Fault-tolerant Greedy Forest Routing for Complex Networks

Rein Houthooft, Sahel Sahhaf, Wouter Tavernier, Filip De Turck, Didier Colle, Mario Pickavet
{rein.houthooft, sahel.sahhaf, wouter.tavernier, filip.deturck, didier.colle, mario.pickavet}@intec.ugent.be
Department of Information Technology (INTEC), Ghent University - iMinds
Gaston Crommenlaan 8, 9050 Gent, Belgium

***Abstract*—Geometric routing has been proposed in literature as a memory-efficient alternative to traditional lookup-based routing and forwarding algorithms. However, existing geometric routing schemes lack the ability to address network link and node failures in a natural way, while maintaining a low path stretch. The main contribution of this paper is a novel routing scheme called Greedy Forest Routing (GFR) based on the principles of geometric routing. By employing a graph embedding based on low-redundancy spanning trees, its fault-tolerant characteristics are enhanced. Using a multi-dimensional tree embedding enables natural traffic redirection while still attaining a low average hop count.**

***Index Terms*—geometric routing; fault-tolerance; forest routing**

I. Introduction

For over a decade, geometric routing algorithms are emerging as an alternative to lookup-based routing schemes. Though they were originally designed for ad-hoc wireless networks and wireless sensor networks (WSNs) [1], researchers have demonstrated their applicability to wired networks as well [2]. This entails a different design approach because wired networks are generally modelled as scale-free graphs[1] rather than unit-disk graphs (UDGs)[2]. Geometric routing makes use of a graph embedding in which every network node is assigned coordinates corresponding to a point in a mathematical space. Using these coordinates, and a distance function, packets are routed towards the destination by following a distance-decreasing path.

In geometric routing a node only requires information about its local neighbourhood rather than the whole network. Therefore an advantage over a lookup-based approach is their capability of generating routing paths with limited router memory overhead. On the other hand, a clear disadvantage is the lack of routing fault-tolerance because the underlying embedding may lose its greedy characteristics upon the occurrence of network failures. This means that there no longer exists a distance-decreasing path towards the destination (also see Definition 1). Traffic redirection in case of failures is deemed essential in many large-scale communication networks. However, how low path stretch[3] and fault-tolerance can be effectively combined in geometric routing remains largely unsolved.

This paper investigates how geometric routing can combine low path stretch with reliable routing behavior. The main contribution is a geometric routing scheme called Greedy Forest Routing (GFR) which ensures a 100% packet delivery success rate. Furthermore it is highly resilient towards node and link failures by utilizing a multi-dimensional embedding based on low-redundancy[4] spanning trees. GFR is applicable to a wide range of network types, but is designed with scale-free complex networks in mind.

II. Related work

Many geometric routing algorithms target UDGs, rather than scale-free networks. It is however still useful to investigate what techniques these algorithms use to achieve fault-tolerance. In general, geometric routing suffers from so-called voids. These are nodes reached by a forwarded packet for which no distance-decreasing (to the destination) neighbour exists, which likely causes the packet to be dropped.

Several techniques exist for coping with failures or packet voids using a face routing-like construct [3]. But, these techniques are tied to UDGs and are hence not applicable to scale-free networks because these do not possess the required graph properties. Sahhaf et al. focus on finding alternate pathways in greedy hyperbolic embeddings for fast traffic re-routing, applicable to a wide variety of networks [4]. They also proposed the use of a backup tree to cope with single link failures [5]. The work in this paper differs from the former in the sense that it improves routing fault-tolerance by (i) using a multi-dimensional embedding, allowing for natural traffic re-routing, and (ii) generating spanning trees by minimizing tree redundancy in a heuristic manner.

A technique not relying on pre-determined alternate pathways is Gravity-Pressure routing [6]. When routing voids are encountered, a potential function is applied to the routed packet at every network node. This causes the packet to be steered away from the network failure. An advantage of this technique

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

[2]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.

[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.

