Hyperbolic Embedding and Routing for Dynamic Graphs

Andrej Cvetkovski
Department of Computer Science
Boston University
Email: andrej@cs.bu.edu

Mark Crovella
Department of Computer Science
Boston University
Email: crovella@cs.bu.edu

***Abstract*—We propose a scalable scheme for routing in arbitrary network connectivity graphs, based on greedy routing and utilizing virtual node coordinates. In dynamic multihop packet-switching communication networks, routing elements can join or leave during network operation or exhibit intermittent failures. We present an algorithm for online greedy graph embedding in the hyperbolic plane that enables incremental embedding of network nodes as they join the network, without disturbing the global embedding. Even a single link or node removal may invalidate the greedy routing success guarantees in network embeddings based on an embedded spanning tree subgraph. As an alternative to frequent reembedding of temporally dynamic network graphs in order to retain the greedy embedding property, we propose a simple but robust generalization of greedy distance routing called Gravity–Pressure (GP) routing. Our routing method always succeeds in finding a route to the destination provided that a path exists, even if a significant fraction of links or nodes is removed subsequent to the embedding. GP routing does not require precomputation or maintenance of special spanning subgraphs and, as demonstrated by our numerical evaluation, is particularly suitable for operation in tandem with our proposed algorithm for online graph embedding.**

I. Introduction

The applicability of the greedy geometric packet routing paradigm to large internetworks was studied in details by Finn [1] as an alternative to the classical routing table approach. According to this type of addressing and routing scheme, each router in the network is assigned a coordinate denoting its location – a process referred to as *network embedding* – and the shortest path geometric distances between the nodes can be calculated based on these coordinates. Each packet is then forwarded toward the destination, choosing intermediate routing elements that provide most progress toward the destination. This makes the straight-line distance from the packet to the destination node a monotonically decreasing function throughout the journey, thus eventually reaching zero – when the packet arrives at the destination.

The simplest deterministic greedy routing rule picks as a next hop a directly connected neighboring node that would decrease the distance of the packet to the destination the most. The notable advantages of greedy routing – its small computational complexity, the small memory requirement per node, and the use of local information only (each node finds a next hop based on the coordinates of its neighbors), make this scheme suitable for distributed operation in large scale networks.

Greedy geometric routing using the actual *physical* coordinates of the nodes has been studied due to its simplicity and scalability ([1]–[3]), and regained popularity in the research community recently with the proliferation of GPS-capable communication devices ([4], [5]). Greedy routing based on node locations and Euclidean distances has been shown to have a high rate of success, but fails when a packet reaches a node that is closer to the destination than all of its direct neighbors even if a path to the destination exists (e.g. [6]).

Changing the actual physical locations of nodes to address this issue and improve the success rate of a greedy routing scheme is perhaps a rather impractical idea, but the observation that the distance-to-destination function of each path in an embedded graph is determined by the embedding motivated the quest for network embeddings that would support a 100% successful greedy routing, even if nodes have to be assigned artificial, *virtual* coordinates that do not necessarily represent locations in physical space ([4], [5]). Limiting this quest to true metric spaces, R. Kleinberg, in his groundbreaking work [7] presented a constructive proof that every finite, connected, undirected graph has a *greedy embedding* in two-dimensional hyperbolic space – that is, an embedding that allows greedy routing for all source-destination pairs.

The embedding in [7] uses the hyperbolic plane as an underlying space, and the corresponding distance function is the standard hyperbolic distance (see e.g. [8]). The generic algorithm of [7] can find a greedy embedding of an infinite d-regular tree for any integer $d \geq 3$. To embed an actual graph G, first a spanning tree T of G is chosen to serve as a minimal, loop-free subgraph that spans all the vertices and provides a unique simple path between any two of them. Subsequently, the maximal degree d of T is determined. Finally, the nodes of T are identified with the embedded nodes of the d-regular tree, as obtained by the embedding algorithm. This completes the greedy embedding of the tree T. It is easy to see that this embedding is also a greedy embedding of the graph G.

Although the procedure described in [7] is aimed at ad-hoc wireless networks and sensornets, there seem to be some obstacles to successful application of this type of embedding to communication networks whose topology can change over

time. In [7], the entire embedding is a function of a local topological property of the graph, namely the maximum degree of the chosen spanning tree T. Since newly added nodes increase the node degree locally, their embedding is not always possible without changing the coordinates of all nodes in the network. Further, the greediness of the embedding in [7] depends critically on the connectivity provided by the underlying embedded spanning tree. This implies that local changes in connectivity caused by nodes leaving the network or failing links, can invalidate the greedy property of the entire embedding. Such properties are undesirable of embedding algorithms intended for distributed operation.

Our goal in this paper is to address the above issues by (i) constructing a greedy graph embedding that supports addition of an arbitrary number of nodes to the embedding in an online fashion while requiring *no changes* to the previously assigned node coordinates in order to retain the greedy property; and (ii) constructing a greedy routing procedure that guarantees delivery even in the presence of *disturbances of the greedy property* of the embedding caused by nodes and/or links failing unexpectedly or exhibiting intermittent periods of downtime or standby-time.

Toward this end, in this paper we present an algorithm for online calculation of a greedy embedding in the hyperbolic plane for a given arbitrary, connected graph with edges representing the two-way connectivity in a communication network. Our algorithm supports *incremental* embedding of network nodes as they join the network during network operation time, without affecting the rest of the embedding.

In greedy embeddings with guarantees based on the existence of "at least one" greedy next hop at each node, even a single node or link failure may invalidate the greediness of the embedding, thus causing the need to re-embed the entire network if the greedy property is to be reestablished. As an alternative to frequent reembedding of network graphs due to intermittent node or link failures, or nodes leaving the network, in this paper we propose a simple but robust greedy routing method called *Gravity–Pressure* (GP) *routing*. Our routing algorithm can be viewed as a generalization of the simplest greedy distance routing, and always succeeds in finding a route to the destination if a path in the network exists. Since no assumptions are made about the type of the network coordinates, GP routing can be applied to embedded networks using physical coordinates as well as virtual coordinates in Euclidean or hyperbolic space. However, as the results of our experimental study show, GP routing is particularly suitable for application in graphs embedded using the online embedding procedure described in this paper. For its operation, GP routing does not require precomputation or maintenance of special spanning subgraphs.

