Geometric Routing without Geometry

Mirjam Wattenhofer[1], Roger Wattenhofer[2], and Peter Widmayer[1]

[1] Department of Computer Science, ETH Zurich
[2] Computer Engineering and Networks Laboratory, ETH Zurich

Abstract. In this paper we propose a new routing paradigm, called *pseudo-geometric routing*. In pseudo-geometric routing, each node u of a network of computing elements is assigned a *pseudo coordinate* composed of the graph (hop) distances from u to a set of designated nodes (the *anchors*) in the network. On theses pseudo coordinates we employ greedy geometric routing. Almost as a side effect, pseudo-geometric routing is not restricted to planar unit disk graph networks anymore, but succeeds on general networks.

1 Introduction

With the advent of ad hoc, sensor, mesh or peer-to-peer systems, networking research has recently received a second wind of attention. In the center of interest is the routing problem. Contrary to established networks such as the Internet, these new networks ask for a novel generation of routing protocols. First, ad hoc or sensor networks run on plain and feeble hardware that does not allow nodes to store large routing tables as needed by classic routing algorithms such as distance-vector or link-state routing. Second, peer-to-peer or ad hoc networks are highly dynamic – the topology of the network is changing constantly, at much higher rates than in conventional networks such as the Internet. Consequently, this leads to an immense exchange of control messages in classic routing protocols.

In order to tackle the routing problem for ad hoc/sensor/mesh/peer-to-peer networks, the research community has proposed an array of innovative routing protocols and paradigms. Originated from multihop radio networks, geometric routing is a prominent representative, combining small memory overhead with few updates per topology change. In geometric routing the nodes do not have routing tables at all, instead it is assumed that the nodes have *coordinates* in the Euclidean plane.

The early proposals of geometric routing—suggested twenty years ago by Takagi and Kleinrock [25]—were of purely greedy nature: Each node knows its own coordinate, as well as the coordinates of its neighbors. When receiving a message containing the coordinate of the destination, a node forwards the message to its "best" neighbor – the neighbor node geometrically closest to the destination.

Yet, already in simple configurations greedy geo-routing can fail if the message reaches a local minimum with respect to the distance to the destination, that is a node without any "better" neighbors. This deadlock, however, can be resolved by using more elaborate geo-routing protocols (see Section 2). In this paper we advocate using the original greedy geo-routing, however, with a higher-dimensional geometric space.

Another problem is the availability of position information (coordinates) which is needed to run a geo-routing algorithm. Clearly, one possible technical solution is to equip each node with a Global Positioning System (GPS) receiver. However, in comparison to a sensor node, a GPS receiver is clumsy, expensive, and energy-inefficient. Moreover, GPS reception might be obstructed by climatic conditions; if nodes are deployed indoors, there is no reception at all.

As a GPS-alternative, researchers proposed to compute so-called *virtual coordinates* merely from connectivity or distance information and employ geometric routing schemes on those coordinates. However, the apparent computational complexity of virtual coordinates [20, 18] discourages from using them in real systems.

In this paper we advocate a paradigm shift. Instead of trying to solve the tough virtual coordinates problem and then use advanced geo-routing techniques, we go back to the roots and use greedy geometric routing on down-to-earth virtual coordinates.

In particular, we propose that the virtual (or *pseudo*) coordinate of a node u is a vector, composed by the graph (hop) distances from u to a set of designated nodes (the *anchors*) in the network. Almost as a side effect, pseudo-geometric routing is not restricted to planar unit disk graph networks anymore, but succeeds on general networks. We believe that the coordinate of a node will be relatively stable even if the network topology changes, hence making our routing scheme applicable for highly dynamic networks.

In order to gain a deeper insight into this new routing paradigm and to explore its algorithmic foundations and limits, we assess its potential by investigating various basic network topologies.

After giving an overview of related work in the following section, we state the model used in this paper in Section 3. In Sections 4 – 9 we analyze the properties of our routing paradigm for different network topologies, namely rings, trees, grids, unit disk graphs, butterfly networks, and hypercubes. Section 10 concludes on the paper.

2 Related Work

There are a hand full of routing protocols which are similar in spirit. We discuss selected protocols in this section.

The link-reversal paradigm [11] improves significantly on the standard distance-vector routing protocol[3] by not updating the distances with each topology change. Recently, the performance of the link-reversal paradigm was analyzed [4]. Besides being more apt than distance-vector routing to be applicable in highly dynamic networks, the link-reversal protocol also requires less memory.

Speaking of memory-efficiency: In classic large scale communication networks a dominant problem is to develop *compact* routing schemes which feature low memory overhead per node and still produce efficient routes between source and destination. The first routing scheme which addresses the efficiency-memory tradeoff was proposed in [16], where the idea of hierarchically clustering a network into levels and using the resulting structure for routing was introduced.

