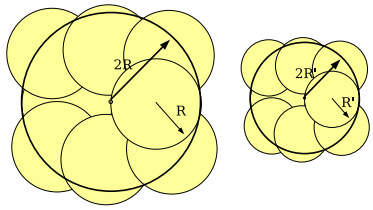

Figure 1: The metric space defined by a set of points in $\mathbb{R}^2$ and the Euclidean distance is doubling. Indeed, we can cover a disc of radius $2R$ by a constant (8 in this case) number of discs of radius R, whatever the value of R.

graph model, which will be the canonical model we will use to illustrate the ideas of the paper. In Section 2.2, we will develop the model where connectivity is determined by the SINR, and in Section 3.1 we will focus on inhomogeneous topologies. We give the requirements for the mobility model to result in a *smooth* sequence of wireless network graphs in Section 2.3. We state the underlying assumptions and give a table of notations in Section 2.4.

2.1 Geometric random graph

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

DEFINITION 4. *A random geometric graph $\mathcal{G}(n, r_n)$ has an unweighted edge between nodes u and v if and only if $||x(u) - x(v)|| < r_n$, where $\{x(u)\}$ are chosen independently and uniformly in $[0, \sqrt{n}) \times [0, \sqrt{n})$.*

In this paper we will be interested in fully connected geometric random graphs, and therefore focus on the case $r_n > \sqrt{\log n}$ [7]. As a natural extension, we can also define a sequence of random graphs $\mathcal{G}^{(t)}(n, r_n)$ with an unweighted edges between u and v at time t if $||x^{(t)}(u) - x^{(t)}(v)|| < r_n$. Whether each graph in the sequence $\mathcal{G}^{(t)}(n, r_n)$ 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 2.3.

In Figure 2, we illustrate a non-homogeneous random network where connectivity is not completely geometric as in Definition 4. This example is revisited in Section 3.1, where we show that though this connectivity graph is more complicated than $\mathcal{G}(n, r_n)$, it is still doubling, and therefore the algorithms developed in this paper are applicable. This illustrates the advantage of our approach to network modeling.

2.2 SINR full connectivity

Since the wireless channel is a shared medium, the transmissions between nodes nterfere with each other. However,

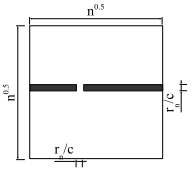

Figure 2: n nodes are distributed uniformly at random on a square area of side $\sqrt{n}$. A wall of width r_n/c is added which only has a small hole in the middle. Again, we assume $r_n > \sqrt{\log n}$. Nodes cannot communicate through the wall. Finally, we remove the nodes below the wall, which leads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

the signal strength decays as a function of the distance travelled, and therefore we can define the SINR for transmission from node u to v as,

$$\text{SINR} = \frac{P||x(u) - x(v)||^{-\beta}}{N_0 + \sum_{w \neq u,v} P||x(w) - x(v)||^{-\beta}}, \qquad (1)$$

where β is a distance loss (decay) parameter depending on the propagation environment, P 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. 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 $[0, \sqrt{n}) \times [0, \sqrt{n})$. For such a random network, we can design a fully connected wireless network TDMA scheduling scheme [8, 12] for the SINR connectivity model of (1). The structure of such a resulting connectivity graph is identical to that of $\mathcal{G}(n, r_n)$, for $r_n > \sqrt{\log n}$. Therefore, the results we prove for $\mathcal{G}(n, r_n)$, would also be applicable to such graphs.

2.3 Uniform speed-limited (USL) mobility

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, 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 satisfy 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 (on the torus) 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). The system has to be in the stationary regime. (iii) The generalized random direction models from [16], 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.

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 will be modified. Hence, we will work with a sequence of connectivity graphs. In order to design a routing algorithm with a low control traffic overhead, we will need to understand how fast the 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 by which this distance can change in κ 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.,

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

, $\forall u, v$.

Additionally, we simply say that the network is κ-*smooth* if there exists a constant ν such that $\kappa(\nu d, d) \leq \kappa(\nu) = \kappa$ independently of d. In this case, the distances grow at the same speed at all scales. In the sequel, we will bound κ and ν for our model. This USL property holds for a general class of *random trip* mobility models studied in [2], where it is shown that the stationary distribution of such mobility models is uniform and ergodic. We restate this theorem without proof.