The rest of this paper is organized as follows. In Section II-A we formulate a sufficient condition for a graph embedding to be greedy. Based on this formulation, we present our online embedding algorithm in Section II-B and further discuss its construction and properties in Section II-C. Section III-A offers an intuitive overview of the Gravity–Pressure routing, and a precise algorithm statement is given in Section III-B. Section IV presents a brief experimental evaluation of the overall proposed routing and addressing scheme. Concluding remarks and future considerations are given in Section VI.

II. Online Greedy Embedding

A. Preliminaries

We start by considering graph embeddings in a d-dimensional Euclidean or hyperbolic space.

Definition 1: Given a connected finite graph G with vertex set V, an *embedding* of G in $\mathbb{R}^d$ resp. $\mathbb{H}^d$ is a mapping $C(G): V \to \mathbb{R}^d$ resp. $C(G): V \to \mathbb{H}^d$ that assigns to each vertex $v \in V$ a virtual coordinate $C(v)$.

Definition 2: For two points $v, w \in \mathbb{R}^d$ resp. $\mathbb{H}^d$, the *Euclidean* resp. *hyperbolic bisector* of the Euclidean resp. hyperbolic line segment determined by v and w is the locus of points in $\mathbb{R}^d$ resp. $\mathbb{H}^d$ equidistant from v and w in terms of Euclidean resp. hyperbolic distance.

In $\mathbb{R}^d$, the bisector is the Euclidean hyperplane perpendicular to the segment $[v,w]$ at its midpoint. In $\mathbb{H}^d$, the bisector is the hyperbolic hyperplane perpendicular at the segment's midpoint to the hyperbolic line segment joining v and w.

Lemma 1: Let X be either $\mathbb{R}^d$ or $\mathbb{H}^d$ and ρ be the corresponding distance function. Let v and w be different points in X and let b be the bisector of the segment joining v and w. Then for all $u \in X$ it holds that $\rho(v,u) < \rho(w,u)$ if and only if v and u are in the same half-space with respect to the bisector b.

Proof. Follows from the triangle inequality applied to the triangle determined by v, u and x, where x is the intersection of b and the segment joining u and w. Namely, $\rho(v,u) < \rho(v,x) + \rho(u,x)$. From the definition of the bisector, $\rho(v,x) = \rho(w,x)$, and from the definition of x, $\rho(u,x) + \rho(x,w) = \rho(u,w)$. Combining these yields $\rho(v,u) < \rho(w,u)$. □

Likewise, for all $u \in X$ it holds that $\rho(w,u) < \rho(v,u)$ if and only if w and u are in the same half-space with respect to b.

Definition 3: For a graph $G(V,E)$ and its embedding $C(G)$ in $\mathbb{R}^d$ resp. $\mathbb{H}^d$, let $e \in E$ be an edge connecting the vertices u and v. An *embedded edge of* G is the Euclidean resp. hyperbolic line segment $C(e) = C(u,v)$ joining the points $C(u)$ and $C(v)$ in $\mathbb{R}^d$ resp. $\mathbb{H}^d$.

Lemma 2 (Greedy Embedding): For a graph G with embedding $C(G)$, let T be a spanning tree of G. For each edge $e \in T$, let $b(e)$ be the perpendicular bisector of the embedded edge $C(e)$. Then a sufficient condition for C to be a greedy embedding of G is that for each $e \in T$, $b(e)$ intersects no embedded edges of T other than $C(e)$.

Proof. Consider any edge $(u,v) \in T$. To prove the lemma, it suffices to show that u has a greedy route to any node s for which the path in the tree from u has next hop v. Consider the bisector b of (u,v). Since b intersects no other edges of T, s must be in the half-space of v with respect to b (cf. Lemma

1). Therefore $\rho(v,s) < \rho(u,s)$, so u has a greedy next hop to s, namely v. Applying this argument to each edge on the path from u to s confirms that the route in T (and thus in G) from u to s has a monotonically decreasing distance to the destination i.e. is a greedy route. Consequently, C is a greedy embedding of G. □

The rest of this section concentrates on *two-dimensional* hyperbolic space and systematizes the concepts from hyperbolic geometry that will be used in the subsequent presentation.

The Poincaré Disk model will be used throughout for visualization purposes. That is, we will use complex numbers from the set $\mathbb{D} = \{z \in \mathbb{C} \mid |z| < 1\}$ to represent the virtual coordinates of the embedded vertices in the hyperbolic plane. An introductory characterization of the more common hyperbolic geometry models and their elementary geometric objects can be found e.g. in [8].

As a distance function for the greedy embeddings considered in this section, we use the standard hyperbolic distance ρ for the Poincaré Disk model: $\forall z_1, z_2 \in \mathbb{D}$,

$$\cosh\rho(z_1,z_2) = \frac{2|z_1 - z_2|^2}{(1-|z_1|^2)(1-|z_2|^2)} + 1. \tag{1}$$

The existence of a closed-form expression for the hyperbolic distance in $\mathbb{D}$ makes the choice of this model suitable for the implementation of greedy embedding algorithms. The element of hyperbolic length

$$\frac{2|dz|}{1-|z|^2} \tag{2}$$

associated with this distance has circular symmetry: all points on a Euclidean circle in $\mathbb{D}$ centered at the origin have same distortion of the Euclidean element of length $|dz|$.

The Euclidean circle $\partial\mathbb{D} = \{z \in \mathbb{C} \mid |z| = 1\}$ represents the boundary at infinity of this model. We also refer to this circle as the horizon and to its points as the points at infinity of $\mathbb{D}$ or ideal points.