[3] In distance-vector, each node stores the distance to each destination, and which link to follow – a derivative of distance-vector is deployed in the Internet BGP protocol.

Subsequently, the trade-off between memory space and stretch factor was theoretically analyzed [21, 14] and a plethora of compact routing schemes was proposed [8, 5, 26, 2]. For comprehensive surveys on compact routing see [12, 13, 15].

An important branch of compact routing, so called *interval* routing, is based on the idea of grouping nodes in cyclic intervals and was first suggested in [23] for tree networks. Later on this work was extended to other network topologies [27, 9, 7]. It is worth noting that an interval routing scheme, once computed for a graph, can be used to perform other tasks than routing. [10] proposed a $\Theta(n)$ broadcast algorithm that uses only interval routing labels.

The best compact routing schemes provide amazing memory-stretch ratios. However, they are hardly applicable in dynamic networks since all routing tables have to be computed from scratch if the topology of the network changes.

A well-studied routing paradigm for radio networks is geometric (a.k.a. geographic, location-based, position-based, or simply geo-routing) routing. The first proposals were of purely greedy nature. As pointed out in the introduction, already in simple configurations greedy geo-routing can fail if a message reaches a local minimum. This deadlock problem, however, was resolved by the employment of face routing, which explores the boundaries of faces of the planarized network graph [17]. In recent years, geo-routing has experienced several improvements. The routing schemes GFG [3] and GOAFR+ [19] advocate a combination of greedy and face routing. Whereas GFG does not give competitive worst-case guarantees, GOAFR+ is a routing algorithm which is efficient for average-case networks as well as asymptotically worst-case optimal. Recently, the locality aware location service LLS [1] proposes a solution how the coordinates of mobile destinations can be learned efficiently using a peer-to-peer-like scheme.

Unfortunately, it is not always feasible to assume that each node in the network knows its position. As an alternative, researchers proposed to compute so-called *virtual coordinates*, coordinates computed merely from connectivity or distance information, and employ geo-routing schemes on top of these coordinates. In [22] a greedy routing scheme is employed on the virtual coordinates which are obtained by a spring-based algorithm. By solving a convex linear program the coordinates of the network nodes are estimated in [6], whereas the heuristic in [24] is based on multidimensional scaling.

Apart from these heuristics there is little work on virtual coordinates: In [20] the authors present an approximation algorithm for the virtual coordinates problem, with polylogarithmic approximation ratio only. In a lower bound paper [18] it is shown that virtual coordinates cannot be approximated arbitrarily well. These two results dampen our hopes that using geo-routing on virtual coordinates in real systems is practical.

Instead of embedding the nodes in the two dimensional Euclidean space, in this paper we propose to embed the nodes in a sufficiently high dimensional pseudo geometric space and employ a greedy geo-routing scheme on top.

3 Model

Let a network on n nodes be given, where k nodes $a_1, a_2, \ldots, a_k$ are designated *anchors* and there exists a unique order on the anchors with $a_1 \prec a_2 \prec \ldots \prec a_k$. Each node u in the networks knows the underlying network topology and is furthermore able to

determine its graph (hop) distance d_i to each one of the k anchors a_i. The *(pseudo-) coordinate* of node u is then defined to be $(d_1, \ldots, d_k)$. Thus, the network is embedded in a pseudo k-dimensional space. Each node knows in addition to its own coordinate the coordinates of all its direct neighbors.

In *pseudo-geometric routing algorithms* when receiving a message containing the (pseudo) coordinate of the destination, a node forwards the message to its "best" neighbor – the neighbor node geometrically closest (in the pseudo geometric space) to the destination. In the following we say that the pseudo-geometric routing problem can be solved if there is a pseudo-geometric routing algorithm which guarantees message delivery.

In general, a routing algorithm may have two basic problems. The first and foremost problem, we henceforth also call *naming problem*, is that all nodes must have unique identifiers, otherwise the destination is not identifiable in general. Once the naming problem is solved, the algorithm furthermore has to guarantee that any destination must finally be reached from any source –the *routing* problem.

In the following, we exemplarily demonstrate at the showcase where the graph is a line which properties of the pseudo-geometric routing problem we analyze, why these properties are important from a theoretical and practical point of view and how the next sections are structured. In general, we first concentrate on the naming problem before we finally give a pseudo-geometric routing scheme.

The first property we explore is the *minimal* number of anchors we need to solve the naming problem. Theoretically speaking, we give a *lower bound* on the number of anchors. From a practical point of view this is a quite natural property. Amongst others, it gives the minimal amount of storage which is needed per node to solve the naming problem.