Theorem 1. *([2]) The random-trip mobility model has uniform stationary distribution on $[0, a) \times [0, a)$.*

2.4 Assumptions

We consider that a time step ΔT is much larger than the round trip time (RTT) through the network *i.e.* the time scale for mobility is much larger than the time scale for communications. In order to simplify the analysis, we will make the assumption that nodes can communicate instantaneously through the network. We also make the assumption that there is a random permutation π on the nodes, and nodes know their rank in the permutation. In Section 6 we will then 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 $(1 - O(\frac{1}{n^\rho}))$, for some constant $\rho > 0$. In table 1, 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
r_n	Wireless communication radius
$\mathcal{G}(n,r_n)$	Random geometric graph
$\mathcal{B}_R^{(t)}(u)$	Ball of radius R around u
$\kappa(\tau,d)$	$max\left\{\frac{d^{(t)}(u,v)}{d^{(t+\tau)}(u,v)}, \frac{d^{(t+\tau)}(u,v)}{d^{(t)}(u,v)}\right\} \leq \kappa(\tau,d)$

Table 1: Table of notations

3. NETWORK PROPERTIES

We now prove some properties of the network models presented in Section 2, which are necessary to analyze our algorithm. We will focus our attention on the geometric random graph $\mathcal{G}(n,r_n)$, but all the arguments can be extended to other models (Sections 2.2 and 3.1). In particular, for $\mathcal{G}(n,r_n)$, we will now consider the case in which the communication radius r_n is such that $r_n = \sqrt{(1+\epsilon)\log n} > \log^{1/2} n$, where $\epsilon > 0$.

For uniform speed-limited (USL) mobility models discussed in Section 2.3, at each time, the node locations $\{x^{(t)}(u)\}$ have an empirical distribution that is uniform over $[0,\sqrt{n}) \times [0,\sqrt{n})$. Therefore, we now discuss the property of a sequence of geometric random graphs, $\mathcal{G}^{(t)}(n,r_n)$, with USL mobility model. We subdivide the network area on which the nodes live into smaller squares of side $\frac{r_n}{c}$, 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 $m = \frac{n}{(r_n/c)^2} = \frac{nc^2}{(1+\epsilon)\log n}$ and denote by E_i the event that small square i does not contain any node. In the next theorem, we show that when nodes move according to USL mobility model, all small squares will be populated w.h.p. in a sequence of length n^ρ, for some constant ρ.

THEOREM 2. *There exists a constant ρ such that if we divide the network into small square of side $\frac{r_n}{c}$ (with $r_n > \sqrt{\log n}$), every small square contains at least one node at every time step w.h.p. in a sequence of length n^ρ*

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 $m = \frac{n}{r_n^2}$. We can compute:

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

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

It is immediate that a single instantiation of the connectivity graph, every small square is populated w.h.p.

COROLLARY 1. *There is no empty small square with probability at least $(1 - O(\frac{1}{n^\epsilon}))$ in a sequence of length* 1.

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

THEOREM 3. *$\mathcal{G}(n,\sqrt{(1+\epsilon)\log n})$ 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. 3). It is easy to see that one can cover $\mathcal{B}_{4Rc}^{\mathcal{L}}(u)$ with a constant α $\mathcal{B}_R^{\mathcal{L}}(v_j)$. Hence,

$$B_{upper} \subseteq \mathcal{B}_{4Rc}^{\mathcal{L}}(u) \subseteq \bigcup_{j=1}^{\alpha}\mathcal{B}_R^{\mathcal{L}}(v_j) \subseteq \bigcup_{j=1}^{\alpha}\mathcal{B}_R^{\mathcal{G}}(v_j) \qquad (2)$$

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

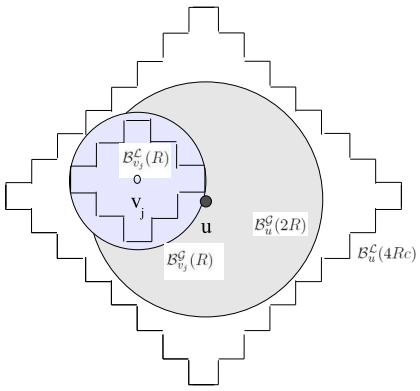

Figure 3: Proof of theorem 3

Note that it is possible to build a deterministic geometric graph for which this property does not hold (see [18]). Further, one can show that $\mathcal{G}(n,r_n)$ are *not* doubling w.h.p when $r_n < \sqrt{\log n}$. We prove this result [18]. At this point, we would like to emphasize that our results depend only on the doubling constant and can be applied to any type of networks or node configuration which lead to a doubling connectivity graph.