In hyperbolic geometry, the path that realizes the hyperbolic distance between two points (i.e. the shortest path) is the hyperbolic line or geodesic. In the Poincaré Disk model, paths realizing (1) are represented by arcs of Euclidean circles in $\mathbb{D}$ that are perpendicular to $\partial\mathbb{D}$. Two distinct points on $\partial\mathbb{D}$ thus determine a hyperbolic line in $\mathbb{D}$. For a hyperbolic line in $\mathbb{D}$ determined by two ideal points, of interest in this work are the center and the radius of the Euclidean circle in the Riemannian sphere $\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ containing the line. It is easy to show that given two ideal points $a = e^{i\alpha}$ and $b = e^{i\beta}$, the center of the Euclidean circle in $\overline{\mathbb{C}}$ containing the hyperbolic line whose endpoints at infinity are a and b, and the corresponding radius are given by

$$c = 1/m^*, \;\; R^2 = 1/|m|^2 - 1 \tag{3}$$

where $m = (a+b)/2$ is the midpoint of the Euclidean chord joining a and b, and m^* is the complex conjugate of m.

Two hyperbolic lines disjoint in $\mathbb{D}$ are said to be *parallel*. Specifically, parallel hyperbolic lines in $\mathbb{D}$ contained in disjoint Euclidean circles in $\overline{\mathbb{C}}$ are termed *ultraparallel*, to be distinguished from parallel hyperbolic lines that share an endpoint at infinity.

B. Online Greedy Embedding

In this section, we present our algorithm for computation of a greedy embedding of a given graph. The algorithm takes as input a connected graph $G = (V,E)$ specified by a set of vertices V and the connections between them $E = \{(u,v) \mid u,v \in V\}$. The graph G serves as an abstraction of a communication network – the nodes in the network correspond to the vertices of the graph and two vertices in the graph are connected if and only if the corresponding nodes can exchange data bidirectionally. The neighbors of a vertex $v \in V$ are the directly connected vertices: $N_v = \{u \mid (u,v) \in E\}$.

As noted in Sec. I, a greedy embedding of a tree graph T which spans a given connected graph G is also a greedy embedding of the graph G. Thus as a first step, the network constructs a spanning tree T of the graph G. Any type of a spanning tree can be used. A type of tree suitable for distributed construction is the minimal-depth tree. To form a minimal-depth tree, first the network nodes elect a root node. Subsequently, each node n elects from N_n its parent node to be the node that has the smallest distance in hops to the root node. Except for the root node r, each node in G is thus assumed to have identified a parent node for itself. The parent of a node n is referred to as p_n.

Fig. 1 contains a precise statement of the online embedding algorithm.

Procedure Online Embedding $C(G)$

1) Initialize by assigning to the root node r of the tree: (i) a virtual coordinate $C(r)$ in the hyperbolic plane; and (ii) the angles $\alpha_r = \pi$ and $\beta_r = 2\pi$ corresponding to the ideal points $a_r = e^{i\alpha_r}$ and $b_r = e^{i\beta_r}$.
2) For each node $n \in G$:
   a) Its parent p_n: (i) sends $C(p_n)$, $\alpha_n = \alpha_{p_n}$ and $\beta_n = (\alpha_{p_n} + \beta_{p_n})/2$ to n; and (ii) updates $\alpha_{p_n} := \beta_n$.
   b) Node n: (i) calculates c and R according to (3) with $a_n = e^{i\alpha_n}$ and $b_n = e^{i\beta_n}$ and its own coordinate

$$C(n) = \frac{R^2}{(C(p_n))^* - c^*} + c \tag{4}$$

   and (ii) updates $\alpha_n := (\alpha_n + \beta_n)/2$.

Fig. 1. Online embedding procedure

In the initialization step of the algorithm, the values of α_r and β_r determine the possible choices for the root location $C(r)$. $C(r)$ is chosen from the interior of the hyperbolic triangle OAB defined by the geodesic G_1, its bisector OB, and the ray OA defined by β_r as shown in Fig. 2. With this choice of initial conditions, the assignment of virtual coordinates to the vertices of the spanning tree T (and thus the graph G)

obtained using the procedure in Fig. 1, corresponds to a greedy embedding. We formalize this claim in the following

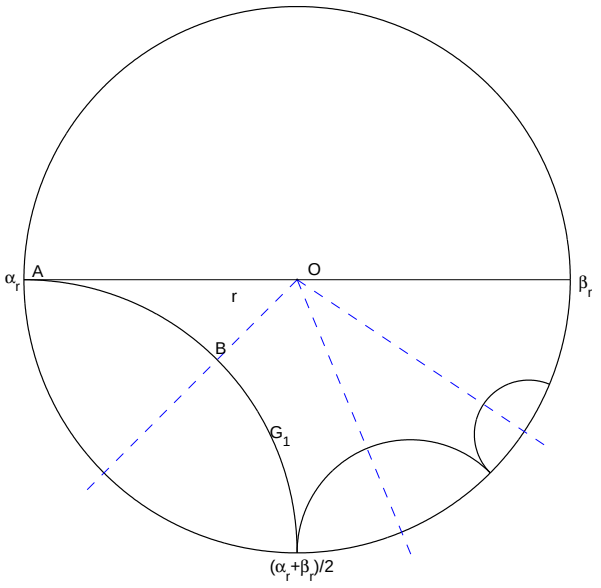

Fig. 2. Positioning of the root node for a greedy embedding

Proposition 1 (Correctness): If $C(r)$ is an interior point of the hyperbolic triangle OAB as in Fig. 2, then the embedding $C(G)$ obtained with the online embedding algorithm for an arbitrary graph G with a spanning tree T is a greedy embedding.

Proof. According to the greedy embedding lemma from Sec. II-A, it suffices to show that no bisector of an edge $e \in T$ embedded in the hyperbolic plane intersects other edges of the embedded tree. We begin by observing several properties of the online embedding procedure above.

