Optimizing Robustness in Geometric Routing Via Embedding Redundancy and Regeneration

Rein Houthooft, Sahel Sahhaf, Wouter Tavernier, Filip De Turck, Didier Colle, and Mario Pickavet
Department of Information Technology (INTEC), Ghent University– iMinds, Gaston Crommenlaan 8, 9050 Ghent, Belgium

Geometric routing is an alternative to traditional routing algorithms in which traffic is no longer forwarded using lookup tables, but using coordinates in an embedding of the underlying network. A major downside of current geometric routing algorithms is their inability to handle network failures in a graceful manner. Moreover, they cannot deal with dynamic graph topologies. This article presents a geometric routing scheme that uses an embedding based on a spanning forest. Allowing nodes to select the optimal spanning tree leads to both shorter paths and natural traffic redirection in case of network failures. By constructing the forest in such a way that its disconnected components have low redundancy, their coverage is maximized. Results show that this system is able to operate gracefully in severe failure scenarios, without any form of path protection or restoration. By means of an embedding regeneration procedure, the routing scheme is able to continuously adapt to an altering network topology. This geometric routing algorithm effectively combines two key objectives, namely low path stretch and high robustness.
NETWORKS, Vol. 000(00), 000–000 2015

Keywords: geometric routing; robustness; forest routing

1. INTRODUCTION

In geometric routing, nodes are matched with points in a particular mathematical space, a so-called graph embedding. By calculating distances within the embedding space, traffic is steered toward its destination along a distance-decreasing path. This avoids the need of large lookup tables, as is the case in traditional routing schemes. As in geometric routing a node only has to store local coordinates, rather than the full network state, it is scalable by its very nature. As such, it does not deal with computational overhead as a result of storing and inspecting lookup tables. A first disadvantage, however, is its lack of robustness because the underlying embedding can lose its greedy (also see Definition 2) characteristics when network failures occur. A second disadvantage is that the embedding cannot keep up with a dynamic graph topology. In both cases, it cannot be guaranteed that a distance-decreasing path toward the destination (also see Definition 1) exists. Figure 1 shows a routing void for a graph embedded in the 2-D Euclidean plane.

Originally, geometric routing was introduced as a way of routing in unit-disk graphs,[1] an accurate model for *ad hoc* and wireless sensor networks [1]. Recently, it has been shown that this style of routing can also be used in scale-free graphs,[2] which are a better model for wired networks such as the Internet [2]. In such large-scale communication networks traffic redirection in case of failures is essential. The question of how to combine low path stretch[3] with routing robustness toward failures and dynamic topologies, however, has yet to be solved.

This article extends our work [3, 4] by further exploring robustness in geometric routing. We present a geometric routing algorithm called Greedy Forest Routing (GFR) using an embedding based on a spanning forest. GFR is by its nature very robust toward network failures by maximizing the coverage of the embedding component, while attaining low path stretch. Furthermore, an embedding regeneration procedure has been designed to deal with a continuously changing network topology. Moreover, we show that adding a backup routing mechanism allows for 100% routing success rate, even in extreme network failure scenarios. Although we have designed GFR specifically for scale-free networks, it is general enough to be applied to other network topologies as well.

Received March 2015; accepted July 2015
Correspondence to: R. Houthooft; e-mail: rein.houthooft@ugent.be
Contract grant sponsor: Rein Houthooft is supported by a Ph.D. Fellowship of the Research Foundation – Flanders (FWO). European Commission through the EULER project (part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme [FP7]); Contract grant number: 258307.
Contract grant sponsor: UGent BOF/GOA project "Autonomic Networked Multimedia Systems."
DOI 10.1002/net.21630
Published online in Wiley Online Library (wileyonlinelibrary.com).

[1] A graph $G = (V, E)$ is a unit-disk graph when $\forall u, v \in V : v \in N(u) \Leftrightarrow \delta(u, v) \leq 1$ in case G is embedded into a Euclidean space, with δ the Euclidean distance.

[2] In scale-free networks the degree distribution follows a power-law $P(d) \sim d^{-\gamma}$ with a parameter $\gamma \in \mathbb{R}^+$ and d the vertex degree.

[3] The stretch of a path is its length, as a number of hops, divided by the shortest path length between its source and destination nodes.

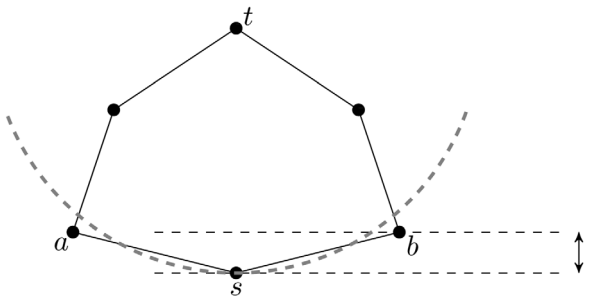


FIG. 1. Illustration of a routing void for a graph embedded into the 2-D Euclidean plane. The gray dashed arc, which represents points equidistant to t, indicates that both a and b are farther away from t than s itself. Hence, s cannot route to t along a distance-decreasing path using the 2-D Euclidean distance. The dashed horizontal lines show that using the 1-D Euclidean distance along the vertical axis allows s to route to t through either a or b, without encountering a local distance minimum.

2. RELATED WORK

Although most geometric routing algorithms target unit-disk graphs rather than scale-free graphs, their robustness techniques are still worth investigating. A basic concept of routing errors in geometric routing are so-called voids. These are nodes reached by a packet being forwarded for which no neighbor exists that has a lower distance to the destination than the node itself. Several techniques exists to avoid having to drop the packet at that specific point along the routing path. Many of them make use of a face routing-like construct [5]. But, these techniques require the underlying network to be a unit-disk graph and are hence not applicable to scale-free networks due to their different graph properties. Sahhaf et al. [6] propose a backup strategy for geometric routing based on a greedy hyperbolic embedding by searching alternative paths for fast traffic rerouting, applicable to a wide variety of networks. Their other work presents the use of a backup tree to deal with single failures [7]. Our work here differs from the former as it improves routing robustness by (i) allowing for natural traffic rerouting, by means of a multidimensional embedding, and (ii) using a spanning forest for which the tree redundancy has been heuristically minimized.

A different fault-tolerance strategy, not relying on predetermined alternative paths, is gravity-pressure routing [8]. Upon encountering routing voids, a potential function is applied to the routed packet. This causes the packet to be steered away from the void location. The main advantage of this technique is that its reliability can be proven. A severe disadvantage, however, is the increased stretch by steering the packet around the void. Other techniques focus on creating minimally overlapping trees, however, most are theoretical by nature and are hard to implement in a realistic distributed scenario, or are restricted by a fixed number of trees [9–11]. Moreover, these algorithms do not construct embeddings with the geometric routing paradigm in mind. Our approach uses a forest construction mechanism that reduces redundancy in a heuristic fashion. Herein, coordinates are assigned parallelly with as goal robust geometric routing. A recent work on geometric routing in complex networks is the Practical Isometric Embedding Protocol [12] in which Herzen et al. focus specifically on the scalability of their routing algorithm. Tang et al. [13] also use multiple trees for geometric routing, but they do not focus on, or test for, reliability. Moreover, multiple trees are constructed in a breadth-first manner rather than aiming for maximal coverage. Contrary to their approach, our method is able to combine high robustness with low path stretch.

3. THEORETICAL FOUNDATION

This section sets forth the theoretical foundation of geometric routing. First, we explain key concepts regarding graph embeddings. Next, we elaborate on fault-tolerance in geometric routing. Then we investigate how using a multi-embedding can increase routing robustness. Finally, we give a complexity analysis of the embeddings used in GFR.

3.1. Graph Embeddings

In this section a theoretical foundation for the GFR algorithm is built based on the work of Chávez et al. [14] and Korman et al. [15]. Geometric routing systems make use of a graph embedding. Such an embedding is a mapping between vertices of a graph and a particular mathematical space, formally defined by the following definition.

Definition 1. *Let S be a set and δ a metric function over S. Let $G = (V, E)$ be a graph, then an embedding of G into S is a mapping $f : V \to S$ such that $\forall u, v \in V : u \neq v \Leftrightarrow f(u) \neq f(v)$ [16].*

To guarantee a 100% routing delivery success rate a greedy graph embedding is required. This means that for every two non-neighbor vertices, there exists a third vertex for which the distance to either of these two former vertices is lower than the distance between them. This is formally defined by the following definition.

Definition 2. *A greedy embedding of a graph $G = (V, E)$ into a metric space (S, δ) is a mapping $f : V \to S$ with the following property: for every pair of distinct vertices $u, w \in V$ there exists a vertex $v \in V$ adjacent to u such that $\delta(f(v), f(w)) < \delta(f(u), f(w))$. [17]*

Herein, a metric space is the double formed by a space and a corresponding distance function, more formally defined by Definition 3. This metric space is used to steer packets toward their target by following a distance-decreasing path, as mentioned in the introduction.

Definition 3. *For a set S and a function $\delta : S \times S \to \mathbb{R}$ such that the following conditions hold $\forall u, v, w \in S$:*

1. $\delta(u, v) \geq 0 \wedge \delta(u, v) = 0 \Leftrightarrow u = v$
2. $\delta(u, v) = \delta(v, u)$
3. $\delta(u, w) \leq \delta(u, v) + \delta(v, w)$

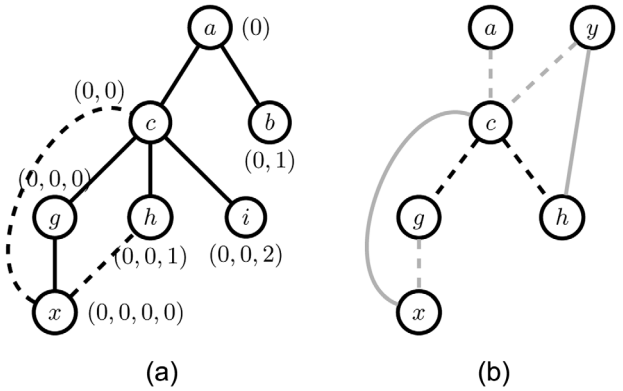


FIG. 2. Graph embeddings illustrated: Shortcuts, labeling, redundancy.