3.1 Inhomogeneous Topologies

In this subsection we state a result that shows that 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 one to the other. 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'[5]. A *metric embedding* is a bijective function $\phi : \mathcal{X} \to \mathcal{X}'$ which associates to a point in one metric space a point in another metric space.

DEFINITION 7 (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 $K = \frac{1}{D}$.

We can now relate the doubling dimensions.

THEOREM 4. *Consider a metric space $\mathcal{E}$ with doubling dimension β. A metric space $\mathcal{H}$ that can be divided in k sets $S_1, S_2, ..., S_k$ (with distances in $\mathcal{H}$), 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. A complete proof is given in [18]. □

Hence, if we can subdivide the communication graph into a constant number of subsets, such that each one is embedded with constant distortion into the Euclidean plane, the whole network is doubling. In common language, if the network is such that there exist a constant number of subsets of nodes, and for each of these subsets the Euclidean distance between nodes is proportional to the hop distance, the network is doubling. Consequently, topologies such as the one shown in Figure 2 are doubling. Here, we embed the communication graph into the Euclidean plane[6]. In such cases, it is obvious that geographic routing (greedy forwarding based on geographical co-ordinates) 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.

3.2 Dynamic Topologies

We now show that a sequence of $\mathcal{G}^{(t)}(n, r_n)$ of length n^ρ, for some constant ρ, with the USL mobility model is κ-smooth. As already seen in Theorem 3, such a sequence of graphs is doubling at every time instant.

THEOREM 5. *A sequence of $\mathcal{G}^{(t)}(n, r_n)$ of length $\leq n^\rho$, where nodes move according to the USL mobility model with maximum constant speed S is*

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

smooth w.h.p.

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 $\frac{q_2^{(t)}}{r_n} \leq d^{(t)} \leq \frac{\sqrt{5}\sqrt{2}q_2^{(t)}}{r_n}$. 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 r_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 $\frac{r_n}{c} = \frac{r_n}{\sqrt{5}}$, 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 $\frac{\sqrt{5}q_1^{(t)}}{r_n}$. 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

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

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 4, we illustrate this point. Similarly, at

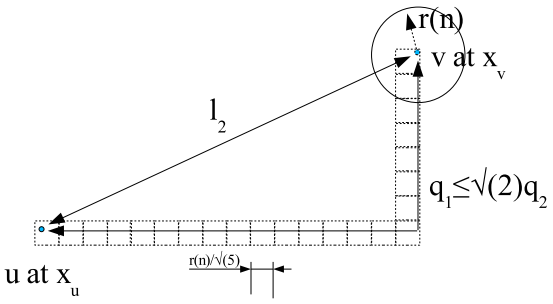

Figure 4: Upper and lower bounds for the shortest path

time $t + \tau$, the shortest path distance will be bounded by $\frac{q_2^{(t+\tau)}}{r_n} \leq d^{(t+\tau)} \leq \frac{\sqrt{5}\sqrt{2}q_2^{(t+\tau)}}{r_n}$. However, we know that the Euclidean distance can change by at most $2\tau S$ in τ time steps[7]. Consequently,

$$\frac{q_2^{(t)} - 2\tau S}{r_n} \leq d^{(t+\tau)} \leq \frac{\sqrt{5}\sqrt{2}(q_2^{(t)} + 2\tau S)}{r_n} \quad (3)$$

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

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

[5]We could for instance consider the points in the plane with the Euclidean distance and the nodes in the graph with the shortest path distance

[6]In our case, $\rho r_n d(u, v) \leq ||x(u) - x(v)|| \leq O(1)\rho r_n$. If this equation is true for all pairs of nodes, then the distortion is $O(1)$. The above rule implies that the Euclidean distance between neighbors in the communication graph should be at least $O(r_n)$. However, we can ignore the distances below 2 as we have a broadcast medium and the degree of a node does not impact the communication overhead (see Section 5)

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

□

One can now observe that 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 r_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 5, we know that the sequence is