For a node $n \in T$, let G_n be the hyperbolic line in $\mathbb{D}$ associated with n, whose endpoints at infinity are a_n and b_n as in step 2a of the algorithm, and denote by H_n the corresponding region of $\mathbb{D}$ bounded by G_n and containing the point $C(n)$. The virtual coordinate of the node n obtained via (4) is the reflection of the location of the parent node $C(p_n)$ in the hyperbolic line G_n. Therefore, the hyperbolic line segment joining $C(p_n)$ and $C(n)$ is the embedded edge $C(p_n,n)$ of T and that G_n is its perpendicular bisector. To see this, pick an isometry transform on $\mathbb{D}$ that maps the endpoints of the segment of the embedded edge to a point p on the imaginary axis in $\mathbb{D}$ and its complex conjugate p^* while mapping the intersection of G_n and $C(p_n,n)$ to the origin. Since the isometries on $\mathbb{D}$ are conformal, it is easy to see that under the chosen transform, the image of the Euclidean circle in $\overline{\mathbb{C}}$ containing G_n maps to the extended real axis $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. From the symmetry of the hyperbolic length element (2), $\overline{\mathbb{R}}$ is the perpendicular hyperbolic bisector of the hyperbolic line segment joining p and p^* and consequently, G_n is the bisector of the embedded edge $C(p_n,n)$ as desired.

It remains to show that for any node $n \in T$, G_n intersects no embedded edges of T other than $C(p_n,n)$. We observe that a point in $\mathbb{D}$, its reflection in a hyperbolic line, and the center of the Euclidean circle containing the hyperbolic line are collinear in the Euclidean sense. Therefore a point p in $\mathbb{D}$ and its reflection from a hyperbolic line G_n always lie in the same half of the subspace H_n with respect to the Euclidean bisector $b(G_n)$ of the arc in $\mathbb{D}$ containing G_n. Since a node n and the hyperbolic line G_c associated with a child node of n c are by construction contained in opposite halves of H_n with respect to $b(G_n)$, it follows that the embedded edge $C(p_n,n)$ and G_c are disjoint. Finally, note that by construction, for any node $n \in T$, the hyperbolic line containing the embedded edge $C(p_n,n)$ is ultraparallel to the hyperbolic line associated with any sibling of n. Thus any embedded edge $C(p_n,n)$ is disjoint with the hyperbolic bisector of any other embedded edge of the tree T. Consequently, the embedding $C(G)$ is a greedy embedding. □

When a new node, say n, joins an already embedded graph, it can obtain a virtual coordinate simply by identifying a parent node for itself, say p_n, and executing step 2) of the algorithm in Fig. 1. Note that this method does not require changes to the virtual coordinates of the existing nodes when a new node enters the graph. This is possible since our algorithm allows allocation of disjoint subspaces of the hyperbolic plane in an online fashion. By construction, the number of child-nodes any node can have is not limited, and there is always free space to be allocated for a newly added node.

Figure 3 illustrates an example of a graph embedded in the Poincaré Disk according to our online embedding procedure. The figure shows the embedded edges of a spanning tree of the graph; for clarity, the non-tree edges are not shown.

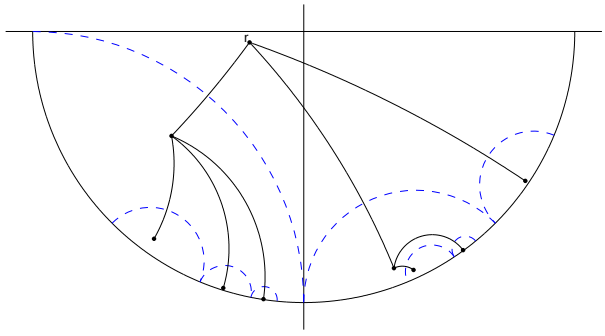

Fig. 3. Example of a greedy embedding of an irregular spanning tree in the Poincaré disk model

C. Remarks

All steps of the presented algorithm are suitable for distributed and asynchronous computation. Communication takes place only between a node joining the embedded graph and its parent node, which is elected from the immediate topographic neighborhood in the graph.

The online embedding algorithm presented in Sec. III-B generates node coordinates without use of any information

about physical locations of the nodes. The only initial virtual coordinate needed is the root coordinate, which can easily be chosen by the elected root node.

The region of allowable virtual coordinates for the root node in the initialization of the algorithm can be derived from the requirement in the proof of Proposition 1 that the point in $\mathbb{D}$ representing any embedded node n and the associated hyperbolic line of any child-node of n be at the opposite sides of the Euclidean bisector of the region H_n associated with n. This requirement ensures that no hyperbolic bisector associated with an embedded node intersects the embedded edge of its parent node. The allowable region for the root node is obtained as the intersection of the allowable regions with respect to the associated geodesics of all possible child-nodes of the root node:

$$J = \bigcap_{n=1}^{\infty} J_n.$$

It is easy to show that the region J corresponds to the hyperbolic triangle OAB whose vertices are the origin O, the ideal point A with coordinate $-e^{i\beta_r}$ and the midpoint B of the arc containing the geodesic G_1 associated with the first child-node of the root node. (See Fig. 2.) It can be shown that for this triangle to have a non-zero area, it is sufficient to choose values of α_r and β_r that satisfy $\beta_r - \alpha_r < 4\pi/3$. Fig. 3 illustrates the case when $\beta_r - \alpha_r = \pi$.

Satisfying the condition of the greedy embedding lemma in Sec. II-A is not the only way to achieve a greedy embedding, but is sufficient. We remark that the construction implied by the lemma is possible in the hyperbolic plane owing to the fact that parallelism is a less restrictive quality in hyperbolic space than in Euclidean space. More specifically, parallelism is not a transitive relation in hyperbolic space and allows every embedded edge to be parallel to the bisectors of all other embedded edges. This is not possible in Euclidean space without violating the condition of the greedy embedding lemma, but is easily done in hyperbolic space. The online embedding algorithm can thus embed an irregular tree directly rather than identifying a regular tree as a superset of nodes to be embedded in the hyperbolic plane.