Then δ is called a metric and the double (S, δ) is called a metric space.

In this work, a spanning tree $T = (V, E')$ of the underlying network $G = (V, E)$ is used to generate a graph embedding. This embedding makes use of a vertex labeling procedure [15] and a distance function representing the shortest path length in T [14]. Herein, the embedding target space S is denoted as the *tree space* $\mathbb{T}$. This tree space is defined as

$$\mathbb{T} = \bigcup_{n \in \mathbb{N}} ((0)^\frown \mathbb{N}^n), \tag{1}$$

in which the function $(^\frown) : \mathbb{N}^m \times \mathbb{N}^n \to \mathbb{N}^{m+n}$ represents the concatenation of two tuples. The labels assigned by the labeling procedure (see Section 5) are now interpreted as coordinates in $\mathbb{T}$. These coordinates form a greedy embedding [14] which is denoted as $\mathcal{T}$. Therefore each vertex $v \in V$ corresponds to a point in $\mathbb{T}$ identified by the coordinates $\mathcal{T}(v)$.[4] An illustration of this embedding can be found in Figure 2a. The distance function δ is defined as

$$\delta(u, v) = |u| + |v| - 2|\phi(u, v)|, \tag{2}$$

with $\phi : \mathbb{N}^n \times \mathbb{N}^m \to \mathbb{N}^*$ a function that generates the largest common prefix of two tuples; $|u|$ represents the length of the coordinate tuple of vertex u. This results in a distance function $\delta : \mathbb{T} \times \mathbb{T} \to \mathbb{R}^+$ that, together with the space $\mathbb{T}$, forms the metric space $(\mathbb{T}, \delta)$ (the properties of a metric space can easily be proven). This metric space can now be used according to the principles of geometric routing. This means that every vertex is aware of the coordinates of its neighborhood and the destination coordinates are encoded in the packet header. As such, each node tries to forward a packet along a distance-decreasing path to the destination based on this header.

[4] Instead of writing $\mathcal{T}(u)$ we will make no distinction between the point to which a vector is mapped in its embedding space and the vertex itself. The exact meaning should be clear from the context.

3.2. *Fault-Tolerance*

In this section, we analyze fault-tolerance in geometric routing, the capability of handling node or link failures. Link failures are defined as any link state in which the link is no longer capable of transmitting data between its endpoints. A node failure is defined as the inability of a node to process packets correctly. Assume a network modeled by a graph $G = (V, E)$ with an embedding $\mathcal{T}$ into $\mathbb{T}$, induced by a spanning tree $T = (V, E')$ with $E' \subseteq E$. Within this model, two types of link failures can be distinguished:

1. A link $e \in E$ fails for which $e \notin E'$ holds, therefore T does not become disconnected.
2. A link $e \in E$ fails for which $e \in E'$ holds, as such T becomes disconnected.

The first type of link failures does not critically affect routing in a negative way. Packet delivery is still guaranteed because e is a shortcut link (see Figs. 3a and 2a), although the stretch and load balancing behavior may worsen. In contrast, the second kind of failure is critical. Now, packet delivery can be obstructed for particular source-destination pairs, because the underlying embedding may lose its greedy property (see Fig. 3b). Node failures can also be divided into two classes:

1. A node $v \in V$ fails which is a leaf node of T, thus T does not become disconnected.
2. A node $v \in V$ fails which is not a leaf node of T, therefore T becomes disconnected.

In case a leaf node fails, no harm is done. Of course this prevents the node from becoming the source or destination of any path, but we ignore this trivial case. The failed node will never be a part of the sole path between two other nodes. Nonleaf nodes can, however, be critical transit nodes for network traffic flows. This can be explained as follows. In case no shortcut exists between two disjoint subtrees, all traffic has to be routed along their common ancestor node a (see Fig. 3a). Therefore, if a fails, all traffic between these two subtrees will encounter a void at a neighbor of a. However, due to the likelihood of the existence of shortcut links, negative effects of such node failures are diminished, but no guarantees can be made.

3.3. *Increasing Fault-Tolerance by means of a Multiembedding*

A first step toward increased fault-tolerance is the use of k spanning trees $T_i = (V, E'_i)$ with for any $i \in \{0, 1, \ldots, k-1\} : E'_i \subseteq E$, to form k greedy embeddings of G into $\mathbb{T}$, rather than only one [13]. These different greedy embeddings are denoted as $\mathcal{T}_i$. The notation δ is now used to denote the k-tuple representing the concatenation of the different distances in the k embeddings, thus $\delta = (\delta_0, \ldots, \delta_{k-1})$. Herein, δ_i represents the tree space distance for the i th embedding $\mathcal{T}_i$. Link or node failures must now disconnect all of the embedding-inducing trees $T_i = (V, E'_i)$ to certainly obstruct

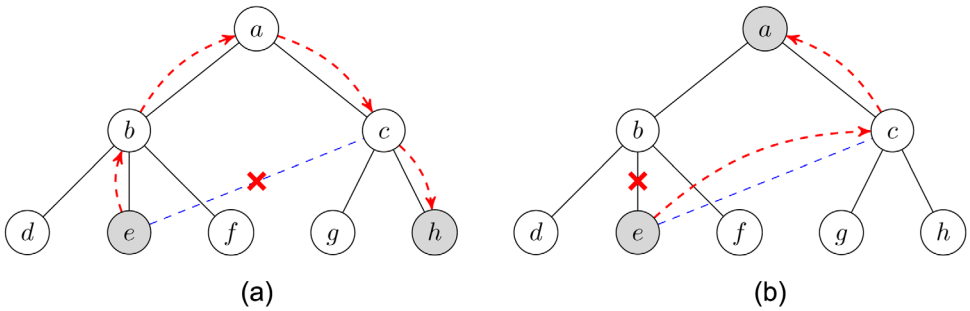


FIG. 3. Possible failure scenarios for an embedding into $\mathbb{T}$: Shortcut and critical links. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

a geometric routing algorithm. These link and node failures may once more be of different types:

1. A link $e \notin \bigcap_{0\le i<k} E_i'$ fails, thus only a fraction m of all trees T_i become disconnected.
2. A link $e \in \bigcap_{0\le i<k} E_i'$ fails, thus all k trees T_i become disconnected.
3. A node $v \notin \bigcap_{0\le i<k} V^{(i)}$ fails with $V^{(i)}$ the set of nonleaf nodes of tree T_i. Only a fraction m of the trees T_i becomes disconnected.
4. A node $v \in \bigcap_{0\le i<k} V^{(i)}$ fails. Now all k trees T_i become disconnected.

In the first and the third failure types, geometric routing is still theoretically possible due to the existence of a distance-decreasing path between every source-destination pair in the remaining $(k-m)$ embeddings. To illustrate with an example (see Fig. 2b): Link (h, y) is a critical parent–child link in embedding $\mathcal{T}_2$ but is a nonessential shortcut link in $\mathcal{T}_1$. The same holds for the network nodes: Nonleaf node y in $\mathcal{T}_2$ is a leaf node in $\mathcal{T}_1$. If (h, y) or y fails, routing in $\mathcal{T}_2$ may fail because $\mathcal{T}_2$ loses its greedy property, but it is still guaranteed in $\mathcal{T}_1$.

In case two and four, routing once more cannot be guaranteed. Therefore, to ascertain a 100% success rate, a backup mechanism is required, as will be explained in Section 3.3. However, as more embeddings are used, the chances of a link failure disrupting each of the embeddings becomes lower.

Another efficient way to increase routing robustness, without adding complexity to the routing forwarding layer, is increasing the number of viable alternative shortcut links. For this, the spanning tree creation mechanism should minimize the *tree redundancy*. With tree redundancy the overlap of different spanning trees in the same network is meant. For example, when two trees $T_i = (V, E')$ and $T_j = (V, E'')$ have a maximum redundancy, they completely overlap, meaning that $E' = E''$. Therefore, there is no advantage in using them both for routing purposes as they will lead to the same set of coordinates, as well as the same set of shortcut links. When these two trees are said to have low redundancy, the set $E' \cap E''$ is small. This principle is illustrated in Figure 2b in which two trees try to use different edges of the graph to avoid overlap. High tree redundancy should be avoided because it leads to low fault-tolerance: A failing link which happens to be a parent-child link for a large fraction of the forest is likely to cause routing voids.[5] How to achieve low tree redundancy is explained in Section 5.

3.4. *Embedding Complexity Analysis*

According to the previously explained embedding theory, each child of a vertex $v \in V$ is labeled with a number in $\{0, \ldots, (d_G(v) - 1)\}$ with $d_G(v)$ the degree of v in a graph G. The root node enjoys a special treatment and gets assigned the coordinate (0). Assuming an underlying spanning tree $T = (V, E')$ of $G = (V, E)$, then $d_G(v)$ is at most $(|V| - 1)$. Therefore the binary representation has a complexity $O(\log |V|)$ [15]. A balanced k-regular tree with $k > 2$ has a depth complexity of $O(\log |V|)$ and the coordinate lengths can be at most equal to depth of the tree. From this follows that every coordinate tuple can be represented with at most $O(\log^2 |V|)$ bits in such a tree. This polylogarithmic complexity in $|V|$ means that the coordinate representations are succinct [18]. Scale-free networks have a diameter $d \sim \log(\log |V|)$ which means that in the worst-case scenario (i.e., when the root node has a distance to another node equal to the diameter d), using a minimal-depth tree building algorithm such breadth-first mode (BFM) (see Section 5), the tree depth is at most equal to the network diameter [19]. Thus the complexity of the tuples becomes $O(\log(\log |V|) \times \log |V|)$. However, when taking into account the most general scenario, a network can have a diameter that is nearly equal to the number of vertices in the network, namely $(|V| - 1)$. In this worst-case scenario (with a worst-case tree depth) the binary storage complexity becomes $O(|V| \times \log |V|)$.