Example: *If a node* a *with degree one on the line is chosen to be an anchor, each node on the line has a unique coordinate, since each node has a unique distance to* a. *This leads to the following lemma.*

Lemma 1 (min). *Choosing a node on the line with degree one solves the naming problem.*

The second property we are interested in is an *upper bound* on the number of anchors which are needed to solve the naming problem, in the following sense. If we allow an adversary to choose the anchors arbitrarily, how many anchors must be chosen until the naming problem is solved? In practice it is often not feasible to deliberately assign anchors, but anchors are chosen more or less arbitrarily, hence we might come across this upper bound.

Example: *For the line, this upper bound is trivially obtained by observing that for any two nodes on the line each node has a unique distance vector.*

Lemma 2 (any). *Two arbitrarily chosen anchors on a line solve the naming problem.*

In networks another crucial factor is the degree of *locality*. Having a local solution is clearly favorable to having a global solution, where we use local in the sense of the anchors being in a constant size neighborhood of each other. The advantage of local

solutions lies in the fact that failures and updates can be dealt with locally. Leading to the third property we look at our example.

Example: *Since two arbitrary anchors on the line solve the naming problem, clearly two* incident *anchors solve the naming problem.*

Lemma 3 (local). *Two* incident *nodes on the line solve the naming problem.*

Finally, we give a pseudo-geometric routing scheme. Towards this goal we explicitly assume that the anchors are chosen in some way and based on this choice of anchors show how nodes pass a message such that it eventually reaches the destination. For most of the analyzed topologies we show that the chosen path is actually the shortest path between source and destination.

Example: *Choose one anchor on the line, namely a node with degree one. Based on its own coordinate and the coordinate of the destination, a node immediately knows to which neighbor to pass the message such that it reaches the destination on the shortest path.*

Theorem 4. *The pseudo-geometric routing problem on the line can be solved (locally) with one anchor.*

Wrapping up, in the following sections we concentrate for each topology on three important properties related to the naming problem. The first property called *min* is a lower bound on the number of anchors we need to solve the naming problem, whereas *any* is an upper bound. *Local* gives a choice of anchors which solves the naming problem and is local, if such a choice exists. With local we mean that the anchors are within constant graph distance from each other. We then give a pseudo-geometric routing scheme which guarantees message delivery.

4 Ring

4.1 Naming

With one anchor a only, there are $\lfloor (n-1)/2 \rfloor$ pairs of nodes with pairwise same distance to a, that is, they cannot be distinguished. On the other hand, with two anchors a and b, which are at distance $d \neq n/2$ to each other, or arbitrary three anchors each node has a unique coordinate. Leading to following lemmas.

Lemma 5 (min, local). *If chosen properly, exactly 2 anchors solve the naming problem in the ring, specifically two* incident *anchors solve the naming problem.*

Lemma 6 (any). *3 arbitrarily chosen anchors solve the naming problem in the ring.*

4.2 Routing

Suppose we have two anchors which solve the naming problem in the ring. A node u can reconstruct the position of the anchors based on its and its neighbors' coordinates. Furthermore, u knows the position of the destination relative to the anchors and hence can pass the message to the neighbor which is nearest to the destination. By this discussion and Lemma 5 the following theorem can be deduced immediately.

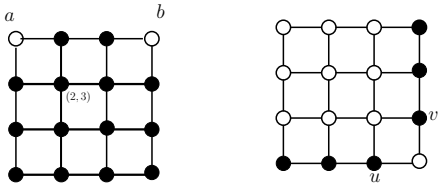
Theorem 7. *The pseudo-geometric routing problem on the ring can be solved (locally) with 2 anchors. Furthermore, the chosen route between source and destination is a shortest path.*

5 Grid

5.1 Naming

With one anchor only at least $\sqrt{n}$ nodes do have the same coordinate in the grid. On the other hand, if we choose one anchor a such that it lies in the upper left corner of the grid and another anchor b which lies in the upper right corner of the grid, all nodes have different coordinates (see Figure 1(a)).

Lemma 8 (min). *If chosen properly, we need exactly 2 anchors in the grid to solve the naming problem.*



(a) Two Anchors Solve the Naming Problem.

(b) $(\sqrt{n}-1)^2+1$ anchors do not solve the naming problem.

Fig. 1. *min* and *any* for the grid.

Lemma 9 (local). *It is not possible to solve the naming problem in the grid locally.*

Proof. Consider an arbitrary subgraph of the grid with constant diameter, where all nodes in the subgraph are anchors. Then there are two nodes incident to a corner node which are not anchors. Those two nodes cannot be distinguished by the anchors. (See Figure 1(b) for an example.)