III. The Gravity–Pressure Greedy Routing Algorithm

A. Overview

The choice of a spanning tree as a subgraph type to be used in the graph embedding procedure described in Sec. II is based on the fact that spanning trees have simple enough structure to allow incremental embedding, yet they contain a path between any two nodes in the original graph. Adding a new node to an existing spanning tree amounts to adding a single edge to the already embedded spanning subgraph, and the condition of Lemma 2 can be easily satisfied.

However, every spanning tree provides exactly one path for each pair of nodes in the graph. Thus removal of any graph edge that is a non-leaf tree edge in the embedded subtree, partitions the spanning tree into two unconnected subgraphs. Similarly, removal of any node from the original graph other than leaf nodes in the tree, partitions the spanning tree into a forest of d subtrees, where d is the node degree of the removed node, and thus disturbs the connectivity property of the tree. It is easy to construct examples of graphs where partitioning of the embedded spanning tree violates the greedy property of the embedding. In fact, we have produced a number of such embedded graphs for the purposes of Sec. IV of this paper.

To cope with greedy routing failures caused by local maxima of the packet progress toward the destination, one could reinitiate the network embedding procedure on demand, or use more sophisticated routing schemes that would either be able to avoid such local maxima, or to continue the routing after a data packet had reached a dead end. For the latter approach, numerous advanced routing and route discovery procedures have been proposed in the recent literature on location-based routing (see e.g. [9]). These procedures can be roughly divided into proactive, reactive, and hybrid, based on whether they precompute auxiliary data structures for possible use in finding a non-greedy route if a greedy route to the destination does not exist.

In real network environments, link and node failures are expected to happen often. Recent experimental studies have shown that most failures are temporary, and in fact very short-lived (e.g. [10]). In such conditions, repeating the embedding procedure to regain the greedy property, or precomputing data structures every time a network element or link becomes unavailable, may be unjustified from the standpoints of efficiency and conservation of resources. Instead, we propose a simple generalization of the greedy distance routing rule that does not require proactive computation or maintenance of special data structures for its operation, and as such, is suitable for application in temporally dynamic graphs. Our routing method, called Gravity–Pressure (GP) routing, always succeeds in finding a route to the destination, if a path in the network exists.

In the rest of this section we provide an intuitive overview of the GP routing procedure. A precise statement of the routing algorithm is postponed to Section III-B. We will discuss some of the advantages and disadvantages of GP routing when used in conjunction with the greedy embedding algorithm of Sec. II in more detail in Section V.

GP routing normally forwards packets to the neighbor that provides most progress toward the destination. By analogy with a liquid flowing through a system of pipes in gravitational field of spherical symmetry toward the center located at the destination node, we refer to this routing mode as the *gravity routing mode*. The packet may occasionally reach a local minimum, or a "valley". In that case, GP forwards the packet to a next hop that provides the least negative progress with respect to the location of the destination. To deal with the possibility of the packet entering a loop and periodically returning to the same local lowermost point, we introduce the

concept of pressure as a second "field" that helps steer the packet out of the valley. We refer to this routing mode as the *gravity–pressure routing mode*. We note that in contrast to other proposed routing procedures that switch to another routing mode when a packet reaches a dead-end, GP retains the locally greedy decision making process and thus can be viewed as a generalization of greedy distance routing as opposed to a hybrid, dual routing technique.

Fig. 4 illustrates the principle of the GP routing technique. We note that while this example uses physical Euclidean node coordinates and Euclidean distances to facilitate understanding, GP is by no means limited to this metric space. The packet starts from the source node in gravity mode, but reaches a dead end at node N_1. At this point, the packet enters gravity–pressure mode and the trajectory it follows subsequently is shown with a dashed line. The backpressure that helps the packet get out of the valley is realized by keeping track of the number of visits of each node until node N_2 is reached, which is closer to the destination than the node where a dead end was detected (N_1). At this point the packet switches back to greedy mode.

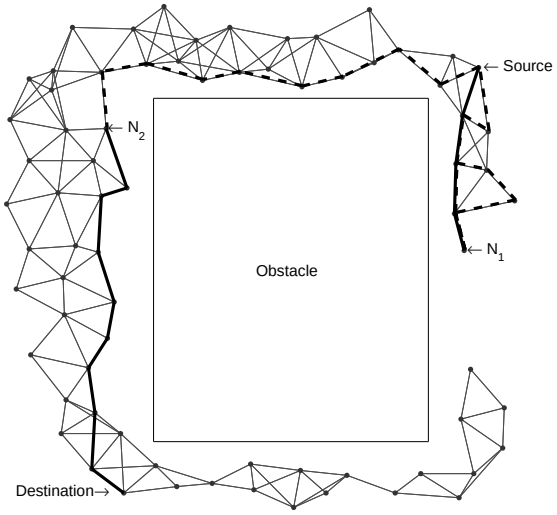

Fig. 4. An example route of Gravity–Pressure route discovery in an ad-hoc wireless network using physical node locations and Euclidean distance

B. Formal Statement of the GP Algorithm

Fig. 5 contains a precise statement of the Gravity–Pressure (GP) routing algorithm. An instance of this algorithm is assumed to run at each node in the network. At present, the network graph is assumed to be connected. The discussion of the extensions of the algorithm for handling partitioned graphs is relegated to Sec. II-C.

Each packet in the network is assumed to contain a flag bit, determining the current routing mode of the packet. When a packet is created, its packet mode flag is initially set to gravity mode, but can be toggled at each routing element between gravity and pressure mode, as described below. The originator of the packet, say N_{src}, is assumed to know the ID of the destination N_{dest} as well as its virtual coordinates, and these data are also included in the packet header.