4. GREEDY FOREST ROUTING

In this section, the theoretical foundation presented in Section 3 will be combined with a multiembedding, forming GFR. A straightforward way of routing with multiple embeddings is to allow a node to freely alternate between the different embeddings, due to their individual greediness. However, this naive forwarding mechanism lacks reliability as routing cycles may be introduced. Routing along a

[5] Recall that routing voids are nodes that are unable to forward a packet because there exists no distance-decreasing neighbor.

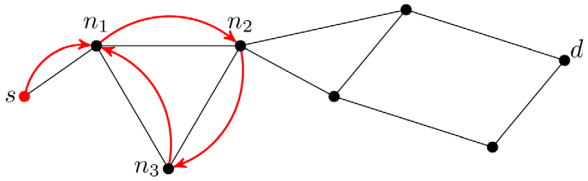


FIG. 4. Naive routing example with multiple embeddings that leads to a cycle. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

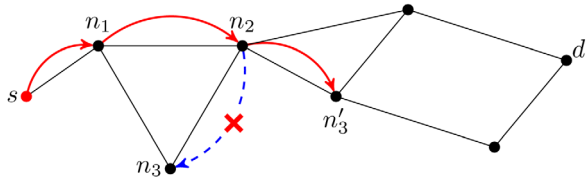


FIG. 5. GFR example in which a cycle is avoided due to its ε-distance. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

distance-decreasing path in $\mathcal{T}_i$ may increase the distance in a different embedding $\mathcal{T}_j$. At a certain vertex along the routing path, the packet may be sent back to its origin, routing the packet along a cycle, as illustrated by the following example.

Example. To illustrate what happens when routing naively with multiple embeddings, a fictional example is given for a multiembedding consisting of three embeddings, thus $k = 3$. This is depicted in Figure 4. Assume a source node s and a destination node d for which $\delta(s,d) = (9,5,7)$. This means that the δ-distance is 9 in embedding $\mathcal{T}_0$, 5 in $\mathcal{T}_1$, and 7 in $\mathcal{T}_2$. When routing naively the following scenario can occur:

- $\delta(n_1,d) = (8,4,7)$, s, the source, decides to route to n_1 in embedding $\mathcal{T}_0$ because δ_0 decreases from 9 to 8.
- $\delta(n_2,d) = (3,6,4)$, n_1 decides to route to n_2 in embedding $\mathcal{T}_2$ because δ_2 decreases from 7 to 4. However, at the same time the distance in embedding $\mathcal{T}_1$ increases from 4 to 6.
- $\delta(n_3,d) = (6,5,7)$, n_2 decides to route to n_3 in embedding $\mathcal{T}_1$ because δ_1 decreases from 6 to 5. Notice that the distance in embedding $\mathcal{T}_2$ increases from 4 to 7 and that the distance in $\mathcal{T}_0$ increases from 3 to 6.
- $\delta(n_1,d) = (8,4,7)$, n_3 decides to route to n_1 in embedding $\mathcal{T}_1$ because δ_1 decreases from 5 to 4.

In this scenario a cycle has been introduced because although the packet is routed greedily in each embedding individually, this does not hold for their aggregation.

Cycles can be avoided by requiring that each vertex along the routing path decreases its minimum distance (over the k embeddings) to the destination. This way of working is similar to the tree cover based geographic routing (TCGR) mechanism [13]. In our work a new distance function $\varepsilon : \mathbb{T}^k \times \mathbb{T}^k \to \mathbb{R}^+$ is defined as

$$\varepsilon(u,v) = \min_{0 \le i < k} \{\delta_i(u,v)\} \text{ with } u,v \in V. \tag{3}$$

The k embeddings into $\mathbb{T}$ can now be treated as a single k-dimensional greedy[6] embedding into $\mathbb{T}^k$. As such, the principles of geometric routing can be respected using the semimetric space $(\mathbb{T}^k, \varepsilon)$. This double cannot be regarded as a true metric space as the triangle-inequality no longer holds (Definition 3, property 3), however, this is not necessary for geometric routing. When forwarding, multiple neighbors may have an equal ε-distance to the destination. In this case a random choice is made among them. This ε-based routing scheme, in combination with the embedding procedure presented in the next section, is called GFR. In the next example we illustrate what happens when using GFR in the previous scenario.

Example. Reusing the same scenario as before, a multiembedding is used consisting of three embeddings, thus $k = 3$, which is depicted in Figure 5. Assume a source node s and a destination node d for which $\delta(s,d) = (9,5,7)$. This means that the δ-distance is 9 in embedding $\mathcal{T}_0$, 5 in $\mathcal{T}_1$, and 7 in $\mathcal{T}_2$. As such, based on Equation (3), $\varepsilon(s,d) = \min\{9,5,7\} = 5$. GFR avoids cycles as shown in the following scenario:

- $\delta(n_1,d) = (8,4,7)$, s, the source, decides to route to n_1 because $\varepsilon(n_1,d) = 4$, which is lower than its previous value 5.
- $\delta(n_2,d) = (3,6,4)$, n_1 decides to route to n_2 because $\varepsilon(n_2,d) = 3$, down from 4.
- $\delta(n_3,d) = (6,5,7)$, n_2 cannot rout route to n_3 because $\varepsilon(n_3,d) = 5$, which is larger than 4. Therefore, GFR avoids the cycle.
- $\delta(n_3',d) = (2,2,3)$, n_3 decides to route to n_3' because $\varepsilon(n_3',d) = 2$, down from 3.

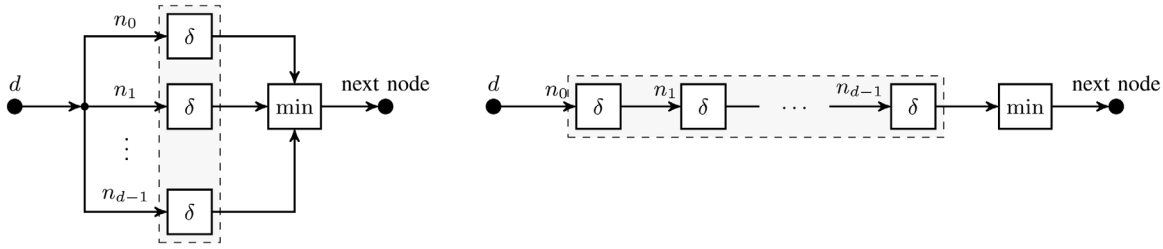
Contrary to the previous example, GFR avoids the cycle because it uses the ε-distance, which leads to a greedy embedding into $(\mathbb{T}^k, \varepsilon)$.

As some nodes can have a high number of neighbors in scale-free graphs, the number of calculations needed for forwarding decision making can be equally large. Now, the router processing time for packet forwarding using GFR will be examined. For each neighboring node, the distance in k embeddings has to be calculated to generate the ε-distance, after which the node with the minimal distance is identified. When computing this in a sequential way, it takes a processing time equal to

$$t_{\text{proc}}(u) = kd_G(u)t_{\text{dist}} + \log_2(kd_G(u))t_{\text{comp}} \tag{4}$$

for every packet, for a node $u \in V$. In this equation, t_{comp} represents the time required to compare two distances and select the lowest one, leading to $\log_2(kd_G(u))$ comparisons; t_{dist} stands for the time required to calculate a single distance

[6] Its greedy characteristics can easily be proven: When the minimum distance of all the embeddings decreases, at least one embedding will reach a new minimal distance $\delta(u,d)$ at the current node u for a destination d. Assume this embedding is $\mathcal{T}_i$, then there will exist a node $v \in N(u)$ for which $\delta_i(v,d) < \delta_i(u,d)$ because $\mathcal{T}_i$ is a greedy embedding (see Definition 1).



(a) parallel packet processing

(b) sequential packet processing

FIG. 6. Schematic overview of router architecture: Routers can parallelize distance calculations for a destination node d for each neighboring node $n_0, n_1, \ldots, n_{d-1}$. The comparison operation is the only sequential part. In this figure we show only the calculations for a single embedding to avoid clutter, the extension toward a k-dimensional multi-embedding can be done by duplicating the distance calculations k times for $n_0, \ldots, n_{d-1}$ and adding them parallelly to (a) and sequentially to (b).

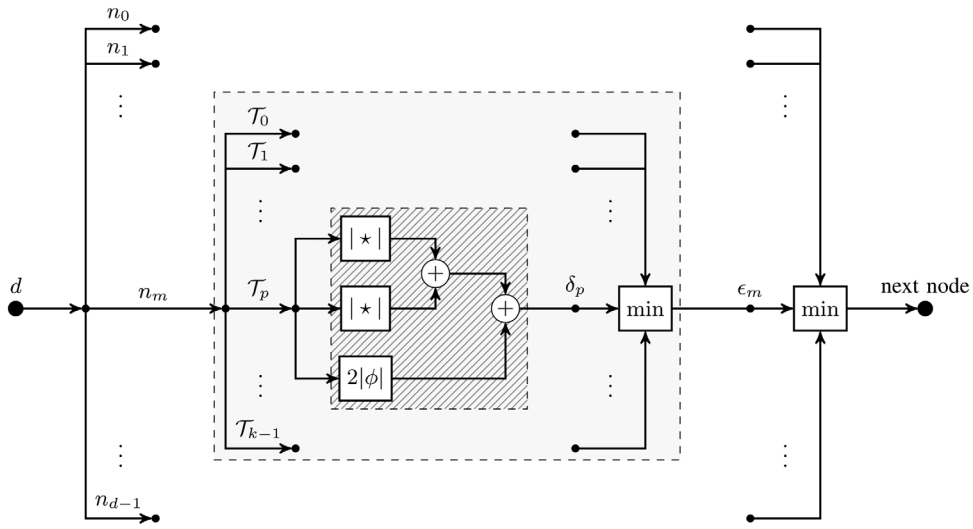


FIG. 7. Overview of sequential and parallel processing steps in GFR forwarding. The shaded box represents the δ-function based on Equation (2), which is calculated in parallel for every embedding $\mathcal{T}_i$, while the dashed gray box represents the distance function ε based on Equation (3), which is calculated for every neighbor n_i, assuming a destination node d.

between two points in the tree coordinate space; $d_G(u)$ represents the degree of node u; and k is the number of embeddings used in the routing system. An advantage is, however, that the routing system is naturally parallelizable. The system can be broken down in two pieces: The distance calculation and the computation to select the next node. The distance calculation can be performed in parallel as the calculations of the different distances are not interdependent. The comparison computation has to be done after the distance calculations, assuming all distance calculations are equally fast. This processing scheme is depicted in Figure 6. When computing the distances in parallel, the total time becomes

$$t_{\text{proc}}(u) = t_{\text{dist}} + \log_2(kd_G(u))t_{\text{comp}}. \tag{5}$$

Herein, the term with t_{dist} is not scaling with the number of neighbors or the number of embeddings at all, which leads to a constant complexity. The term with t_{comp} does scale however, but the task of comparing two values and extracting the minimum is not computationally intensive.