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

-smooth w.h.p. Note that both terms decrease as a function of the communication radius r_n. Hence, we can set $r_n = 1$ without decreasing $\kappa(\tau, d)$. Similarly, both terms go down when the distance $d^{(t)}$ goes up. We can therefore also set $d^{(t)} = 1$, which is the smallest possible distance in an unweighted graph. Consequently, if we set $\tau = \nu d^{(t)} = \nu$, we can now write

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

which is constant for ν constant. □

4. ROUTING ALGORITHM

We develop the routing algorithm and performance analysis for a general class of dynamic networks which produce a sequence of doubling and smooth connectivity graphs. We have seen in Sections 2 and 3 that this applies to a class of wireless connectivity models with USL mobility. For notational convenience we will illustrate the ideas for a sequence $\mathcal{G}^{(t)}(n, r_n)$ geometric random graphs with USL mobility.

Our algorithm does *not* require the nodes to be equipped with a positioning device, such as the Global Positioning System (GPS). It is based on a hierarchical decomposition of the network uniquely dependent on the distance between nodes in the communication graph. This hierarchy is maintained and refreshed over time. Every node is a member of one cluster at each level in the hierarchy. To establish a route for a given source destination pair, the source searches this hierarchy until it finds a cluster of which the destination is a member. In turn, a recursive search down the hierarchy inside this cluster allows us to reach the destination. Hence, our algorithm does not require a location service per say. Rather, we perform an efficient search for the destination based on the destination's identifier. In the sequel, we will show how we designed such a scheme and prove its efficiency analytically.

We decompose a time step into two phases: a *beaconing* phase and a *forwarding* phase. In the former phase, a set of routes are established by letting all or a subset of nodes flood the network at geometrically decreasing radii and nodes register with beacon nodes. In the latter phase, this subset of routes is then utilized by source nodes to efficiently search for the destination. Every node is equipped with a routing table as shown in Table 2. We will first describe two procedures used in the beaconing and the routing phase.

Node identifier	*distance [hops]*	*level*	*next hop*
⋮	⋮	⋮	⋮

Table 2: Routing Table RT

flood(R,level) procedure: When a node u initiates the $flood(R, level)$ procedure, it broadcasts a *flood packet* as shown in Table 3 to its direct neighbors in $\mathcal{G}(n, r_n)$. The *hop count* field is initialized to 0 and the content of the *Level* field will be specified in the sequel. All nodes can compute the maximum hop count given the level of the source. The

Pkt. Type	*Node Id.*	*Hop Count*	*Level*
$O(1)$ bits	$O(\log n)$ bits	$O(\log n)$ bits	$O(\log \Delta)$ bits

Table 3: Flood Packet

neighbors which receive this packet, after increasing the hop count by 1, add an entry to their routing table for node u if no entry for the same node with a lower or the same hop count is present in the RT for the same level. The *next hop* field is set to the identifier of the node from which the packet was received. The level field in the routing table is set to the level given in the packet. In turn, these nodes broadcast this packet to their neighbors which follow the same procedure and update the routing table if necessary. The packet is discarded when the hop count reaches the maximum hop count (which is a function of the level). Note that with this procedure, every node forwards the packet at most once and the distance added to the routing table is the shortest path distance in $\mathcal{G}(n, r_n)$.

probe(relay,destination) procedure: This procedure consists in sending a *probe packet* (see Table 4) to a relay node for which the source has an entry in its routing table. The relay node will set the success bit to 1 if it has an entry for the destination and 0 otherwise. We will make sure that all nodes on the path between the source and the relay node have an entry for the relay node in their routing table. Additionally, nodes on the path add a temporary entry for the source in the routing table. They set the *next hop* field to the identifier of the node from which they received the packet and leave the *level* and *distance* field empty. Upon

Pkt. Type	*Relay Id.*	*Dest. Id.*	Success
$O(1)$ bits	$O(\log n)$ bits	$O(\log n)$ bits	1 bit

Table 4: Probe Packet

receiving the packet, the relay node can either answer to the source on the reverse path we just created if the answer is negative. Alternatively, it can take action as explained in the sequel if it has an entry for the destination.