When a packet arrives at a routing node, say N_i, it is either in gravity or in pressure mode. The preferred forwarding mode of the GP algorithm is the gravity mode. Thus, if the packet is in gravity mode, node N_i first tries to forward the packet to the next hop N_{next} in gravity mode (block $(b1)$ in Fig. 5), and uses pressure mode only if it is not possible to forward in gravity mode (block $(b2)$). On the other hand, if the arriving packet is in pressure mode, node N_i forwards in pressure mode (block $(b3)$), but only until the packet gets closer to the destination than the valley distance d_{v}, that is, the distance from the node where pressure mode was last set to the destination. Once the packet is closer to N_{dst} than d_{v}, the forwarding mode of the packet is changed back to gravity mode (block $(b4)$) and the packet is forwarded accordingly. All distances in Fig. 5 are calculated according to Eq. (1), using the virtual coordinates of the pair:

$$\text{dist}\,(N_1,N_2) = \rho\,(C\,(N_1)\,,C\,(N_2))\,. \tag{5}$$

In pressure mode, the next hop is also chosen greedily to be the node that makes most progress to the destination. However, in pressure mode, the next hop is chosen from the subset of neighbor nodes that share the lowest number of visits.

Proposition 2 (Correctness): The GP routing algorithm always succeeds in finding a path from the source to the destination node, if a path in the network exists.

Proof. The routing algorithm always finds a next hop. If the packet reaches a dead end in gravity mode, it enters pressure mode, and can only switch back to gravity mode if it gets closer to the destination than d_{v}. The sequence of valley distances d_{v} is thus monotonically decreasing and a packet cannot enter the same valley point in gravity mode more than once. (That is, the packet cannot "get stuck" at the same node more than once.) On the other hand, provided that there is a path to the destination, the packet cannot stay in pressure mode indefinitely – it will either go to gravity mode or reach the destination in pressure mode. Namely, assuming on the contrary, that the packet keeps looping in the network in pressure mode indefinitely without reaching the destination implies that the pressure on the set of nodes L that form the loop increases indefinitely. But since there is a path from any node of this loop to the destination, this implies that some node $n \in L$ has a neighbor with a constant pressure that is never chosen as a next hop in block $(b3)$ of the algorithm – a contradiction. □

The Visits vector is stored in the data packet and is implemented as a table containing the nodes that the packet visited and the corresponding numbers of visits. Regarding the storage of the Visits vector, we note that several optimizations are possible. First, only nodes visited in pressure mode need be kept in the table. Those nodes not found in the table are assumed to have 0 visits in $(b3)$. This reduces the space

Procedure Forward Packet (at node N_i)

On arrival of packet P at node N_i:
Initialize Visits=[];

If $N_i \neq N_{\text{dest}}$ {

Gravity:
If Pkt_mode = Gravity {
$N_{\text{next}} := \underset{M \in \text{Nbrs}(N_i)}{\text{argmin}}\ \text{dist}(M, N_{\text{dest}})$;
If $\text{dist}(N_{\text{next}}, N_{\text{dest}}) < \text{dist}(N_i, N_{\text{dest}})$ { $(b1)$
Forward_pkt_to(N_{next});
}
Else { $(b2)$
Pkt_mode := Pressure;
$d_{\text{v}} := \text{dist}(N_i, N_{\text{dest}})$;
$\text{Visits}(N_i) \mathrel{+}= 1$;
}
}

Pressure:
If Pkt_mode = Pressure {
If $\text{dist}(N_i, N_{\text{dest}}) \geq d_{\text{v}}$ { $(b3)$
$\text{Visits}_{\text{min}} := \underset{M \in \text{Nbrs}(N_i)}{\text{min}}\ \text{Visits}(M)$;
$\text{Candidates}(N_i) :=$
$\{M \in \text{Nbrs}(N_i) \mid \text{Visits}(M) = \text{Visits}_{\text{min}}\}$;
$N_{\text{next}} := \underset{M \in \text{Candidates}(N_i)}{\text{argmin}}\ \text{dist}(M, N_{\text{dest}})$;
$\text{Visits}(N_i) \mathrel{+}= 1$;
Forward_pkt_to(N_{next});
}
Else { $(b4)$
Pkt_mode := Gravity;
Goto Gravity;
}
}
}
Else If $N_i = N_{\text{dest}}$ { process_packet( ); }

Fig. 5. Packet forwarding procedure at node N_i

needed for storage of the Visits table. Second, data packets that require pressure mode can serve at the same time as route discovery packets for subsequent communication. Namely, the sequences of nodes that were visited in pressure mode suffice to reconstruct the route. Routing between these segments can be done in gravity mode. Finally, it is possible for a packet to return to a node visited earlier in pressure mode. In such case all nodes in the Visits table after the revisited node can be deleted. If this is done, then the discovered route is loop free. We implemented these optimizations for the purposes of the experimental evaluation presented next.

IV. Experimental Evaluation

In this section we briefly report the results of an experimental evaluation of the GP routing algorithm running on graphs embedded in the hyperbolic plane using the online embedding algorithm of Sec. II. For each source-destination pair, we use the relative path stretch metric, defined as the ratio of the hop length of the path found by GP routing to the corresponding shortest path in the graph.

GP routing can always find a route to the destination, but it is easy to contrive node coordinates, at least in the Euclidean plane, where GP routing would produce paths with rather unfavorable stretch. Further, since node failures impair the greedy property of the embedding, one might expect that GP stretch increases adversely with the increase of the number of nodes that failed since the graph was last embedded. The goal of the experiments presented here is to show that this is not the case when GP routing is used in conjunction with the virtual coordinates produced by the online hyperbolic embedding algorithm.

To examine the path stretch distribution and its scaling with the fraction of failed nodes in the graph, we have conducted a series of experiments using synthetic graphs of 50 nodes each, with randomly generated edges. The average node degree is 3. We embedded each graph in hyperbolic space using the algorithm of Fig. 1. For each generated graph we produced several versions with different fractions of randomly chosen failed nodes. For each such graph version, we exhaustively enumerated the routes found by GP routing for all possible source-destination pairs.