Figure 7 shows the GFR calculations needed for forwarding, with its serial and parallel steps. The symbols used in the figure are based on Equations (2) and (3). Although many calculations are necessary, it can be noticed that they are very parallelizable: All neighboring node cost values can be calculated independently. Note that in this figure we disregard the random selection mechanism required when multiple nodes have an equal cost value. For any neighbor, ε can be calculated in parallel by calculating every δ-value independently for each embedding. Assume a node u has $d_G(u)$ neighbors, and the embedding is k-dimensional, then δ has to be calculated $kd_G(u)$ times in parallel. The output value δ_p is then fed to a minimum selection mechanism, obtaining ε. The node for which this ε is minimal will become the next node along the routing path to the destination. This architecture allows for a highly efficient GFR routing forwarding procedure.

5. EMBEDDING CONSTRUCTION

In this section, the GFR graph embedding construction mechanism is described. By constructing the forest

underlying the embedding in an intelligent manner, GFR can be inherently robust toward network failures. A prerequisite for constructing this embedding is that each node has a unique identification number, for example, this can be a node's MAC-address in a real network. This is needed for having a deterministic solution to the election process. In what follows, the identification number of a vertex v is called its key and is denoted as $\mathcal{K}(v)$. The GFR tree construction procedure consists of two phases. First, a set of k distinct root nodes $\left\{r^{(i)}\right\}$ are elected through an arbitrary election mechanism. Second, the spanning trees T_i are constructed along with the corresponding greedy graph embeddings $\mathcal{T}_i$. Hereafter we only explain the construction of a single $\mathcal{T}$, multiple embeddings are a trivial parallel extension. The embedding procedure is based on the work of Chávez et al. [14] and Korman et al. [15]. The major difference lies within the way the tree is formed.

Initially, the root node r is assigned the coordinate set (0). Next, it will send its coordinates to its neighbors $N_1(r)$.[7] Furthermore, each recipient is assigned a unique child number by the sender (the parent). Upon receiving this message, nodes compute their own coordinates and inform their neighbors $N_2(r)$ with a similar message. In turn these will recursively inform their neighbors $N_j(r)$. An illustration is shown in Figure 2a of Section 3.2. In case a node that already has coordinates assigned to it receives a new coordinate assignment message, it may react in different fashions, which are called different tree construction modes:

- *Breadth-first mode* (BFM): Override the previous coordinate tuple only if this leads to a tuple of lower length. This results in a tree of minimal depth.
- *Redundant mode* (RM): Only override previous coordinates when the cost function value of the new coordinates is lower. This cost function is based on the coordinate tuple lengths and the overlap of the spanning trees (tree redundancy), which is explained later on.
- *Breadth-first RM* (BFRM): A hybrid mode combining BFM and RM, constructing a minimal-depth tree while focusing on minimizing tree overlap.

In BFM, coordinates are only overridden when the coordinate tuple resulting from the received message has a lower length than the current tuple. When the parent node has acquired a new set of coordinates, the child node should also be updated accordingly. As such, a tree of minimal depth is built, rooted at r. An advantage of using BFM is that the introduction of cycles is impossible. Its high tree redundancy is a clear downside, as this leads to low robustness: If a link fails which happens to be the link between parent and child for a large fraction of the trees, this will likely cause routing voids. To enhance reliability, a second mode has been developed called RM. RM emphasizes low tree redundancy, leading to high tree coverage. This is done by ensuring that every link is an edge in approximately the same number of trees. For this, a novel metric is introduced to measure the amount of tree redundancy of the spanning trees inducing the different embeddings. Every edge $e \in E$ is part of a number of sets E'_i for a number of different trees $T_i = (V, E'_i)$. If minimal redundancy is desired, the goal is to make this number more or less equal for every edge in E of $G = (V, E)$. Based on this description of redundancy, a metric τ is defined by the following definition.

Definition 4. *Assume k spanning trees. Denote the number of different sets E'_i, corresponding to $T_i = (V, E'_i)$ with $0 \leq i < k$, that an edge $e \in E$ in a graph $G = (V, E)$ belongs to, as n. The amount of tree redundancy, termed the τ-ratio, is defined as the standard deviation $\sigma(n)$ divided by the average $\bar{n}$ over all $e \in E$. This can be formulated as $\tau = \frac{\sigma(n)}{\bar{n}}$.*

A low τ-ratio indicates that the trees are spread evenly over the network, while a high τ-ratio indicates that there is a huge difference in the number of spanning trees a link is part of. Also note that $\tau \geq 0$. To decrease the tree redundancy while constructing the embedding, a cost function $f_i : V^3 \to \mathbb{R}$ is defined for each embedding $\mathcal{T}_i$ used:

$$f_i(u, p, p') = \eta a_i(u) + \beta b_i(u) + c_i(u, p, p') \tag{6}$$

with $\eta, \beta \in \mathbb{R}$ adjustable parameters; a_i, b_i, and c_i are cost terms defined as

$$a_i(u) = |u'| - |u|, \tag{7}$$

$$b_i(u) = \begin{cases} 0 & \text{if } a_i(u) \leq 0 \\ |u'| - |u^*| & \text{if } a_i(u) > 0, \end{cases} \tag{8}$$

$$\begin{aligned} c_i(u, p, p') = {} & |\Psi(u, p') - \overline{\Psi}(u) + 1| + |\Psi(u, p) - \overline{\Psi}(u) - 1| \\ & - (|\Psi(u, p') - \overline{\Psi}(u)| + |\Psi(u, p) - \overline{\Psi}(u)|), \end{aligned} \tag{9}$$

with $u \in V$ also representing the current coordinates for embedding $\mathcal{T}_i$; u' are the new coordinates for embedding $\mathcal{T}_i$; u^* are the coordinates with the lowest length encountered so far for embedding $\mathcal{T}_i$; $\Psi(x, y)$ represents the number of trees T_i that are making use of the edge $(x, y) \in E$; $\overline{\Psi}(x)$ is the average of $\Psi(x, y) \forall y \in N(x)$; p is the current parent of u, and p' is the potential new parent.

Thus $a_i(u)$ indicates the decrease in length of the new coordinates compared to the current coordinates of u, while $b_i(u)$ represents the decrease in length of the coordinates under consideration for node u when comparing them to the coordinates with the lowest length ever assigned (and possible overridden) to u. The cost term $c_i(u, p, p')$ describes whether the number of trees that each link $e \in I(u)$[8] is part of equalizes or not (which is our goal, ensuring that every link is part of an equal number of trees, to minimize τ). In this cost term, $|\Psi(u, p') - \overline{\Psi}(u) + 1|$ describes the offset of the number of trees link (u, p') is part of versus the average of this value.

[7] $N_i(u)$ represents the i-th hop neighbors of u.

[8] $I(u)$ represents the set of incidents links of node u.

This average is denoted as $\overline{\Psi}(u)$, and is calculated over all links in $I(u)$. Furthermore, the term $|\Psi(u,p) - \overline{\Psi}(u) - 1|$ is added. It describes the offset of the number of trees link (u,p) is part of, when p is no longer a parent of u, versus the average over the links $I(u)$, denoted as $\overline{\Psi}(u)$. The sum of these two offsets is compared to the sum of the offsets without parent replacement, which is $(|\Psi(u,p') - \overline{\Psi}(u)| + |\Psi(u,p) - \overline{\Psi}(u)|)$. Therefore, c_i describes whether overriding the coordinates (accepting the new parent) will reduce the variance in the total number of trees each link in $I(u)$ is part of. Also note that $c_i(u,p,p')$ is bounded by $-2 \leq c_i(u,p,p') \leq 2$.

Algorithm 1 Tree construction, redundant mode (RM)

input : vertex u has sent its coordinates to its neighbors; vertex $v \in N(u)$ is listening for packet receipt

output : vertex v has coordinates assigned according to embedding $\mathcal{T}$ in $\mathbb{T}$

1 receive incoming packet `packet` of node u along with its coordinates $\mathtt{u}$, a child number c_u and key $\mathcal{K}(u)$
2 look up own coordinates $\mathtt{v}$ and parent key $\mathcal{K}(p)$ corresponding to embedding $\mathcal{T}$
3 **if** $\mathtt{v} = \emptyset$ **then**
4 go to line 11
5 **else if** $\mathcal{K}(u) = \mathcal{K}(p) \vee (f_i(v,p,u) < 0)$ **then**
6 **if** $\mathcal{K}(v) \in P \wedge \mathcal{K}(u) \neq \mathcal{K}(p)$ **then**
7 cycle avoided, drop `packet`
8 **else if** $\mathcal{K}(v) \in P \wedge \mathcal{K}(u) = \mathcal{K}(p)$ **then**
9 cycle detected because parent update message received, resolve cycle
10 **else**
11 $c_u \leftarrow$ child number extracted from `packet`
12 calculate own coordinates $\mathtt{v} \leftarrow \mathtt{u}^\frown\{c_u\}$
13 update own parent $p \leftarrow u$
14 record own key in `packet` in sequence of traversed nodes: $P \leftarrow P^\frown\mathcal{K}(v)$
15 **foreach** $n \in N(v)$ **do**
16 send `packet` containing $\mathtt{v}$ and neighbor number c_v to n
17 **else**
18 drop `packet`

The graph embedding procedure in RM is described in Algorithm 1. When growing a tree in RM, precautions have to be taken to avoid the introduction of cycles, especially when using low η-values in Equation (6), as nodes will then frequently switch parents. The fact that now longer coordinates may override shorter ones is the main reason for the possible introduction of cycles in the tree. This happens when an ancestor node accepts new coordinates from one of its descendants. As a result, the embedding $\mathcal{T}_i$ may no longer be greedy. A solution at first glance is to check whether a receiving node is an ancestor of the sending node by checking if $|\phi(u',u)| \neq |u|$ holds [see Equation (2)]. But even then it is still possible to generate cycles as explained in Figure 8. To counter this, every coordinate assignment packet also holds information about the current path P up to the root in the tree [e.g., (x,g,c,a) in Fig. 8a]. The key $\mathcal{K}(v)$ of every vertex $v \in P$ is recorded. If a node u receives a coordinate assignment message, it will first check whether it is part of P by examining if its own key is present in this set of recorded keys. Only when $u \notin P$ it will consider accepting the new coordinates. This mechanism is called *cycle avoidance*.