We now separately detail the beaconing and the routing algorithms underlying our routing protocol

4.1 Beaconing Algorithm

We will first describe the first time step, where nodes have not yet moved. Let the *cover radius* at level i, for $i = 1, ..., \log \Delta$ (Δ 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 $\kappa(\nu d, d) \leq \kappa d$, $\forall d$. The idea of the algorithm is to build

a hierarchical cover of the network *i.e.,* we would like every node in the network to be within r_i of a beacon node at every level i. We say that when a node is within r_i of a beacon b at level i, it is a member of b's *cluster* at level i. A node can only be in one cluster at every level. To achieve this, we let the nodes flood in a random order which can change at every time step. Every node u is a beacon at a given level $\beta(u)$. The flooding radius, however, will depend on the highest level at which a node is not covered. Let us denote by $h(u)$ the highest level at which node u is not covered. When node u's turn to flood comes, it will determine the value of $h(u)$ set $\beta(u) = h(u)$ and call $flood(f_{h(u)}, h(u))$. A node v which receives this flood will determine the lowest level at which it could be a member of u's cluster, say $l(v)$. That is, it will determine the lowest value j for $l(v)$ such that $d(u, v) \leq 2^j$. This distance $d(u, v)$ is known since v just received a flood packet from u. It will then become a member of u's cluster for all levels above $l(v)$ for which it has no membership yet and are below $\beta(u)$. If a node becomes a member of u's cluster, it will send a *membership packet* (see Table 5) back to u which will store the identifier of the

Pkt. Type	*Node Id.*	*Beacon Id.*	*Level*
$O(1)$ bits	$O(\log n)$ bits	$O(\log n)$ bits	$O(\log \Delta)$ bits

Table 5: Membership Packet

nodes in its cluster. Note that a node v could send back none or several such packets to u. Note that u also applies this procedure to itself, and consequently could be a beacon at level i but not at level $j < i$.

The control traffic will be dominated by the messages sent back by nodes to beacons when they become members of a cluster. On the other hand, we do not want the distance between a node and the beacons it belongs to to grow by more than a constant factor. Since we consider that the maximum speed of the node is constant, the higher the level of a beacon, the more time it will take for nodes to double their distance to this beacon. We want to *elect* new beacons and

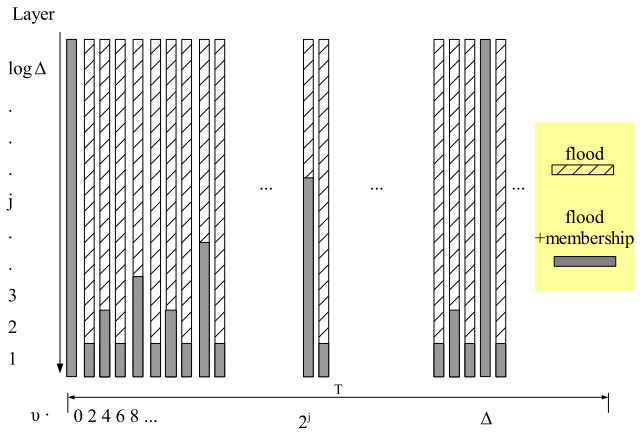

Figure 5: The memberships up to level i are updated every $\nu 2^i$ time steps. At the levels above, beacons elected at earlier time steps simply flood again.

update memberships only for levels at which the distances could have been multiplied by κ. Recall that the network is κ-constrained. Consequently, the distance $d^{(t)}(u, v)$ between two nodes u and v cannot change by a factor κ in less than νd time steps (see Corollary 2). In particular, if a node is at distance 2^i of a beacon at the time it becomes a member of its cluster, then we have $d^{t+\nu 2^i} \leq \kappa 2^i$. Hence, we will update the memberships at level i every $\nu 2^i$ time steps (see Figure 5). This will lead 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. This is why 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 pair of nodes[8]. In Figure 6, we give a simple with three levels. The beaconing algorithm is presented in Algorithm 1. It is important to note that the routes are updated at every time step and consequently routing toward a beacon will always be successful. Further, when the membership at a given level i is