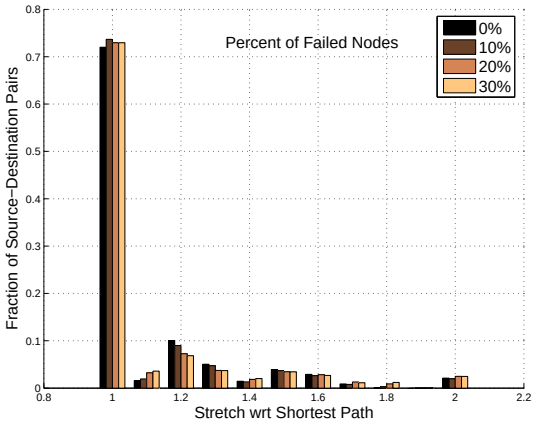

Fig. 6. Distribution of GP route stretch after a fraction of the nodes is removed. The network was initially embedded in a hyperbolic space

Figures 6 and 7 show the results of the path stretch distribution measurement. The results are averaged over 30 randomly generated graphs. For the greedy embedding (no failed nodes), 72% of the nodes had stretch <1.1 and 94% had stretch <1.5. The stretch distribution did not change significantly even for large fractions of failed nodes, such as 10, 20 and 30%. Figure 7 shows the stretch distribution *only for* those routes that needed to use the pressure mode of GP. 43% of those had stretch <1.1 and 80% had stretch <1.5.

Figures 8 and 9 show the results of the average stretch measurement. Each point is the average stretch for all possible source-destination pairs of 30 randomly generated graphs.

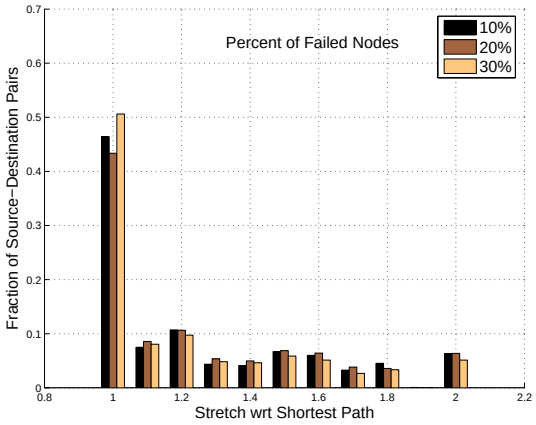

Fig. 7. Distribution of GP stretch, for source–destination pairs *requiring pressure mode routing,* vs. GP route stretch after a fraction of the nodes is removed. The network was initially embedded in a hyperbolic space

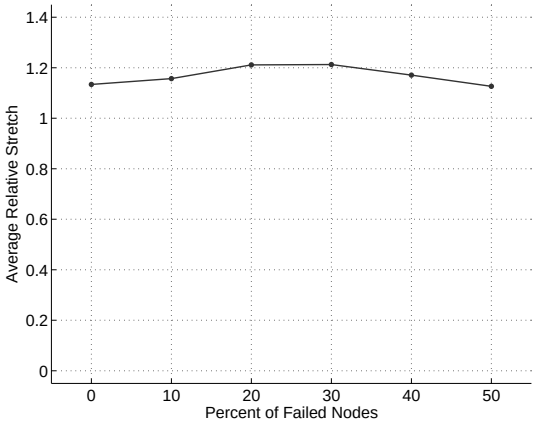

Fig. 9. Average path stretch of a network graph embedded in hyperbolic space vs. the percentage of nodes that was removed after the initial embedding in hyperbolic space

Figure 8 shows that the fraction of path grows significantly with the number of defunct nodes. From this, one would expect that the average stretch would grow similarly. However, Figure 9 shows, on the contrary, that the average stretch, after reaching a maximum of around 1.2, starts to decrease. This behavior is the result of the reduction of the node degree with the removal of nodes from the network. As the graph gets sparser, even though the number of valleys grows significantly (as implied by Fig. 8), the number of possible paths decreases and the packets in pressure mode are able to find their ways out of the valleys more efficiently.

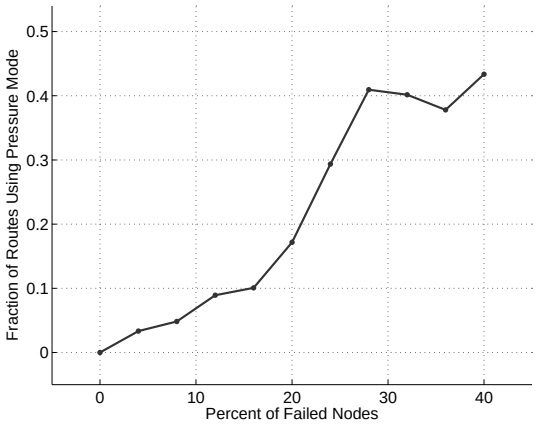

Fig. 8. Fraction of source-destination pairs using pressure mode routing vs. the percentage of nodes that was removed after the initial embedding in hyperbolic space

V. Discussion

The statement of the GP routing algorithm (Fig. 5) assumed that a path does exist between the source and the destination node. To avoid packets wandering in the network indefinitely in case a path does not exist, a hops-to-live parameter should be introduced in the packet header. Another possible limiting parameter is the maximum allowed distance from the packet to the destination. In case the distance from the packet to the destination exceeds this limit, the packet is dropped.

GP routing can be classified as reactive, on-demand routing technique. It does not require advance computation or maintenance of special spanning subgraphs or other structures for its correct operation. Each node needs to keep alive only the communication with its one-hop neighborhood in order to perform the routing function. GP routing requires no more information to be stored at each routing element than needed by the simplest greedy routing based on geometric distances. No assumptions are made about existence of super nodes in the network distinguished by larger energy supply, communication, or computational capabilities. These properties, combined with the notable simplicity of implementation, make GP routing suitable for distributed operation in networks containing a large number of nodes.