[4]In this work, minimizing redundancy means minimizing the overlap of the edges of different trees. Trees whose edges completely overlap are maximally redundant, i.e., the removal of a tree does not affect the routing system.

is that it is proven to be 100% reliable. A severe disadvantage is that the stretch may increase arbitrarily. Some techniques focus on creating minimally-overlapping trees, however most remain hard to implement practically in a distributed fashion, or are restricted by a fixed number of trees [7], [8], [9]. Furthermore these algorithms do not attempt to construct embeddings with the geometric routing paradigm in mind. Our approach uses a tree construction mechanism that reduces redundancy in a heuristic manner. Herein parallel coordinates are assigned to allow for geometric routing. Tang et al. also use multiple trees for geometric routing [10], however they do not focus on – or test for – reliability. Furthermore, multiple trees are constructed in a breadth-first manner rather than aiming for low tree redundancy. Contrary to their approach, our method is capable of combining high fault-tolerance with low path stretch.

III. Theoretical foundation

In this section a theoretical foundation for the Greedy Forest Routing (GFR) algorithm is built based on the work of Chávez et al. [11] and Korman et al. [12]. Geometric routing systems make use of a graph embedding. Such an embedding is a mapping between vertices of a graph and a certain 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 $\forall u, v \in V : u \neq v \Leftrightarrow f(u) \neq f(v)$.
To guarantee a 100% routing delivery success rate a greedy graph embedding is mostly required, 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 $\delta(f(v), f(w)) < \delta(f(u), f(w))$.
Herein a metric space is the double formed by a space and its complementary distance function, more formally defined by Definition 3. This metric space is used to steer packets towards 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) $\delta(u, w) \leq \delta(u, v) + \delta(v, w)$

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 an embedding. This embedding makes use of a vertex labelling procedure [12] and a distance function representing the shortest path length in T [11]. Here the embedding target space S is denoted as the *tree space* $\mathbb{T}$. This tree space is defined as

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

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 labelling procedure (see Section VI) are now interpreted as coordinates in $\mathbb{T}$. These coordinates form a greedy embedding [11] which is denoted as $\mathcal{T}$. Therefore each vertex $v \in V$ corresponds to a point in $\mathbb{T}$ identified by the coordinates $\mathcal{T}(v)$.[5] An example of this embedding can be found in Figure 1(a). The distance function δ is defined as

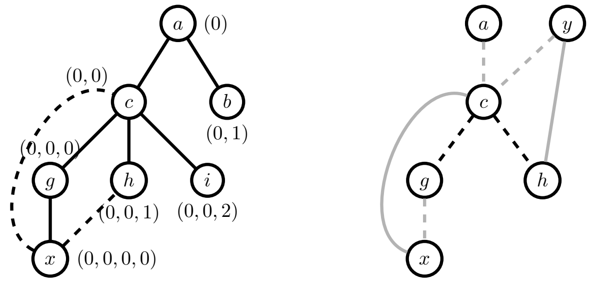
$$\delta(u, v) = |u| + |v| - 2|\phi(u, v)| \tag{2}$$

with $\phi : \mathbb{N}^n \times \mathbb{N}^m \to \mathbb{N}^*$ the function which 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). Now this metric space can be used according to the principles of geometric routing. This means every vertex is aware of the coordinates of its neighbourhood 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.

IV. Multi-embedding

A first step towards 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 $\mathbb{T}$, instead of only one [10]. These different greedy embeddings are denoted as $\mathcal{T}_i$. The notation δ is now used to denote the k-tuple of distances in each of the k embeddings. Therefore δ_i represents the tree space distance for the i-th embedding $\mathcal{T}_i$. Furthermore, assume a graph $G = (V, E)$ in which a spanning tree $T = (V, E')$ has been constructed. For a link $e \in E$ either $e \in E'$, which means that e is a critical link (solid lines in Figure 1(a)), or $e \notin E'$, which means that e is called a shortcut link (dashed

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



(a) Tree T (solid edges, root a) forming embedding $\mathcal{T}$ into $\mathbb{T}$ by labelling each node. Failure of dashed edges or nodes x or h does not result in failures.