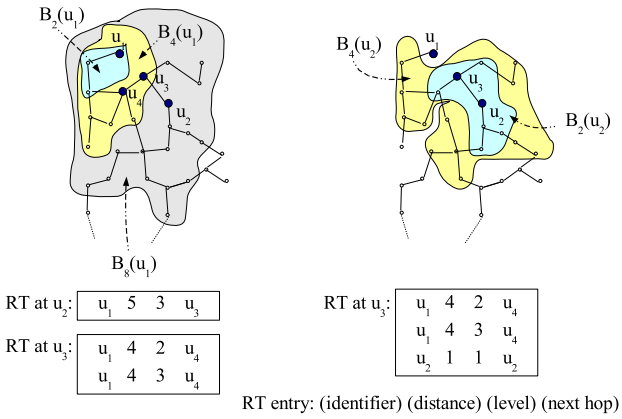

Figure 6: The example start with empty routing tables. First, on the left, node u_1 floods at level 3. We focus on nodes u_2 and u_3. Node u_2 is within 8 hops from u_1 but further away than 4 hops. Consequently, it can only had an entry for node u_1 at level 3. At the same time, node u_3 can add an 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.

updated, all the memberships at the levels $j < i$ will also be updated, and all memberships at these levels canceled.

4.2 Forwarding Algorithm

The forwarding algorithms works as follows: a source node u wishing to communicate with a target node v will search

[8]Note that this gives us a way to detect beacon failures. A recovery mechanism would let nodes that do not hear their beacon at a given level anymore start a local beacon election process *i.e.,* such a node starts a random timer and can elect itself as a beacon if it is not covered by another node's flood. Memberships could then be updated.

Algorithm 1: Beaconing Algorithm at node u

```
Data: Routing Table, Time t
begin
    Let Γ = max{0 ≤ j ≤ log Δ | t mod ν2^j = 0};
    Clear routing table entries with level ≤ Γ;
    Let β(u) be the level at which u is a beacon;
    if π(ℓ) = u then
        if β(u) ≤ Γ then
            Let h(u) be the highest level at which u is
            not covered;
            β(u) = h(u);
        end
        flood(f_h(u), h(u));
    end
end
```

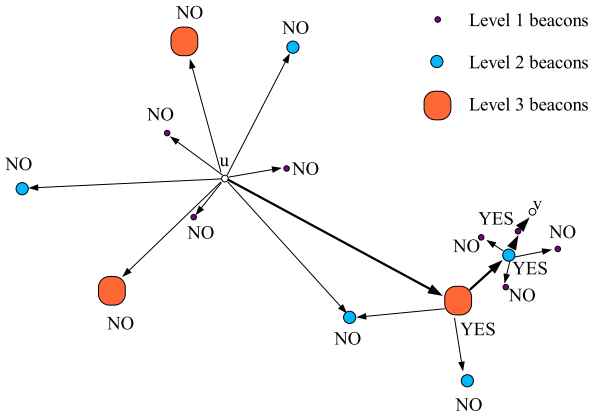

Figure 7: Node u has a packet for node v. It searches in its routing table for all beacons it knows of at level 1 and sends them a probe packet containing v's identifier. None of the beacons at level 1 has an entry for this node and consequently they all answer negatively to node u. Next, node u repeats 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. A 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.

for v by first probing all the level 1 beacon it knows of. To do so, it looks at the last column of its routing table and selects all nodes it knows of at level 1. If all answers are negative, it will probe all level 2 beacons it knows of. The procedure is repeated as long as all beacons answer negatively. A beacon at level i that has an entry for the destination in its routing table will not answer directly to the source. Rather, it will probe all the level $i-1$ beacons it knows of. We will show in the next section that one of these beacons must have an entry for the destination. That beacon in turn probes all the beacons in knows of at level $i-2$. Meanwhile, the other beacons at level $i-1$ will answer negatively to the beacon at level i. The procedure is repeated recursively until the target itself is reached[9]. The target will then answer to the source on the reverse path which will later be used for communication between the source and the destination. We illustrate the forwarding procedure conceptually in Figure 7.

4.3 Load-balancing

This approach guarantees a low network wide control traffic overhead. Even though over a long period of time all nodes will get approximately the same average overhead, beacons at the highest levels might get overloaded by the membership packets of the nodes in their cluster when a membership update takes place. 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 a $j < i$ 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 anymore be reachable 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 will no longer get overloaded. Rather, the traffic will 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

[9]If the destination node fails, at least one beacon on the path will fail in finding a next step for the routing path. In this case, this beacon can reply to the source that the destination is not available anymore.

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. Even though it is not the focus of this paper, we expect the memory requirements for this load-balanced approach to be polylogarithmic in n.