Lemma 10 (any). *If the anchors are chosen arbitrarily, then at least* $(\sqrt{n}-1)^2+2$ *anchors are needed to solve the naming problem.*

Proof. We prove the Lemma by giving an example where $(\sqrt{n}-1)^2+1$ anchors are already chosen, but there are still two nodes which are not distinguishable. Hence, at least $(\sqrt{n}-1)^2+2$ nodes are needed to solve the naming problem. The example is depicted in Figure 1(b). The white nodes are the anchors which are already chosen, whereas u, v are the nodes with same coordinate.

5.2 Routing

Given that the anchors a, b are placed as proposed in Lemma 8, a node u can compute the position of each anchor, based on its coordinate and the coordinate of its neighbors. It thus knows where the destination lies and hence can pass the message to one of its neighbors which lies in the quadrant of the destination guaranteeing that the message always reaches the destination on a shortest path.

Theorem 11. *The pseudo-geometric routing problem on the grid can be solved with two anchors. Furthermore, the chosen route between source and destination is a shortest path.*

6 Tree

6.1 Naming

Before we prove the minimal number of anchors needed to solve the naming problem in a tree, we define the following helpful term. Given a tree $T=(V,E)$, let the root r of the tree be an arbitrary node and call the such rooted tree T_r. Consider those nodes in T_r which have degree at least three and have no descendant with degree at least three. Formally $L(T_r)=\{v\in V; deg(v)\geq 3, \max_{u\in T_r(v)} deg(u)\leq 2\}$, where $deg(v)$ is the degree of a node v and $T_r(v)$ is the subtree of T_r rooted in v. Then the *minimal coverage number* $mc(T)$ of T is defined as $mc(T)=\max_{r\in V}\sum_{v\in L(T_r)}(deg(v)-2)$.

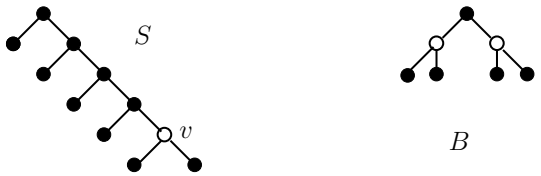


Fig. 2. Minimal coverage number in stair-tree S and complete binary tree B.

To get an intuitive understanding of the minimal coverage number, we depict two examples in Figure 2.

In the stair-tree S, $L(S)=\{v\}$ and $mc(S)=deg(v)-2=1$. Note, that choosing v and the root as anchors also solves the naming problem.

In the complete binary tree $B=(V,E)$, $L(B)=\{v\in V; \exists l\in V, deg(l)=1, d(v,l)=1\}$, that is all nodes which are neighbors of a leaf. Then, $mc(B)=|L(B)|\cdot$

$(3-2) = (n+1)/4$. Again, note that choosing every second leaf as an anchor also solves the naming problem.

Lemma 12 (min). *Let $mc(T)$ be the minimal coverage number in a tree $T = (V, E)$. Then, we need at least $mc(T)$ anchors.*

Proof. Each node $u \in V$ with degree at least 3 must have at least one anchor in each but one of its neighbor-subtrees, where we use the term neighbor-subtree for subtrees of T rooted in neighbor nodes of u. Otherwise, if there are no anchors in more than one of its neighbor-subtrees, there are two neighbors u_1, u_2 of u which cannot be distinguished, since the distance from each anchor to u_1, u_2 is exactly the distance to u plus one for both. Hence, u_1 and u_2 have the same coordinate. Thus, the number of anchors we need is at least the minimal coverage number. (We have to subtract 2 from the degree of each node to avoid double counting and have a true lower bound.)

It is worthwhile to observe that the above discussion of Figure 2 shows that the minimal coverage number is –at least for some trees– (almost) tight.

By the Lemma above and the depicted example (Figure 2) the following lemma is self-explaining.

Lemma 13 (local). *It is not possible to solve the naming problem in a tree locally.*

Lemma 14 (any). *If the anchors are chosen arbitrarily, we need up to $n-1$ nodes to solve the naming problem.*

Proof. If in the star we choose the center as an anchor, we additionally have to choose $n-2$ of the $n-1$ siblings in order to distinguish each pair of siblings.

Lemma 15. *In a tree it is always sufficient to choose all leaves as anchors.*

Proof. Consider two arbitrary nodes u, v in the tree. The nodes lie on a path p connecting u, v via their least common ancestor and furthermore connecting u, v to a leaf-anchor l. A leaf-anchor is a node with degree one on this path p. Hence, by Lemma 1, l can distinguish between all nodes on this path, specifically between u and v. This shows that any two nodes can be distinguished.

6.2 Routing

Assume now that each leaf is an anchor[4]. Based on its own coordinate and the coordinate of its neighbors a node u knows for each anchor in which direction it lies. By the choice of the anchors, the coordinate of the destination t is smaller than u's coordinate in at least one position i. Then t must lie in the direction of the corresponding anchor a_i, since in all other directions the distance to a_i is increasing. Thus, u passes the message to its neighbors which lies in this direction and consequently the message always reaches the destination on the shortest (and only) path.