(b) Tree T_1 (root a, dashed edges) and T_2 (root y, gray edges) minimizing redundancy. Notice the avoidance of overlapping edges.

Figure 1. Graph embeddings illustrated: shortcuts, labelling, redundancy.

lines in Figure 1(a)). When a shortcut link or a leaf node (nodes x and h in Figure 1(a)) fails, routing is still possible for all source-destination pairs since the underlying spanning tree T is still intact. On the other hand, when a critical link or non-leaf node fails, voids may occur for certain node pairs.

A benefit of using multiple embeddings is the increased fault-tolerance of the routing scheme towards node and link failures. As more embeddings are available, the chances of a link failure disrupting each of the embeddings becomes lower. To illustrate with an example (see Figure 1(b)): link (h, y) is a critical parent-child link in embedding $\mathcal{T}_2$ but is a non-essential shortcut link in $\mathcal{T}_1$. The same holds for the network nodes: non-leaf 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 $\mathcal{T}_1$.

To have access to a maximum number of shortcuts, 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 $E' = E''$. Because of this 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 1(b) 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 in many of the trees is likely to cause routing voids[6]. How to achieve low tree redundancy is explained in Section VI.

V. Greedy Forest Routing (GFR)

In this section we will see how multiple embeddings can be combined with the theoretical foundation presented in the Section III. A straightforward way of routing using multiple embeddings would be to allow a vertex to freely alternate between the different embeddings, due to their individual greediness. However, this naive forwarding mechanism is not reliable because routing cycles may be introduced: routing along a distance-decreasing path in $\mathcal{T}_i$ may increase the distance in $\mathcal{T}_j$. 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, an example is given for a multi-embedding consisting of three embeddings, thus $k = 3$, which is depicted in Figure 2. Assume a source node s and a destination node d for which $\delta(s, d) = (3, 5, 7)$. This means that the δ-distance is 3 in embedding $\mathcal{T}_0$, 5 in $\mathcal{T}_1$, and 7 in $\mathcal{T}_2$. When routing naively the following scenario can occur:

- $\delta(n_1, d) = (3, 4, 7)$, s decides to route to n_1 in $\mathcal{T}_1$
- $\delta(n_2, d) = (3, 6, 6)$, n_1 decides to route to n_2 in $\mathcal{T}_2$. However, the distance in $\mathcal{T}_1$ increases from 4 to 6.
- $\delta(n_3, d) = (3, 5, 7)$, n_2 decides to route to n_3 in $\mathcal{T}_1$. Notice that the distance in $\mathcal{T}_2$ increases from 6 to 7.
- $\delta(n_1, d) = (3, 4, 7)$, n_3 decides to route to n_1 in $\mathcal{T}_1$

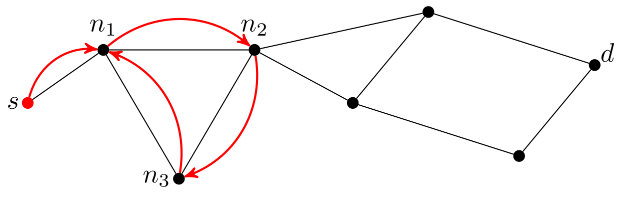


Figure 2. Naive routing with multiple embeddings

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 [10]. In our work a new distance function $\epsilon : \mathbb{T}^k \times \mathbb{T}^k \to \mathbb{R}^+$ is defined as

$$\epsilon(u, v) = \min_{0 \le i < k} \{\delta_i(u, v)\} \ \ \forall u, v \in V \tag{3}$$

The k embeddings into $\mathbb{T}$ can now be treated as a single k-dimensional greedy[7] embedding into $\mathbb{T}^k$. As such the principles of geometric routing can be followed by using the relaxed metric space $(\mathbb{T}^k, \epsilon)$. This double cannot be regarded as a true metric space as the triangle-inequality no longer holds (Definition 3, property 3). When forwarding, multiple neighbours may have an equal ϵ-distance. In this case a random choice will be made among them. This ϵ-based routing scheme, in combination with the embedding procedure presented in the next section, is called Greedy Forest Routing (GFR).