5. PERFORMANCE ANALYSIS

We will analyze the performance of our algorithm analytically both in terms of control traffic and of route stretch for a sequence of doubling and smooth connectivity graphs. We will use $\mathcal{G}^{(t)}(n, r_n)$ with USL mobility for illustration. We derive order results for which the constants can be computed explicitly. Our numerics confirm that these results do not only hold in asymptotia.

The bounds derived in this section hold with w.h.p. when we are in a sequence of α-doubling connectivity graphs of length n^ρ. In the sequel, α, κ and ν are the constants derived in Section 3. Let us denote by $\Delta = O(\sqrt{(\frac{n}{\log(n)}}))$ the diameter of the network. To bound the control traffic necessary for beaconing, we will rely on the α-doubling property of the metric space to show that a node can only hear a constant number of beacons at every layer. We will first show that a ball of radius $2R$ around any node u can only contain at constant number of balls (clusters) of radius R, when we select the centers of the balls of radius R in an arbitrary order and ensure that two centers cannot be closer than R. We will later use this result to show that a node can hear at most a constant number of beacons at any given level.

THEOREM 6 (RANDOM COVER). *Let $\mathcal{B}^X_{2R}(u)$ be a ball of radius $2R$ in a graph metric (X, d) with doubling constant α centered at u. Then, one can select at most $k \leq \alpha^2$ nodes v_i, $(i = 1, 2, .., k)$ such that $\mathcal{B}^X_{2R}(u) \subseteq \bigcup_i^k \mathcal{B}^X_R(v_i)$ and $min_{(i,j)} d(v_i, v_k) > R$.*

PROOF. By definition of an α-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 α-doubling metric space (X, d). Then, one can select at most $k \leq (\frac{R}{R'})^{2log(\alpha)}$ nodes v_i, $(i = 1, 2, .., k)$ such that $\mathcal{B}^X_R(u) \subseteq \bigcup_i^k \mathcal{B}^X_{R'}(v_i)$ 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 $(\eta)^{2log(\alpha)}$ independent of n.*

PROOF. Let $R = 2^i R'$. Hence, R' is doubled $log \frac{R}{R'}$ times to obtain R. By Theorem 6, B can be covered by $\alpha^{2log \frac{R}{R'}} = (\frac{R}{R'})^{2log(\alpha)}$ 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 $(\frac{R}{R'})^{2log(\alpha)}$ beacons, since beacons must be at least R' apart.

5.1 Control Traffic

THEOREM 7. *The average control traffic overhead per time step for beaconing is at most $O(n \log^2 n)$ bits.*

PROOF. We will 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. Further, at the time the memberships are updated at level i, a beacon node at this level *cannot* be within $r_i = 2^i$ of another beacon at that level. If it were the case, this node would not elect itself as a beacon at this level. Level i is updated every $\nu 2^i$ time steps. Consider a node u. By corollary 5, only nodes that are within κf_i at the time the memberships are updated at level i could move within f_i of u in at most $\nu 2^i$ time steps. 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 level i beacons in a ball of radius κf_i at the time the membership is updated. In turn, node u will broadcast[10] the flood packets of at most that many beacons for this level i. By corollary 3, this number is a constant[11] given by $(\frac{\kappa f_i}{2^i})^{2log(\alpha)} = (3\kappa^2)^{2 \log \alpha}$. 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 8 (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 $\sum_{i=1}^{\log \Delta} \frac{T}{\nu 2^i} 2^i \log n = \frac{\log \Delta}{\nu} \log n$. □

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

THEOREM 9 (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.

[10] 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 beacon of which a flood packet is received.

[11] In the load-balanced scheme, this constant is $(5\kappa^2)^{2 \log \alpha}$.

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. □

5.2 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 destination location discovery takes a negligible fraction of a flow throughput.

THEOREM 10 (ROUTING STRETCH). *The worst case multiplicative routing stretch is $O(1)$.*

PROOF. We first analyze the stretch without load balancing. 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 $d(u, b_v(k)) \le \kappa(2^{k+1}+2^k)$, 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 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 $d(u, b_v(k)) \le f_k$. 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:

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

In the worst cast, 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-constrained. 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 $\frac{r^{(t+\nu 2^k)}(u,v)}{d^{(t+\nu 2^k)}} \le 6\kappa^2 = O(1)$. □

Every node can only hear floods from a constant number ($\mu = (3\kappa^2)^{2\log\alpha}$, see Theorem 7) 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 throughput of a flow since it consumes roughly the equivalent of a few packet headers of a flow from source to destination. A similar statement can be made for the load-balanced case.