The GP routing algorithm is applicable to networks embedded in any metric space. Specifically, it can be used in Euclidean space, if the corresponding Euclidean distance function is used in (5) instead of (1). Further, GP routing can work with either physical or virtual node coordinates. Thus, it can be used in conjunction with any algorithm for network embedding and in networks that are not aware of their own physical locations. In this work, GP routing was applied to virtual coordinates generated by our online embedding algorithm. While optimality of stretch is not guaranteed, for the coordinates generated by the online embedding, we demonstrated experimentally that the relative path stretch is reasonably close to its optimal value of 1.

The embedding and routing algorithms presented in this paper do not make any restrictive assumptions about the graph type. This makes the algorithms usable not only for typical wireless graphs where connectivity is largely correlated with the geometric distance between the terminals, but also for general internetworks, where the topology is much less

predictable from the node positions. No requirements are made for a particular distribution of the nodes in physical space.

Several routing algorithms proposed in the research literature on position-based routing in ad-hoc networks utilize ideas related to the concept of pressure introduced in the GP routing algorithm in this paper. These ideas could perhaps be jointly termed as "cost-based void handling" [9]. [11] proposes virtual repositioning of network nodes embedded in an n-dimensional space, by adding an $(n+1)$-th "height" coordinate. Node height, like the pressure field in GP, is intended to help the routing algorithm steer the packets away from zones in the network where packets are likely to encounter local minima in the distance-to-destination function. However, the routing scheme in [11] requires several auxiliary algorithms to be executed proactively and periodically in order to maintain the height data structure. While the success rate of the routing algorithm is high, this scheme is not guaranteed to always find a route to the destination even when a physical path to the destination exists. In contrast, GP routing always find the route to the destination if a such exists, and does not need the help of any auxiliary algorithms once the network is embedded.

The PAGER-M [12] and DUA (distance upgrading) [13] aim to remove all dead-end parts of the network by virtually changing the existing node coordinates (as opposed to adding new dimensions, as in [11]) of the nodes where packets encounter a dead end. By contrast, a premise made in the GP routing paradigm is that one and the same node may be a dead end for some subset of destinations, while at the same time being a valid greedy hop for another subset. Therefore, in GP the pressure values are associated with the packets, and not with the network nodes. Also, while in [12] and [13] distances are upgraded greedily, GP's pressure values are always incremented conservatively, by a constant amount.

Finally, a loose analogy can be formed between GP's concept of pressure and the generalized node numbers introduced in [14].

VI. Conclusion

In this paper, we present an embedding and routing scheme for point-to-point geometric routing in arbitrary, internetwork graphs using generated, artificial node coordinates in the hyperbolic plane.

Desirable properties of network embedding and routing schemes are the ability to embed newly added nodes in an online fashion, without having to change the coordinates of previously embedded nodes, as well as the ability to provide routing success guarantees in embedded networks where nodes can join or leave during network runtime or can exhibit unscheduled downtime periods.

Our proposed embedding algorithm supports an arbitrary number of online node joins by providing incremental embedding that does not affect the rest of the embedding, and requires only local communication for its operation.

Our proposed routing algorithm, the Gravity–Pressure routing, provides guarantees of 100% routing success, even in the presence of a significant fraction of link or node failures or nodes leaving the network after the network embedding was completed. Unlike other position routing techniques for embedded graphs which include a separate routing "mode" for routing around local minima in the distance-to-destination function, the technique presented in this paper can be viewed as a generalization of the greedy principle, that always succeeds in finding a route to the destination if a path in the network exists. GP routing is stateless in the sense that each node participating in packet forwarding needs to be aware only of its one-hop neighboring nodes and their locations in order to perform the routing function. GP routing does not make any restrictive assumptions about network node capabilities, graph types, or coordinate types and can work with physical Euclidean coordinates as well as virtual node coordinates in any metric space. As the results of our experimental study show, GP routing is particularly suitable for application in graphs embedded using the online embedding procedure described in this paper.

References

[1] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," Univ. of Southern California, Information Sciences Institute, Tech. Rep. ISI/RR-87-180, March 1987.

[2] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, 1984.

[3] R. Nelson and L. Kleinrock, "The spatial capacity of a slotted aloha multihop packet radio network with capture," *IEEE Transactions on Communications*, vol. 32, no. 6, pp. 684–694, Jun 1984.

[4] M. Mauve, A. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *Network, IEEE*, vol. 15, no. 6, pp. 30–39, Nov/Dec 2001.

[5] S. Giordano, I. Stojmenovic, and L. Blazevic, "Position based routing algorithms for ad hoc networks: a taxonomy," in *Ad Hoc Wireless Networking*. Kluwer, 2003, pp. 103–136.

[6] B. Leong, "New techniques for geographic routing," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, May 2006.

[7] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proceedings of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2007)*, May 2007.

[8] J. W. Anderson, *Hyperbolic Geometry*, 2nd ed. Springer, 2007.

[9] D. Chen and P. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," *Communications Surveys and Tutorials, IEEE*, vol. 9, no. 1, pp. 50–67, Quarter 2007.

[10] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. ACM Sigcomm Internet Measurement Workshop*, Nov. 2002.

[11] N. Arad and Y. Shavitt, "Minimizing recovery state in geographic ad-hoc routing," in *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 13–24.

[12] L. Zou, M. Lu, and Z. Xiong, "A distributed algorithm for the dead end problem of location based routing in sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 54, no. 4, pp. 1509–1522, July 2005.

[13] S. Chen, G. Fan, and J.-H. Cui, "Avoid "void" in geographic routing for data aggregation in sensor networks," *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Special Issue on Wireless Sensor Networks*, vol. 2, no. 1, 2006.

[14] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *Communications, IEEE Transactions on*, vol. 29, no. 1, pp. 11–18, Jan 1981.