VI. Embedding construction

In this section the details of the GFR tree construction mechanism for generating graph embeddings are explained. By constructing the different trees underlying the embedding in an intelligent manner, GFR is inherently robust towards network failures. A prerequisite for the embedding construction phase is the assignment of an identification number to each node. This is required for having a deterministic solution to the election procedure presented next. This identification number may be chosen freely, e.g., the unique MAC-address of a node. 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 root nodes $\{r^{(i)}\}$ are elected. Secondly, the spanning trees T_i are constructed along with the corresponding greedy graph embeddings $\mathcal{T}_i$. In what follows the construction of a single embedding $\mathcal{T}$ will

[6]Routing voids are nodes which are unable to forward a packet because there exists no distance-decreasing neighbour.

[7]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 $\delta(u, d)$ in 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).

be explained, multiple embeddings can be seen as a parallel extension. The embedding procedure itself is exactly the same as explained by Chávez et al. [11] and Korman et al. [12]. The difference lies within the way the trees are formed.

First, the root node r is assigned the coordinate (0). Next it will send its coordinates to its neighbours $N_1(r)$.[8] 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 neighbours $N_2(r)$ with a similar message. In turn these will recursively inform the nodes $N_j(r)$. The final result can be seen in Figure 1(a). When 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 growing modes:

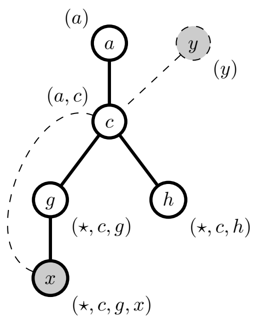
- *breadth-first mode* (BFM): override the previous coordinate tuple only this leads to a tuple of lower length. This results in a tree of minimal depth.
- *redundant mode* (RM): only override previous coordinates when their cost function value is lower. This cost function is based on the coordinate tuple lengths and the overlap of the spanning trees (tree redundancy).
- *breadth-first redundant mode* (BFRM): a hybrid mode combining BFM with RM.

In BFM coordinates are only overridden when the coordinate tuple resulting from the received message has a lower length than the current tuple, or its parent node has acquired a new set of coordinates. In the latter case, the child node should be updated accordingly. By doing so, 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 fault-tolerance: 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 redundant mode (RM). In RM the emphasis lies on minimal tree redundancy, attaining maximal tree spread. This is done by making sure that every link is an edge in approximately the same number of trees. For this, a 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.

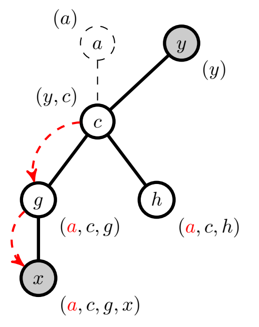
Definition 4: Assume k spanning trees. Denote the number of different sets E'_i, corresponding to $T_i = (V, E'_i)$ with $0 \leq i < k$, that an edge $e \in E$ in a graph $G = (V, E)$, belongs to as n. The amount of tree redundancy, termed the τ-ratio, is defined as the standard deviation $\sigma(n)$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

A low τ-ratio indicates that the trees are spread evenly over the network while a high τ-ratio indicates that there is a

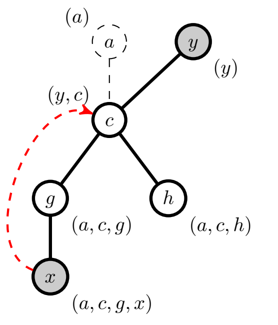
[8] $N_i(u)$ represents the i-th hop neighbours of u.