Theorem 16. *The pseudo-geometric routing problem on the tree can be solved with $|L|$ anchors, where $|L|$ is the number of leaves in the tree.*

[4] There are trees, where this choice is near to optimal, like the complete binary tree, but there are others, like the stair-tree in Figure 2, where choosing all leaves is wasteful.

7 Unit Disk Graph

7.1 Naming

In order to prove a lower bound on the number of anchors needed in the unit disk graph[5] to solve the naming problem, we construct a unit disk graph which experiences this lower bound.[6] Specifically, we construct a unit disk grid tree, that is a unit disk tree, which is a subgraph of the grid graph. As we have seen in Section 6, trees with a large number of leaf-siblings (that is leaves which have a common father) have a large minimum coverage number and thus experience a large lower bound. Hence, we build a unit disk tree in such a way that the number of leaf-siblings is maximal. We henceforth show how the graph is constructed, lower bound the number of nodes in graph distance k from the root and then lower bound the total number of nodes in the whole graph. Based on those bounds we finally prove that there are $\Theta(n)$ sibling-leaves and hence, by Lemma 12 we need $\Theta(n)$ anchors to solve the naming problem.

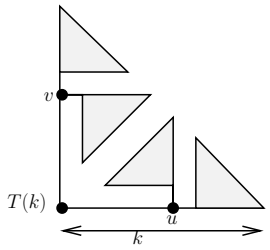


Fig. 3. Recursive construction of unit disk graph.

The unit disk grid tree $T(k)$ is built recursively as depicted in Figure 3. The tree with depth k consists of four trees with depth $(k-3)/2$ each, depicted by shaded triangles in the figure. The root of the new tree is connected through two paths of length $(k+1)/2$ to two nodes u, v which are each connected themselves to two of the smaller trees.

Lemma 17. *In $T(k)$ there are at least $\frac{(k+3)^2}{8}$ nodes which are at distance k from the root and $\frac{(k+3)^2}{16}$ which are at distance $k-1$ from the root.*

Proof. Let $L(k)$ be the number of nodes in distance k from the root. Then

$$L(k) = 4L((k-3)/2)$$

[5] A *unit disk graph* is a graph where there is an edge between two nodes iff their Euclidean distance is at most one.

[6] The complete graph K_n is also a unit disk graph and experiences a lower bound of $n-2$, but from a practical point of view this graph is not interesting since each node can hear each other node and so a message can just be transmitted with maximal radio strength and is immediately received by the destination.Thus, the naming problem is not an issue at all.

and $L(1) = 2$. We now develop this equation recursively, resulting in

$$L(k) = 4^i L(\frac{k - \sum_{j=0}^{i-1} 3 \cdot 2^j}{2^i}) = 4^{\log((k+3)/4)} L(1)$$
$$= ((k+3)/4)^2 \cdot 2 = (k+3)^2/8.$$

Let $L'(k)$ be the number of nodes in distance $k-1$. This number can be computed as above, with the exception that $L'(1) = 1$. Hence we immediately get

$$L'(k) = ((k+3)/4)^2 \cdot 1.$$

Lemma 18. *In $T(k)$ there are at most $9 \cdot k^2$ nodes.*

Proof. Let $N(k)$ be the number of nodes in the tree $T(k)$. Then

$$N(k) < 4N((k-3)/2) + 2k$$

and $N(1) = 3$. As before we develop this equation recursively, and get

$$N(k) < 4^i N(\frac{k - \sum_{j=0}^{i-1} 3 \cdot 2^j}{2^i}) + \sum_{j=0}^{i-1} 2 \cdot 4^j \frac{k - \sum_{p=0}^{j-1} 3 \cdot 2^p}{2^j}$$
$$= 4^t N(1) + 2(k(2^t - 1) - (4^t - 1) + 3(2^t - 1)) \leq 9 \cdot k^2,$$

where we substituted t for $\log((k+3)/4)$ for the sake of readability.

Lemma 19 (min). *There are unit disk graphs where we need at least $((\sqrt{n}+9)/12)^2$ anchors to solve the naming problem.*

Proof. By Lemma 17 and the definition of the minimal coverage number in Section 6 the minimal coverage number of the constructed unit disk graph of depth k is $(k+3)^2/16$. Since by Lemma 18 the graph of depth k has at most $9k^2$ nodes, we can construct a unit disk graph on n nodes with depth at least $\sqrt{n}/3$. Substituting $\sqrt{n}/3$ for k we thus obtain a minimal coverage number of at least $((\sqrt{n}+9)/12)^2$. Following Lemma 12 this concludes the proof.

The locality-lemma is a direct implication of the lemma above.