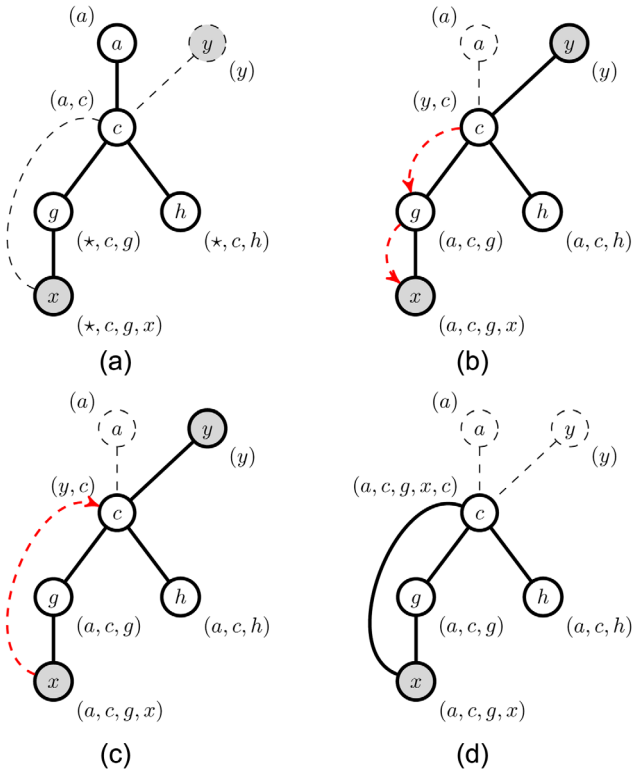


FIG. 8. Cycle introduction by faulty coordinate updates which results in an endless loop of updates. The characters in the coordinate tuples represent integers similar to Figure 2a. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Nonetheless, even with the cycle avoidance active, cycles may still be introduced due to the system's distributed nature. Therefore, a *cycle resolution* procedure is implemented, shown in Algorithm 1 at lines 1 and 9. When a node receives a coordinate message from its parent, it normally accepts these new coordinates. However, when this packet already has its own key in the P field, a cycle was introduced. For example, in node c the P field will look like (a,c,g,x,c) in Figure 8c. Hence, it is informed of the cycle $\langle c,g,x,c\rangle$. Next, c will send on the coordinate message to its children to notify them. Whenever a node detects a cycle, it searches for a new parent among its neighbors that are not part of the current cycle. This is done by querying a new parent p and asking for its coordinates along with a new child number c_p. After the packet has traversed all nodes of the cycle, these will all have had the opportunity to switch parent, thus resolving the cycle.

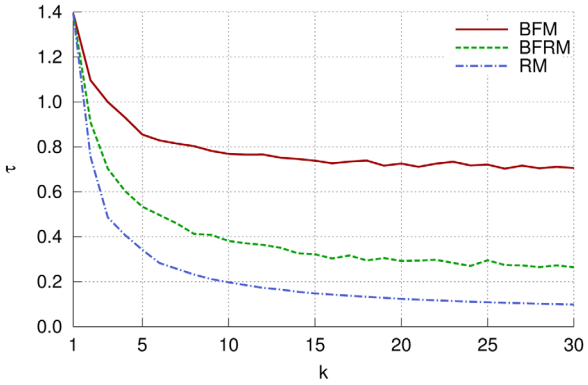


FIG. 9. Tree redundancy τ in function of the dimension k (the number of embeddings) for different tree generation modes on a scale-free (see Section 7) graph with 500 nodes. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

The advantage of generating an embedding in RM is the resulting low tree redundancy τ. However, it is very hard to give an estimated upper bound on the embedding procedure time due to the complex interactions of the cycle resolution and avoidance mechanisms. Moreover, the coordinates will be larger than when building a tree of minimal depth. Therefore, a hybrid mechanism is established by combining BFM with RM, called BFRM. Now, a tree of minimal depth is combined with a cost function to minimize τ. In essence, this comes down to assigning parameter values $\eta > 2$ and $\beta = 0$ in Equation (6). BFM can easily be altered to BFRM by allowing a coordinate assignment message to override the current coordinates when the check $(f_i(v,p,u) < 0)$ is true. Because of its breadth-first behavior, no cycle avoidance or resolution is required, which greatly diminishes the complexity of the embedding procedure. This stems from the fact that a node can never increase its coordinate tuple length. The tree redundancy of the different generation modes in function of the number of trees generated k can be seen in Figure 9. For RM, the values in Equation (6) are set to $\eta = \frac{2}{3}$ and $\beta = \frac{1}{3}$, which leads to good behavior (no excessively deep trees in which lots of cycles have to be resolved). From this it is clear that RM results in the lowest redundancy τ, BFM has the highest τ and BFRM lies somewhere in between. The relation between τ and fault-tolerance, will be shown for the different construction modes in Section 7.1.

Additionally, the difference between the coordinate lengths of BFRM and RM are shown in Figure 10. Figures 10 and 9 are consistent with the theoretical analysis in Section 7.4: When comparing BFRM with RM it can be noticed that RM indeed generates larger coordinates in favor of less tree redundancy (lower τ). The tradeoff made is thus larger storage requirements in both the nodes and packet headers, versus possibly higher robustness. Although the median coordinate length of RM seems similar to BFRM, its variance is much higher, leading to inconsistent coordinate lengths.

6. ROBUSTNESS MECHANISMS

In this section, additional mechanisms for increasing robustness of GFR are investigated. First, the fault-tolerance of the geometric routing system is improved by adding a backup routing mechanism to allow for a 100% success rate in failure scenarios. Second, routing in dynamically changing network topologies is researched. In particular, a re-embedding strategy is laid out to counter routing failure due to network alterations.

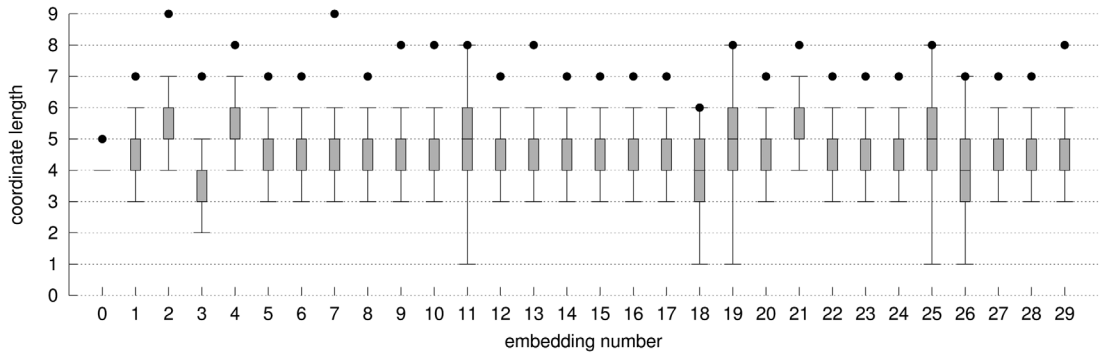
6.1. *Backup Routing Mechanism*

When routing without any sort of backup routing mechanism, a packet traverses a path P based on a function that is computed at every node $u \in P$ as Equation (3) dictates. When a node along the routing path can identify no viable next neighbor, that is, no neighboring node has a lower distance to the destination, the packet encounters a void. Although using a forest embedding reduces the chances of such failures, they can still occur. Therefore, GFR is extended with a backup routing mechanism. Upon encountering voids, the whole neighborhood $N(u)$ is investigated instead of just a subset of nodes, denoted as $S(u)$, for which ε decreases. Doing this headlong, however, may cause some routing paths to become cycles [8], thus prudence is prevalent.

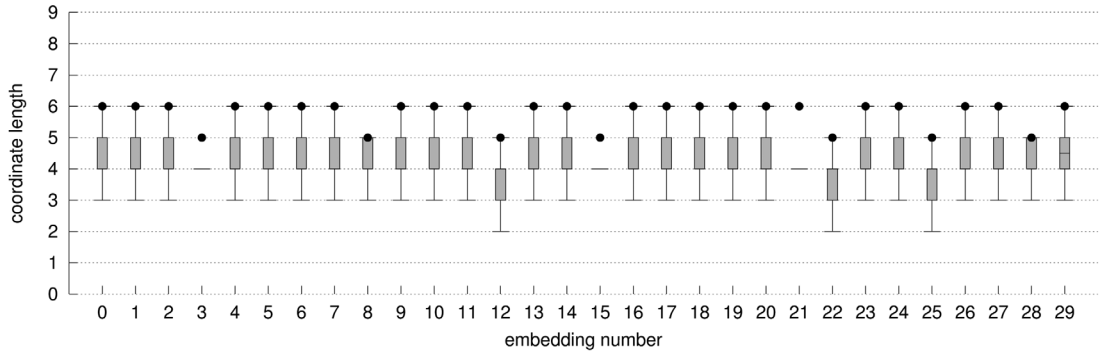
The backup system becomes active upon encountering a routing void, after which the keys of the visited nodes, along with the number of times they are visited, are saved in the header of the data packet being routed. To avoid cycles, each node will inspect these visitation numbers. Each node will now forward packets to the node with the lowest visitation number out of all considered nodes. When multiple nodes have an equal visitation number, the node with the lowest ε-value is selected. Cvetkovski and Crovella [8] have proven that such a mechanism is able to attain 100% routing success rate even in severe failure scenarios. The algorithm is described in Algorithms 2 and 3. When a packet encounters a void (and the backup system is not active), the current ε-distance is saved in the packet header after which the packet enters the *backup mode*, until a lower ε-distance is encountered. This has much in common with the greedy- and face-mode used by [1] in which also a backup mechanism is used to escape voids. The effects of the backup routing mechanism on stretch and success ratio in failure scenarios will be tested in Section 7.3.