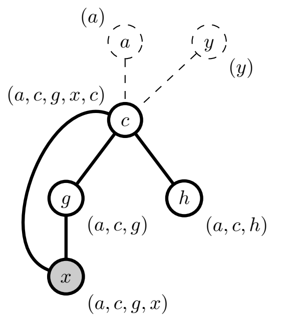
(a) The tree (a, c, g, h, x) exists after a has assigned coordinates to c. This leads to updating g, h and x. Hereafter, the parent a of c is replaced by y because of a decreasing cost.



(b) Node c updates its coordinates according to its new parent y. The tree becomes (y, c, g, h, x). At the same time the update message of the old coordinates of c reaches x.



(c) Node x sends the updates to all of its neighbours, which includes c. It does not notice that c is an ancestor because the coordinates of c have already been updated according to its new parent y.



(d) Node c accepts the new coordinates of x because of a decreasing cost function. However, this was still based on the non-updated coordinates and thus a cycle is formed, corrupting the tree.

Figure 3. 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 1(a).

huge difference in the number of spanning trees a link of the network is part of. Also note that $\tau \geq 0$. To decrease the tree redundancy while constructing the embedding, a cost function $f_i : V^3 \to \mathbb{R}$ was defined for each embedding $\mathcal{T}_i$ in the system:

$$f_i(u, p, p') = \eta a_i(u) + \beta b_i(u) + c_i(u, p, p') \tag{4}$$

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

$$a_i(u) = |u'| - |u| \tag{5}$$

$$b_i(u) = \begin{cases} 0 & \text{if } a_i(u) \leq 0 \\ |u'| - |u^*| & \text{if } a_i(u) > 0 \end{cases} \tag{6}$$

$$\begin{aligned} c_i(u, p, p') = &\\ &|\Psi(u, p') - \bar{\Psi}(u) + 1| + |\Psi(u, p) - \bar{\Psi}(u) - 1| \\ &- \big(|\Psi(u, p') - \bar{\Psi}(u)| + |\Psi(u, p) - \bar{\Psi}(u)|\big) \end{aligned} \tag{7}$$

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 $(x,y) \in E$; $\bar{\Psi}(x)$ 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. $b_i(u)$ 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 $e \in I(u)$[9] equalizes or not (this is our goal, making sure that every link is part of an equal number of trees, to minimize τ). In this cost term, $|\Psi(u,p') - \bar{\Psi}(u) + 1|$ describes the offset of the number of trees link (u,p') is part of versus the average of this value. This average is denoted as $\bar{\Psi}(u)$, and is calculated over all links in $I(u)$, when p' is chosen as the new parent of u. Furthermore the term $|\Psi(u,p) - \bar{\bar{\Psi}}(u) - 1|$ 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 $\bar{\bar{\Psi}}(u)$. The sum of these two offsets is compared to the sum of the offsets without parent replacement, which is $\big(|\Psi(u,p') - \bar{\Psi}(u)| + |\Psi(u,p) - \bar{\Psi}(u)|\big)$. 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 $-2 \le c_i(u,p,p') \le 2$.

The graph embedding procedure in RM is described in Algorithm 1. When growing a tree in RM, precautions have to be taken in order to avoid the introduction of cycles, especially when using low η-values in Eq. (4) as nodes will then frequently switch parents. The fact that now longer coordinates may override shorter ones is the main reason for the potential introduction of cycles. 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 $|\phi(u',u)| \ne |u|$ holds (see Eq. (2)). But even then it is still possible to generate cycles as explained in Figure 3. 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 Figure 3(a)). 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*.

Nonetheless, even with the cycle avoidance active, cycles may still be introduced due to the system's distributed nature. Therefore, a *cycle resolution* procedure was implemented, shown in Algorithm 1 at line 8. 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 3(d). Hence it is informed of the cycle $\langle c,g,x,c \rangle$. Next, c will send on the coordinate message to its children in order to notify them. Whenever a node detects a cycle, it searches for a new parent among its neighbours 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, resolving the cycle.