Lemma 20 (local). *It is not possible to solve the naming problem in a unit disk tree locally.*

Lemma 21 (any). *If chosen arbitrarily, we need up to $n-1$ anchors to solve the naming problem in unit disk graphs.*

Proof. If we choose all nodes in $T(\sqrt{n}/3)$ except two sibling-leaves, we still cannot distinguish between those two siblings. Hence, we have to choose one of them to solve the naming problem and have in total $n-1$ anchors.

7.2 Routing

By the same discussion as in Section 6 the routing problem on the unit disk grid tree can be solved by choosing all leaves as anchors.

Theorem 22. *The pseudo-geometric routing problem on the unit disk grid tree can be solved with* $2 \cdot ((\sqrt{n}+9)/12)^2$ *anchors. Furthermore, the chosen route between source and destination is the shortest path.*

8 Butterfly

Due to the lack of space we omit the proofs for the lemmas in this section and refer the interested reader to the full paper [28].

8.1 Naming

We say that a node in the butterfly network is in column i if it is in the $(i+1)st$ column from the left (that is the leftmost column has index 0) and in row j if it is in the jth row from the top (that is the uppermost row has index 1) (see Figure 4). In a butterfly network on n nodes we have k rows and $(\log k + 1)$ columns, where $k(\log k + 1) = n$, that is $k = cn/\log n$, $c \leq 2$. An optimal choice of anchors, as shown in Lemma 23, is then to select all nodes in column 0 which are in row at most $k/2$ and all nodes in column $\log k$ which are in odd rows (see Figure 4, where the white nodes are anchors).

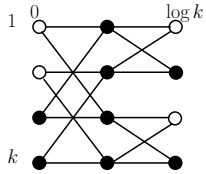


Fig. 4. k anchors solve the naming problem.

Lemma 23 (min). *In the butterfly network we need* k *anchors to solve the naming problem, where* $k(\log k + 1) = n$*.*

Lemma 24 (local). *It is not possible to solve the naming problem in the butterfly locally.*

Lemma 25 (any). *If chosen arbitrarily, we need* $n - 1$ *anchors in the butterfly network to solve the naming problem.*

8.2 Routing

Given that the anchors are chosen as described above we obtain following result.

Theorem 26. *The pseudo-geometric routing problem on the butterfly network can be solved with* k *anchors, where* $k(\log k + 1) = n$*.*

9 Hypercube

9.1 Naming

If we subsequently use the term coordinate we mean the classical hypercube coordinate, with one bit per dimension. For example in a 2-dimensional hypercube the nodes have classical coordinates $(0,0),(0,1),(1,0)$ and $(1,1)$. If we refer to the (pseudo) coordinate as obtained by the anchors, we use the term distance vector. Furthermore, with d we refer to the dimension of the hypercube, where $d = \log n$.

Lemma 27 (min). *To solve the naming problem in a d-dimensional hypercube one needs at least $\log n / \log\log n$ anchors.*

Proof. Each anchor a is able to subdivide the nodes in at most d classes, since the distance of a node to a is at most d. Thus, k anchors are able to differentiate between at most d^k nodes. Since we need to differentiate between all nodes it must hold that $d^k \geq n$ and we immediately get $k \geq \log n / \log\log n$.

Lemma 28 (any). *If chosen arbitrarily, we need up to $n/4+1$ anchors in the hypercube to solve the naming problem*

Proof. The goal is to make the points $u = (0,0,0,\ldots,0)$ and $v = (1,1,0,\ldots,0)$ indistinguishable for as many anchors as possible. Towards this goal we choose all anchors to be of the form $(1,0,0,\ldots,0) + x$ and $(0,1,0,\ldots,0) + x$, where x is a d-dimensional vector with first and second coordinate zero, and the other entries can be chosen arbitrarily. The distance of each anchor to u and v is thus $1+|x|$, where $|x|$ is the number of ones in the coordinate of x. The number of anchors we have chosen, without being able to distinguish u, v is $2^{d-2} = n/4$. Hence, to solve the naming problem we need at least one further anchor and the lemma follows.

We subsequently show how to choose d anchors which solve the naming problem locally.