6.2. *Changing Topology*

The underlying network may have a dynamic topology in which links are added or removed over time. This results in the spanning trees, inducing the graph embeddings of the GFR system, losing their connectivity because they do not adapt to the dynamic topology. Consequentially, routing performance deteriorates as time goes on, which can first be witnessed by an increased stretch. Moreover, in case the network is severely altered, even routing voids may occur. A solution is to create new embeddings while traffic is being



(a) RM embedding



(b) BFRM embedding

FIG. 10. Coordinate sizes in function of the embedding number for BFRM and RM for a scale-free graph with 500 nodes, by means of a boxplot. The dots represent the highest value attained. The x-axis depicts the embedding number, while the y-axis depicts the coordinate sizes corresponding to this embedding.

Algorithm 2 Backup routing in case of failures

input : current node u knows its neighbors $N(u) \in V$ and the current load of its currently incident edges $I(u) \in E$; a k-dimensional embedding $\mathcal{T}^k$ is assumed; a destination node d
output: next node v to forward the packet p to

receive incoming packet p that has to be forwarded
calculate $S(u)$ as in the default GFR mechanism
if $S(u) = \emptyset$ **then**
  if p.backup = false **then**
    p.map $\leftarrow \emptyset$
    p.backup $\leftarrow$ true
  if p.map contains entry for $\mathcal{K}(u)$ **then**
    $h \leftarrow$ p.map.value$(\mathcal{K}(u))$
    p.map.value$(\mathcal{K}(u)) \leftarrow h + 1$
  else
    p.map.value$(\mathcal{K}(u)) \leftarrow 1$
  $v \leftarrow$ SELECTNEXT() (see Algorithm 3)
else
  $v \leftarrow$ apply default GFR selection procedure

Algorithm 3 selectNext() in Algorithm 2

input : input of Algorithm 2; map p.map containing visitation number of nodes along the path that p traversed
output: next node v to forward the packet to

$\mathrm{h}_{\min}, \epsilon_{min} \leftarrow +\infty$
$R(u) \leftarrow \emptyset$
foreach $m \in N(u)$ **do**
  if p.map contains entry for $\mathcal{K}(m)$ **then**
    $\mathrm{h}_m \leftarrow$ p.map.value$(\mathcal{K}(m))$
  else
    $\mathrm{h}_m \leftarrow 0$
  if $\mathrm{h}_m < \mathrm{h}_{\min} \vee (\mathrm{h}_m = \mathrm{h}_{\min} \wedge \epsilon(m,d) < \epsilon_{min})$ **then**
    $R(u) \leftarrow \emptyset$
    $\epsilon_{min} \leftarrow \epsilon(m,d)$
    $\mathrm{h}_{\min} \leftarrow \mathrm{h}_m$
    $R(u) \leftarrow R(u) \cup \{m\}$
  else if $\mathrm{h}_m = \mathrm{h}_{\min} \wedge \epsilon(m,d) = \epsilon_{min}$ **then**
    $R(u) \leftarrow R(u) \cup \{m\}$
$v \leftarrow$ random element from $R(u)$

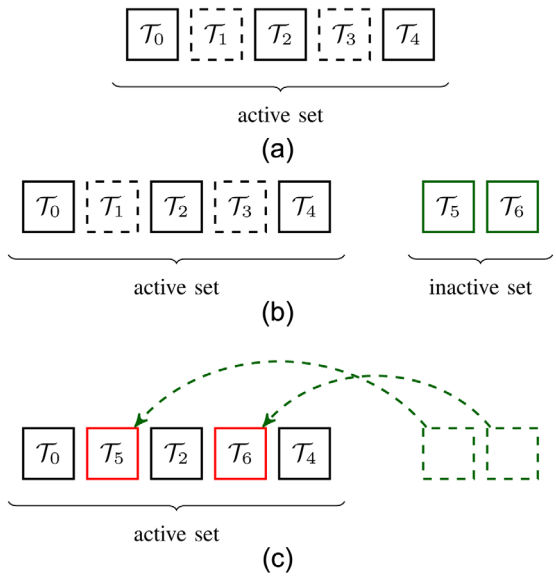


FIG. 11. The regeneration of two embeddings in a set of five embeddings, thus $k = 5$ and $n = 2$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

routed through the network, which ultimately replace older ones. The solution proposed here enables nodes to initiate a voting process, or a voting process is initiated with a certain frequency, in which nodes decide whether or not a new spanning tree—and thus a new embedding—should be created. If the voting is successful, the graph embedding procedure as explained in Section 5 is initiated. An important property that should hold is that as the network changes over time, the chances of a passing vote should increase. The implementation details of this voting scheme is omitted in this work, instead, we focus on its theoretical principles.

When using a k-dimensional tree space $\mathbb{T}^k$, induced by k spanning trees $T_0, T_1, \ldots, T_{k-1}$, a new tree T_k is generated accompanied by new coordinates. When the embedding generation procedure based on T_k has started at a point in time t_0, all nodes will wait a time Δ_1 before they conclude that the embedding procedure has ended. At this point $t_0 + \Delta_1$ there exist $(k + 1)$ embeddings. This can be generalized to the creation of multiple spanning trees in parallel. If this is the case, there might exist $(k+n)$ embeddings at some point, from which there are k active and n nonactive ones. This procedure is illustrated in Figure 11. At a time $t_0 + \Delta_1$ all nodes will communicate their new sets of coordinates with their neighborhood (Fig. 11b).

After some time Δ_2 has passed, all nodes should know the n new coordinate sets of their neighbors. Then after the point $(t_0 + \Delta_1 + \Delta_2)$ in time, all nodes will discard n of the old embeddings from the original working set and exchange them with the newly created ones (Fig. 11c). All nodes should be aware of the new coordinate sets within the time period $(\Delta_1 + \Delta_2)$ after the start of the generation of the new embeddings, as hereafter the new coordinate sets will be used for routing purposes. Note that every node has to exchange the same set of embeddings. A potential issue could be the requirement to frequently exchange embeddings in a scenario where the network is highly dynamic. The effects of using an embedding regeneration scheme on the stretch and the success ratio of GFR will be investigated in Section 7.4.

7. RESULTS AND DISCUSSION

Experimental results are generated by a routing simulator that continuously generates random source-destination host pairs according to a uniform traffic matrix. Between these pairs traffic is simulated by generating routing paths in a multithreaded environment. All experiments are executed on a High Performance Computer. The algorithms are tested on different scale-free networks constructed according to the Barabási-Albert model [20] which have a degree distribution $P(d) \propto d^{-\gamma}$ with $\gamma = 2.2$.

7.1. *Robustness*

This section evaluates the fault-tolerance of GFR. Based on our previous theoretical analysis, we expect that multiple embeddings will lower the chances of a network failure disrupting each of them. Therefore, the routing system is more likely to find a distance-decreasing neighbor to which an incoming packet can be forwarded. Hence, GFR should be more resilient to failures than geometric tree-based routing using a single embedding, such as RTP [14].

In this section, we focus entirely on link failures, as node failures lead to very similar results. In the following experiment a number of links are removed from the network which represent link failures. The number of links removed starts at 0 and goes up by 20 in a step-wise fashion. When representing the network by a graph $G = (V, E)$, the highest number of links that can be removed without disconnecting G is $(|E| - |V| + 1)$. For every step in the number of links failed, the experiment is repeated 100 times. At each of these 100 repetitions, 1000 random source-destination pairs are generated between which uniform traffic is simulated, leading to 10^5 simulated paths for each step. For each of these 1000 pairs, the average success ratio is monitored. The links are removed with a probability

$$p(l) = \exp\left(3\left(1 - \frac{d_G(v)_o}{(d_G(v) - 1)}\right)\right), \qquad (10)$$

with v the vertex that has the lowest probability $p(l)$ of both vertices incident to $l \in E$ and $d_G(v)_o$ the original degree of vertex v before removal of any links. This probability ensures that the link failures are spread evenly across the network. When $d_G(v) = d_G(v)_o$ (no incident links failed), the failure probability $p(l)$ goes to one, while it goes to zero for $d_G(v) = 0$ (nearly all incident links failed). Links are removed according to the description above randomly for each run of the experiment.