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

Algorithm 1: Tree growing: redundant mode (RM)

```
input  : vertex $u$ has sent its coordinates to its
         neighbours; current 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 u, a child number $c_u$ and key $\mathcal{K}(u)$
2  look up own coordinates v and parent key $\mathcal{K}(p)$
   corresponding to embedding $\mathcal{T}$
3  if $v = \emptyset$ 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 $\mathcal{K}(v) \in P \wedge \mathcal{K}(u) \ne \mathcal{K}(p)$ 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 $v \leftarrow u^\frown\{c_u\}$
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 v and neighbour
               number $c_v$ to $n$
17 else
18     drop packet
```

The advantage of generating an embedding in RM is the low resulting 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 was established by combining BFM with RM, called breadth-first redundant mode (BFRM). Now a tree of minimal depth is combined with a cost function to minimize τ. This basically comes down to assigning parameter values $\eta > 2$ and $\beta = 0$ in Eq. (4). 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.

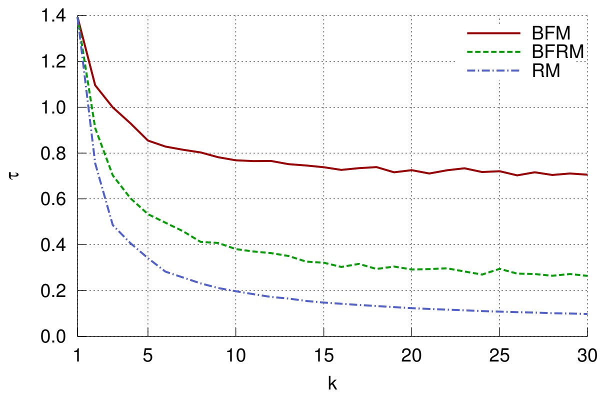


Figure 4. Tree redundancy τ in function of the dimension k (the number of embeddings) for different tree generation modes on a scale-free (see Section VII) graph with 500 nodes.

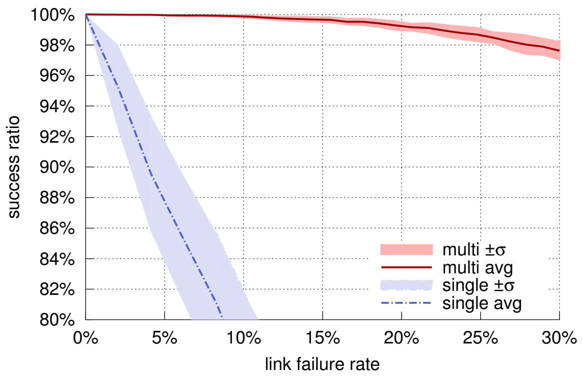


Figure 5. Fault-tolerance: the success ratio of GFR ($k = 15$, BRFM embedding generation) was tested for a varying failure rate. Also GFR with only a single embedding was evaluated. The shaded background represents the average success ratio, plus and minus the standard deviation σ.

Because of its breadth-first behaviour, no cycle avoidance or resolution is required, which greatly diminishes the complexity of the embedding procedure. This results 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 4. For RM, the values in Eq. (4) were set to $\eta = \frac{2}{3}$ and $\beta = \frac{1}{3}$, which led to good behavior (no excessively deep trees in which lots of cycles had 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. In the next section we will show that a lower τ-ratio increases the fault-tolerance of GFR.

VII. Results and discussion

The routing behavior was simulated by a programmatic routing framework in which random source-destination host pairs were generated continuously, corresponding to a uniform traffic matrix. Between these pairs traffic was simulated by generating routing paths in a multi-threaded environment. All experiments were executed on a High Performance Computer. The algorithms were tested on differently sized scale-free networks generated according to the Barabási-Albert model [13] with a degree distribution of $P(d) \propto d^{-\gamma}$ with $\gamma = 2.2$.