Lemma 29. *Suppose each node with distance one to the origin is an anchor. Then for each node u in the hypercube it holds that its distance vector is composed of at most two different values.*

Proof. The coordinate of a node p in the hypercube can be expressed in the coordinates of the anchors $a_1,\ldots,a_d$, where a_i has a 1 in position i of its coordinate, and zeroes elsewhere: $p = t_1 \cdot a_1 + t_2 \cdot a_2 + \ldots + t_d \cdot a_d$, where $t_i \in \{0,1\}$. Let Δ_i be the distance of p to a_i. Then, $\Delta_i = t_1 + \ldots + t_{i-1} + \neg t_i + t_{i+1} + \ldots t_d$, where $\neg 0 \equiv 1$ and $\neg 1 \equiv 0$. We claim that out of any three distances between one point p and three anchors, at least two have to be equal. Without loss of generality (wlog) we consider the distances to the anchors a_1, a_2, a_3.

$$\begin{aligned}
\Delta_1 &= \neg t_1 + t_2 + t_3 + t_4 + \ldots + t_d \\
\Delta_2 &= t_1 + \neg t_2 + t_3 + t_4 + \ldots + t_d \\
\Delta_3 &= t_1 + t_2 + \neg t_3 + t_4 + \ldots + t_d.
\end{aligned}$$

Assume further wlog that $\Delta_1 \neq \Delta_2$. Thus, it has to hold that $t_1 \neq t_2$. We now show that Δ_3 has to be equal to either Δ_1 or Δ_2. First we assume that $\Delta_3 \neq \Delta_1$ and consequently $t_1 \neq t_3$. Then, $t_3 = t_2$ since $t_i \in \{0, 1\}$ and by assumption above $t_1 \neq t_2$. Therefore, $\Delta_2 = t_1 + \neg t_2 + t_3 + t_4 + \ldots + t_d = \Delta_3$. The same argument holds if $\Delta_3 \neq \Delta_2$ and in general for arbitrary three-tuples of distances and the lemma is proved.

Lemma 30 (local). *If each node with distance one to the origin is an anchor, all nodes in the hypercube can obtain unambiguously their coordinate as a function of their distance vector.*

Proof. The coordinate of node p in the hypercube can be expressed in the coordinates of the anchors $a_1, \ldots, a_d$: $p = t_1 \cdot a_1 + t_2 \cdot a_2 + \ldots + t_d \cdot a_d$, where $t_i \in \{0, 1\}$. By Lemma 30 each distance vector is comprised of at most two values x, y, where we assume wlog that $x < y$. We show how the coordinate of p is derived by its distance vector. Start with Δ_1 and Δ_2. There are two cases, either $\Delta_1 = \Delta_2$ or $\Delta_1 \neq \Delta_2$. In the first case, $\neg t_1 + t_2 + t_3 + \ldots + t_d = t_1 + \neg t_2 + t_3 + \ldots + t_d$, and hence $\neg t_1 + t_2 = t_1 + \neg t_2$, which means that $t_1 = t_2$. If on the other hand $\Delta_1 \neq \Delta_2$, we get $\neg t_1 + t_2 \neq t_1 + \neg t_2$. We can distinguish further whether $\Delta_1 < \Delta_2$ or not and get $\neg t_1 + t_2 < t_1 + \neg t_2$. if $\Delta_1 < \Delta_2$ or $\neg t_1 + t_2 > t_1 + \neg t_2$. if $\Delta_1 > \Delta_2$. In the first case the only feasible solution is $t_1 = 1, t_2 = 0$ in the second one $t_1 = 0, t_2 = 1$. The equations above hold for each pair of distances and hence we can deduce that if the distance vector is comprised of *exactly two different* values, then mapping the smaller value to one and the larger value to zero gives the exact coordinate of the node. The mapping is unambiguous, since there is only one feasible solution to the inequalities. If on the other hand the distance vector is comprised of *exactly one* value, we merely obtain the relation $t_1 = t_2 = \ldots = t_d$. Therefore, we basically have two possibilities of mapping the nodes: Either we map to the origin or to the coordinate $(1, 1, \ldots, 1)$. Since the distance vector of the origin contains only 1s, whereas the distance vector of the point $(1, 1, \ldots, 1)$ contains only ds, we can additionally distinguish between those two points and map $(1, 1, \ldots, 1)$ to $(0, 0, \ldots, 0)$ and $(d, d, \ldots, d)$ to $(1, 1, \ldots, 1)$ and the lemma is proved.

9.2 Routing

Assume now that each node with distance one to the origin is an anchor as proposed in Lemma 30. Based on its own, its neighbors' and the destination's distance vector each node u can compute its own, its neighbors' and the destination's coordinate. If u passes the message to the neighbor v with the smallest Hamming distance to the destination's coordinate, the message always reaches the destination on a shortest path.

Theorem 31. *The pseudo-geometric routing problem on the hypercube can be solved (locally) with* $\log n$ *anchors. Furthermore, the chosen route between source and destination is a shortest path.*

10 Conclusions

In this paper we proposed a new routing algorithm called pseudo-geometric routing, and analyzed it for the usual suspects of network topologies. We believe that pseudo-

geometric routing may evolve into a promising new routing paradigm, for highly dynamic real world networks with memory constraints.