Figure 12 compares the routing success rate of GFR with $k = 15$ to geometric tree-based routing with a single embedding, namely RTP [14], exercised on a scale-free graph of 500

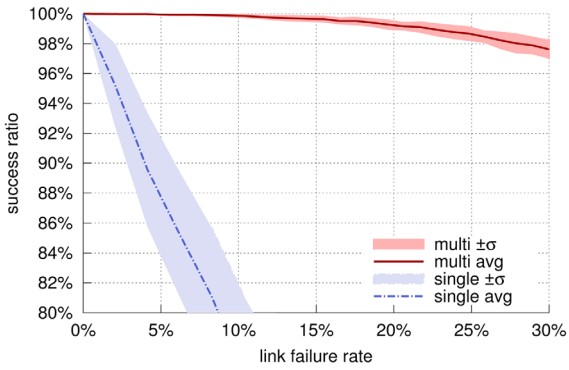


FIG. 12. Fault-tolerance: The success ratio of GFR ($k = 15$, BRFM embedding generation) is tested for a varying failure rate on a scale-free graph of 500 nodes. Also RTP [14] in which only a single embedding is used is evaluated. The shaded background represents the average success ratio, plus and minus the standard deviation σ. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

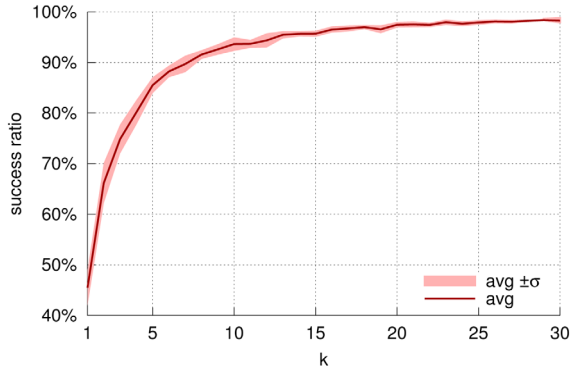


FIG. 13. Success rate at a link failure rate of 30% for a varying dimension k on a scale-free graph of 500 nodes. The embedding is generated according to BRFM. The shaded background represents the average success ratio, plus and minus the standard deviation σ. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

nodes. GFR is able to attain a success rate over 97% when 30% of the removable links are removed, given a connected network. This is a huge improvement over routing with a single embedding where the success ratio quickly declines as the number of failures increases, namely a success ratio less than 50% at a 30% failure rate. This empirically shows that GFR is able to achieve high robustness without using any form of path protection or restoration, as in the work of Sahhaf et al. [6, 7], and Cvetkvoski and Crovella [8].

This inherent fault-tolerance can be explained by the availability of more embeddings, which causes more alternative pathways to exist between source and destination. Therefore, it is less likely that a routing void will occur, a situation in which no new ε-decreasing neighbor can be found. From this can be concluded that GFR is robust by its very nature, an essential property in real large-scale networks where failures are common.

Figure 13 shows how increasing the dimension k affects GFR's robustness at a constant link failure rate of 30%. In this experiment, for each k value, 15 different runs are executed in which 10^5 random source-destination pairs are generated between which traffic is simulated. At low k-values, the success ratio of GFR is low, which is consistent with Figure 12. As k increases, so does the success ratio, until it converges to 100% as $k \to +\infty$. This is consistent with more embeddings allowing for more alternative paths by which packets are naturally rerouted to their destination.

The effect of the different graph embedding procedures, as presented in Section 5, on the success ratio is shown in Figure 14. In this figure can be noticed that RM is more robust to failures than BFM, while BFRM lies somewhere in between. This indicates the correlation of tree redundancy (see Definition 4) with robustness. The τ-ratios of the different embedding modes can be seen in Figure 9 for varying values of k. This result is important as the improved robustness does not require any alteration of the routing forwarding layer. It is entirely dependant on the initial embedding procedure.

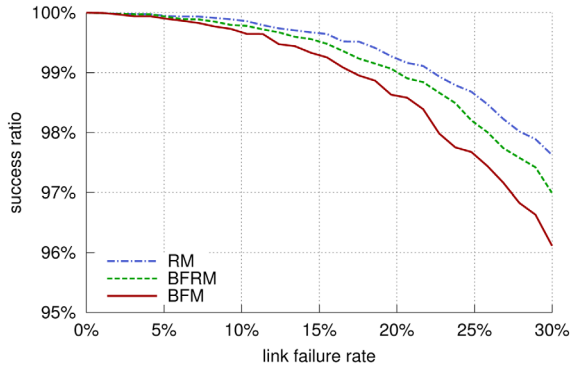


FIG. 14. Effect of the different graph embedding procedures on the robustness of GFR ($k = 15$) on a scale-free graph of 500 nodes. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

7.2. *Path Stretch*

As geometric routing does not guarantee shortest path routing, we have to make sure that increasing fault-tolerance in GFR does not lead to excessively long routing paths. In this section GFR is analyzed regarding its average stretch $\overline{\rho}$. We define the stretch of a routing path between a source node s and a destination node d as the path length, in number of hops, divided by the length of a shortest path between s and d.

In Figure 15, the effect of changing the number of embeddings k (or the dimension of the space $\mathbb{T}^k$) on the average stretch $\overline{\rho}$ is depicted. For each k-value, 15 different runs are executed for which 10^5 random source-destination pairs are generated, between which traffic is simulated. This experiment shows asymptotic behavior $\overline{\rho} \to 1$ as $k \to +\infty$. This means that as k increases, GFR approximates shortest path routing. A logical explanation for this is that having more embeddings available allows for more routing freedom, as a node has more embeddings to choose from when routing packets. As a result, there is an increased chance that a combination of embeddings will lead to a short path between source and destination. Therefore, attaining a lower stretch

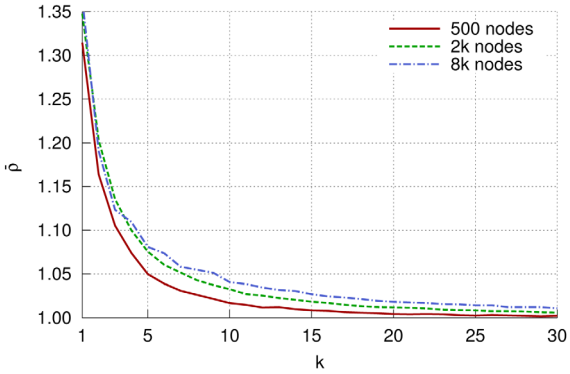


FIG. 15. GFR with BFRM embedding generation: Average stretch $\overline{\rho}$ in function of the dimension k of the embedding space $\mathbb{T}^k$ for scale-free graphs of different sizes. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

by increasing k goes hand in hand with higher fault-tolerance, a highly advantageous situation.

Based on the experiments of Sections 7.1 and 7.2, we recommend using GFR with BFRM embedding generation due to its simplicity and its relatively low tree redundancy. The dimension k of the embedding should be chosen high enough to achieve both low stretch and high fault-tolerance. However, how this parameter k can be estimated for unknown graphs remains future work.

7.3. *Backup Mechanism*

In Figure 16, the difference in stretch and success ratio between routing with and and without backup system (as presented in Section 7.3) is shown. According to Cvetkovski and Crovella [8], this type of backup system should obtain a success ratio of 100% in all cases, which is empirically verified in Figure 16b for our routing scenario. Figure 16a shows how failures affect routing stretch. It can be noticed that, as the number of failed links increases, the stretch increases simultaneously. When no backup mechanism is active, this is explained by the lowered chances of having a neighbor available that leads to a short path to the destination.

When the backup system is active, the increase in stretch can be explained by, on the one hand, the omission of paths with voids (which can be theorized as being $+\infty$) in the average stretch calculations, which biases the true stretch of routing without backup system. Conversely, the stretch increase can be explained by the fact that paths taken by the backup mechanism are generally longer than the paths taken by the default GFR system. This is caused by the inefficient visitation scheme imposed by the backup algorithm, which is required as the coordinates are no longer trustworthy.

Figure 17a shows the cumulative stretch distribution for different link failure rates with the backup system active. This shows that as the backup mechanism is used more often (which happens when more links are defect), a larger fraction of paths have a high stretch—the distribution seems to shift to the right. This supports the fact that the backup routing

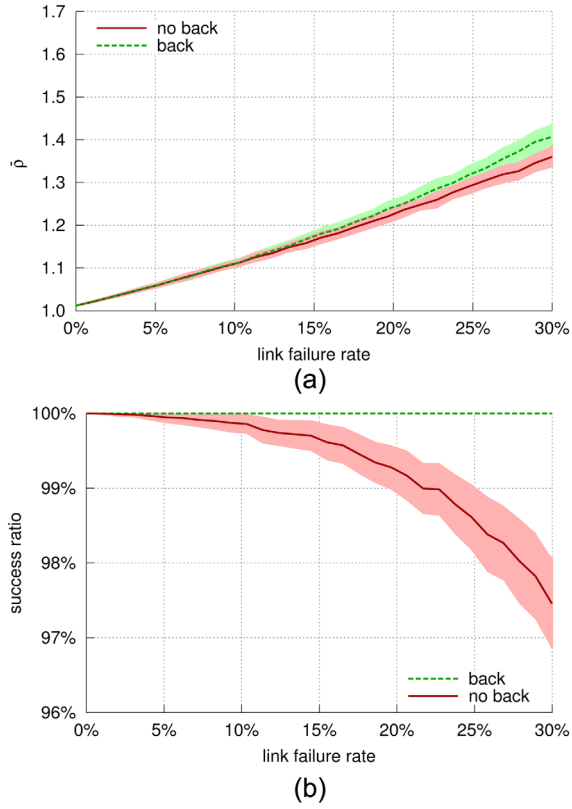


FIG. 16. Average stretch $\overline{\rho}$ and routing success ratio on a scale-free graph of 500 nodes, in function of the percentage of failed links, with and without the backup routing procedure active. The shaded background behind the curves represents $\overline{\star} \pm \sigma_\star$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

mechanism reroutes traffic along paths that are longer than paths generated by GFR in case of no failures. In Figure 17b the average stretch of the paths with the 1% highest stretch values is depicted, in function of a varying number of failed links. Here can also be noticed that the backup mechanism increases routing stretch, which is consistent with the previous experimental results.

7.4. *Embedding Regeneration Procedure*

Real networks are subject to failures and node mobility, resulting in a dynamic graph topology. However, the default GFR system cannot deal with such a changing topology because the underlying embedding loses its resemblance to the network, that is, the forest which induces the coordinates becomes disconnected and no longer spans the network. This leads to an increased stretch and ultimately to routing voids. To handle this, an embedding regeneration scheme has been added to the baseline GFR algorithm, as presented in Section 7.2. By swapping out those embeddings that least resemble the altered topology in favor of newer ones, this mechanisms maintains steady routing performance over time.