6. IMPLEMENTATION ISSUES

In Section 5, we have computed worst case bounds which may be conservative in terms of constants. In this section, we explore this aspect by looking at simulation results for the control traffic and for the stretch. Recall that we had

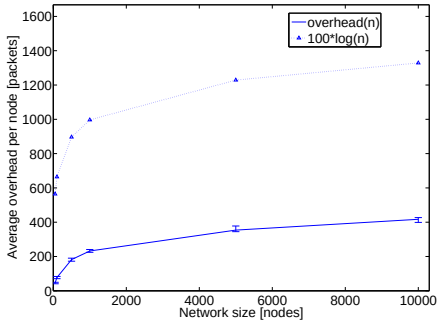

Figure 8: Average control traffic overhead per node in packets as a function of the network size. Nodes move at a speed of maximum speed of 1. The confidence interval is given by the 95% and 5% percentiles. The size of a packet is $O(\log n)$ bits. We also plot $100\log n$ to show that our analytical predictions match the simulation results.

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 8, 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 $O(\log n)$).

Similarly, in Figure 9 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.

In practice, the random permutations on the nodes, which determines the order in which the flooding occurs, could be implemented by using random timers *i.e.,* by letting all nodes draw a random delay independently of each other every ΔT seconds. The interval from which nodes draw this delay should be made sufficiently large so that we can avoid collisions. We conjecture that our scheme can be implemented without synchronization between nodes[12]. A level in the hierarchy will be rapidly covered, and in a practical implementation the covers at different levels could be built in parallel. Hence, we speculate that it is possible to reduce the length of the beaconing phase to a small constant times the maximum round-trip time.

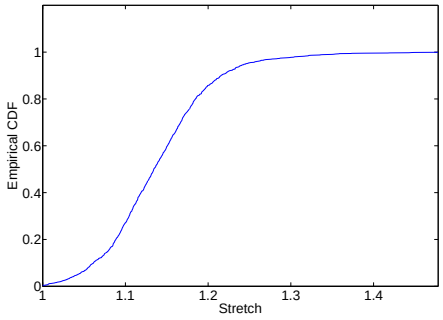

Figure 9: Empirical cumulative distribution of stretch (route length/shortest path) for a network with 1000 **nodes moving at a maximum speed of** 1**.**

7. 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 2.1 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.

8. REFERENCES

[1] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In *ICDCS*, 2006.

[12]A beacon node could for instance flood for a given time period, independently of other beacons, and then indicate in its flood that a new election must take place to cover its region of the network. Nodes would then start their random timer again, making strict synchronization not crucial

[2] J.-Y. L. Boudec and M. Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *INFOCOM*, 2005.

[3] T.-H. H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *SODA*, 2005.

[4] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. a. Stoica. Beacon-vector routing: Scalable point-to-point routing in wireless sensor networks. In *NSDI*, 2005.

[5] C. Gavoille. Routing in distributed networks. *ACM SIGACT News*, page 36, 2001.

[6] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS*, pages 534–543, 2003.

[7] P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications*, 1998.

[8] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[9] D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *Ad Hoc Networking*. 2001. Book.

[10] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *MOBICOM*, page 243, 2000.

[11] G. Konjevod, A. W. Richa, and D. Xia. Optimal stretch name independent compact routing in doubling metrics. In *PODC*, 2006.

[12] S. R. Kulkarni and P. Viswanath. A deterministic approach to throughput scaling in wireless networks. In *ISIT*, page 351, 2002.

[13] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris. Scalable location service for geographic ad hoc routing. In *MOBICOM*, page 120, 2000.

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

[15] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MOBICOM*, page 96, 2003.

[16] G. Sharma, R. Mazumdar, and N. Shroff. Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. In *INFOCOM*, pages 1–12, 2006.

[17] K. Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *STOC*, page 281, Chicago, IL, USA, 2004.

[18] D. Tschopp, S. Diggavi, and M. Grossglauser. Hierarchical routing over dynamic wireless networks. Technical report, EPFL, 2007.

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