In the table below we summarize the results of the previous sections. For each topology we give the lower and upper bound, indicate by a checkmark whether a local solution exists and whether the routing scheme finds the shortest path between an arbitrary source and destination, or only an approximation.

	min	any	local	s.p.
Line	1	2	✓	✓
Ring	2	3	✓	✓
Grid	2	$(\sqrt{n}-1)^2+2$	–	✓
Tree	$mc(T)$	$n-1$	–	✓
UDG	$((\sqrt{n}+9)/12)^2$	$n-1$	–	✓
Butterfly	$\frac{n}{\log n}$	$n-1$	–	–
Hypercube	$\frac{\log n}{\log \log n}$	$n/4+1$	✓	✓

There are still a lot of questions open with regard to pseudo-geometric routing, or routing for dynamic networks in general. Most importantly, we plan to study the "real-world" behavior of our algorithm. In particular, the lower bound for unit disk graphs (which are a generally accepted model for all sorts of multihop radio networks) is rather discouraging. On the other hand, the result for highly regular unit disk graphs such as the grid, is encouraging. Real sensor networks will be somewhere in-between general unit disk graphs and a grid. In this context, an in-depth analysis of mobility vs. updates is also a direction of future research.

References

[1] I. Abraham, D.Dolev, and D. Malkhi. LLS: a Locality Aware Location Service for Mobile Ad hoc Networks. *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2004.

[2] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact Name Independent Routing with Minimum Stretch. *Proc. of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2004.

[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad hoc Wireless Networks. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 1999.

[4] C.Busch, S. Surapaneni, and S. Tirthapura. Analysis of Link Reversal Routing Algorithms for Mobile Ad hoc Networks. In *SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 210–219. ACM Press, 2003.

[5] L. Cowen. Compact Routing with Minimum Stretch. In *Proc. of the ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 1999.

[6] L. Doherty, L. E. Ghaoui, and K. Pister. Convex Position Estimation in Wireless Sensor Networks. In *Proc. of Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[7] T. Eilam, S. Moran, and S. Zaks. A Simple DFS-Based Algorithm for Linear Interval Routing. In *WDAG '97: Proceedings of the 11th International Workshop on Distributed Algorithms*, 1997.

[8] K. Fath, P. Flocchini, and S. Pierre. A Compact Routing Technique for Communication Networks . *IEEE Canadian Conference on Electrical and Computer Engineering*, 1999.

[9] P. Fraigniaud and C. Gavoille. Interval Routing Schemes. *Algorithmica*, 21(2):155–182, 1998.

[10] P. Fraigniaud, C. Gavoille, and B. Mans. Interval Routing Schemes Allow Broadcasting with Linear Message-complexity. *Proc. of Symp. on Principles of Distributed Computing (PODC)*.

[11] E. M. Gafni and D. P. Bertsekas. Distributed algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, 29:11–18, 1981.

[12] C. Gavoille. A Survey on Interval Routing. *Theoretical Computer Science*, 245(2):217–253, 2000.

[13] C. Gavoille. Routing in Distributed Networks: Overview and Open Problems. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 32, 2001.

[14] C. Gavoille and M. Gengler. Space-Efficiency for Routing Schemes of Stretch Factor Three. *Journal of Parallel and Distributed Computing*, 61(5):679–687, 2001.

[15] C. Gavoille and D. Peleg. Compact and Localized Distributed Data Structures. *Journal of Distributed Computing*, 16:111–120.

[16] L. Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks. *Computer Networks*, 1:155 – 174, 1975.

[17] E. Kranakis, H. Singh, and J. Urrutia. Compass Routing on Geometric Networks. *11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.

[18] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit Disk Graph Approximation. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2004.

[19] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proc. of Symp. on Principles of Distributed Computing (PODC)*, 2003.

[20] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual Coordinates for Ad hoc and Sensor Networks. *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2004.

[21] D. Peleg and E. Upfal. A Tradeoff between Space and Efficiency forRouting Tables. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, 1988.

[22] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, and I. Stoica. Geographic Routing without Location Information. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2003.

[23] N. Santoro and R. Khatib. Labelling and Implicit Routing in Networks. *Comput. J.*, 28(1):5–8, 1985.

[24] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from Mere Connectivity. In *Proc. of Intl. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[25] H. Takagi and L. Kleinrock. Optimal Transmission Ranges for Randomly dDistributed Packet Radio Terminals. *IEEE Transactions on Communications*, 32:246–257, 1984.

[26] M. Thorup and U.Zwick. Compact Routing Schemes. *Proc. of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2001.

[27] J. van Leeuwen and R. Tan. Interval Routing. *The Computer Journal*, 30:298 – 307, 1987.

[28] M. Wattenhofer, R. Wattenhofer, and P. Widmayer. Geometric routing without geometry. Technical report, ETH Zurich, 2005.