To accurately test this regeneration scheme, we require that the characteristics of the network remain constant while swapping links. It is, however, nontrivial to keep network

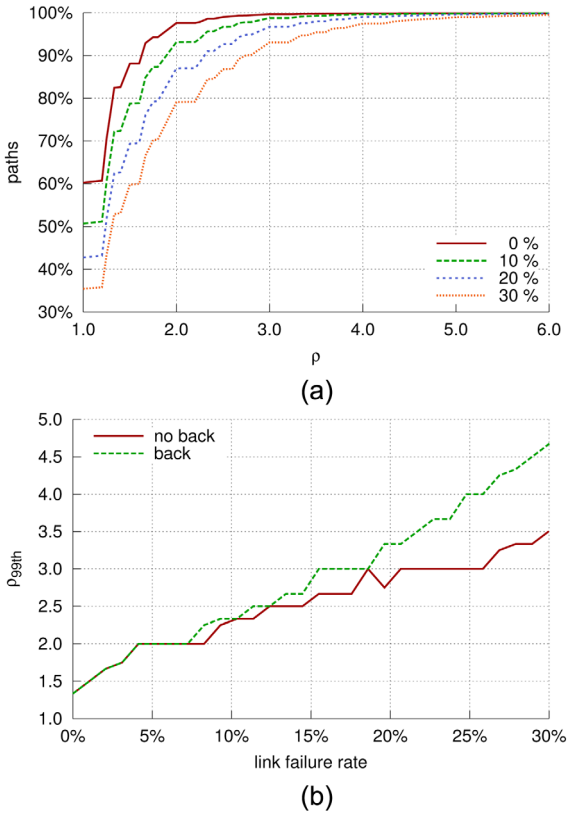


FIG. 17. Cumulative number of paths with a certain stretch ρ for different failure ratios, as well as the stretch ρ_{99th} of the 99th percentile with and without backup system for a scale-free network of 500 nodes. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

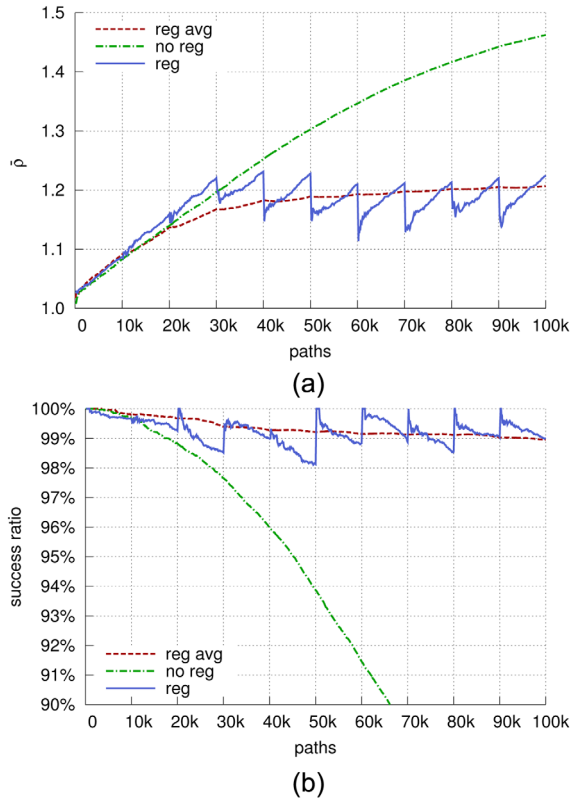


FIG. 18. Embedding regeneration for 10^5 paths generated, while every 100 paths a link is randomly swapped in a random graph of 500 nodes. Every 10^4 paths, 30% of the embeddings are regenerated, which can noticed by the saw-like trend (reg). Contrary to the red line (reg avg), which depicts the total average value (stretch and success ratio), the blue line (reg) resets each value at every embedding regeneration. Using no embedding regeneration procedure (no reg) makes the routing system vulnerable to dynamic topologies. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

characteristics stable in scale-free networks. As this experiment's only goal is to illustrate the effects of embedding regeneration, the type of benchmark network does not matter. Therefore, a random network of 500 nodes is used, because its network properties will not be altered by random link permutation. In the experiment random source-destination pairs are generated and every 100 pairs one link is swapped, which means its incident vertices are changed randomly. Every 10^4 generated paths, 30% of the 10 embeddings are regenerated with the embedding mode set to RM.

Figure 18 shows the results: Not using any regeneration scheme leads to a quick incline in average stretch as well as a quick decline in success ratio. When a regeneration model is used that regenerates 30% of the embeddings every 100 link permutations, it can be noticed that the stretch (Fig. 18a) and success ratio (Fig. 18b) converge toward a stable point. This allows the network to change indefinitely without deterioration of the routing behavior. Contrary to the red (reg avg) and green (no reg) line, the blue (reg) line shows the stretch and success ratio after they have been reset at each re-embedding. This better illustrates the true behavior of the routing scheme in combination with the embedding regeneration procedure. At every re-embedding, a step in the stretch and success ratio curve can be noticed. Hereafter, they both worsen again, until the next re-embedding. By increasing the re-embedding frequency it is possible to reduce the size of this step, however, this also results in higher communication overhead due to the coordinate assignment message exchanges. To conclude, results indicate that our embedding regeneration procedure can help in withstanding highly dynamic networks with continuously changing topologies.

8. CONCLUSION

In this article a theoretical framework is built which serves as the foundation of a geometric routing scheme called GFR. GFR performs greedy routing based on a multidimensional embedding induced by a spanning forest. Results show that using an embedding construction procedure that maximizes the forest's network coverage, both routing robustness and stretch are improved. The scheme is inherently robust toward network failures without using any form of path protection or restoration. It is capable of guaranteeing success ratios as high as 97% at link failure rates of 30% in scale-free networks.

Furthermore, the scheme can be extended with a backup routing mechanism, ensuring a 100% routing success rate, as

long as the network stays connected. Moreover, GFR is able to continuously adapt to a dynamically changing graph topology by means of a novel embedding regeneration scheme. Its low router storage complexity makes GFR robust toward network growth, while complexity analysis demonstrates the efficiency of its forwarding layer.

ACKNOWLEDGMENTS

This work was carried out using the Stevin Supercomputer Infrastructure at Ghent University, funded by Ghent University, the Hercules Foundation and the Flemish Government—department EWI. The open-source graph visualization framework Gephi [21] was used to visualize the different routing algorithms. Rein Houthooft is supported by a Ph.D. Fellowship of the Research Foundation – Flanders (FWO).

REFERENCES

[1] B. Karp and H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, The 6th Annual International Conference on Mobile Computing and Networking, Boston, USA, August 2000, pp. 243–254.

[2] M. Boguñá, F. Papadopoulos, and D. Krioukov, Sustaining the Internet with hyperbolic mapping, Nat Commun 1 (2010), pp. 62–68.

[3] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet, Robust geometric forest routing with tunable load balancing, The 34th Annual IEEE International Conference on Computer Communications (INFOCOM 2015), Hong Kong, P.R. China, April 2015, pp. 1382–1390.

[4] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet, Fault-tolerant greedy forest routing for complex networks, The 6th International Workshop on Reliable Networks Design and Modeling (RNDM 2014), Proceedings, Barcelona, Spain, November 2014, pp. 1–8.

[5] D. Chen and P. Varshney, A survey of void handling techniques for geographic routing in wireless networks, IEEE Commun Surveys Tutorials 9 (2007), 50–67.

[6] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, Link failure recovery technique for greedy routing in the hyperbolic plane, Comput Commun 36 (2013), 698–707.

[7] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, Single failure resiliency in greedy routing, The 9th International Conference on Design of Reliable Communication Networks (DRCN 2013), Budapest, Hungary, March 2013, pp. 306–313.

[8] A. Cvetkovski and M. Crovella, Hyperbolic embedding and routing for dynamic graphs, The 28th Annual IEEE International Conference on Computer Communications (INFOCOM 2009), Rio de Janeiro, Brazil, 2009, pp. 1647–1655.

[9] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz, Disjoint multipath routing using colored trees, Comput Networks 51 (2007), 2163–2180.

[10] M. Médard, S.G. Finn, and R.A. Barry, Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs, IEEE/ACM Trans Network 7 (1999), 641–652.

[11] G. Enyedi, G. Retvari, and A. Csaszar, On finding maximally redundant trees in strictly linear time, IEEE Symposium on Computers and Communications (ISCC 2009), Sousse, Tunisia, July 2009, pp. 206–211.

[12] J. Herzen, C. Westphal, and P. Thiran, Scalable routing easy as PIE: A practical isometric embedding protocol, The 19th annual IEEE International Conference on Network Protocols (ICNP 2011), Vancouver, Canada, October 2011, pp. 49–58.

[13] M. Tang, H. Chen, G. Zhang, and J. Yang, Tree cover based geographic routing with guaranteed delivery, 2010 IEEE International Conference on Communications (ICC 2010), Cape Town, South Africa, 2010, pp. 1–5.

[14] E. Chávez, N. Mitton, and H. Tejeda, Routing in wireless networks with position trees, Ad-Hoc, Mobile, and Wireless Networks, ser. Lecture Notes in Computer Science, 4686 (2007), 32–45.

[15] A. Korman, D. Peleg, and Y. Rodeh, Labeling schemes for dynamic tree networks, The 19th International Symposium on Theoretical Aspects of Computer Science, ser. Lecture Notes in Computer Science, 2285 (2002), 76–87.

[16] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, Geographic routing without location information, The 9th Annual International Conference on Mobile Computing and Networking, San Diego, USA, September 2003, pp. 96–108.

[17] R. Kleinberg, Geographic routing using hyperbolic space, The 26th Annual IEEE International Conference on Computer Communications (INFOCOM 2007), Anchorage, USA, May 2007, pp. 1902–1909.

[18] D. Eppstein and M. T. Goodrich, Succinct greedy graph drawing in the hyperbolic plane, In Graph Drawing, pp. 14–25, 2009.

[19] R. Cohen and S. Havlin, Scale-free networks are ultrasmall, Phys Rev Lett 90 (2003), 058701.

[20] A.-L. Barabási and R. Albert, Emergence of scaling in random networks, Science 286 (1999), 509–512.

[21] M. Bastian, S. Heymann, and M. Jacomy, Gephi: An open source software for exploring and manipulating networks, The 3rd International AAAI Conference on Weblogs and Social Media, San Jose, USA, May 2009.