A. Fault-tolerance

In this section the fault-tolerance of GFR will be evaluated. When using multiple embeddings, the chance of a network failure disrupting each of them becomes lower. Hence, the routing system is more likely to find a distance-decreasing neighbour send on the packet. Therefore GFR should be more resilient to failures than geometric tree-based routing using a single embedding.

We only describe the effects of link failures in this section due to space constraints, however the node failure results are very similar. In the following experiment, the number of links removed starts at 0 and goes up by 20 at each step. 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 or nodes failed, the experiment is run 100 times and every time 1000 random source-destination pairs are generated between which uniform traffic is simulated (as such 10^5 paths are simulated for each step). For each of these 1000 pairs the average success ratio is examined. To test link fault-tolerance, links are removed with a probability $p(l) = \exp\left(3\left(1 - \frac{d_G(v)_o}{(d_G(v)-1)}\right)\right)$, with $v \in V$ the vertex that has the lowest $p(l)$ probability of both vertices incident to $l \in E$; $d_G(v)_o$ the degree of vertex v before removal of any links. This probability makes sure that the link failures are evenly spread 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 5 depicts the routing success rate of GFR with $k = 15$ and routing with a single embedding, exercised on a scale-free graph with 500 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 (success ratio less than 50% at a 30% failure rate). This shows that GFR is able to achieve high fault-tolerance without using any form of protection or restoration procedure, which is frequently used [4], [5], [6].

This inherent fault-tolerance can be explained by the availability of more embeddings. Because of this, more alternate pathways between source and destination exist. Therefore it is less likely that a routing void will occur, a situation in which no new ϵ-decreasing neighbour can be found. From this can be concluded that GFR is fault-tolerant by its very nature (without needing path restoration/protection), which is essential in a large-scale network where link or node failures are common.

Figure 6 shows how increasing the dimension k affects GFR's fault-tolerance at a link failure rate of 30%. In this experiment, for each k value, 15 different runs were executed in which 10^5 random source-destination pairs were generated between which traffic was simulated. At low k-values, the

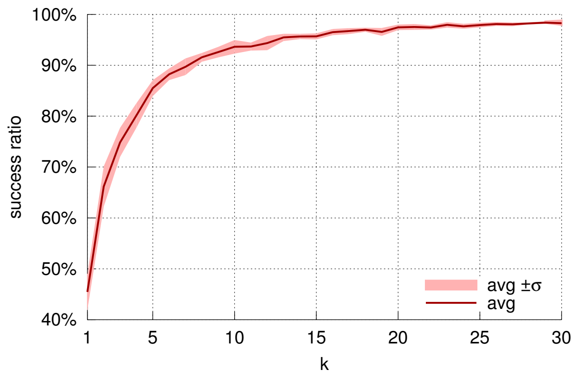


Figure 6. Success rate at a link failure rate of 30% for a varying dimension k on a scale-free graph with 500 nodes. The embedding was generated according to BFRM. The shaded background represents the average success ratio, plus and minus the standard deviation σ.

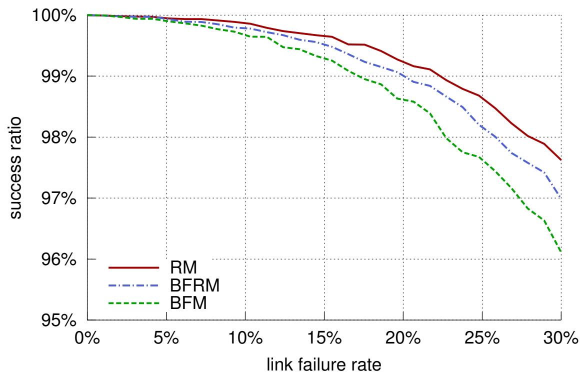


Figure 7. Effect of the different graph embedding procedures on the fault-tolerance of GFR ($k = 15$) on a scale-free graph with 500 nodes.

success ratio of GFR is low, which is consistent with Figure 5. As k increases, so does the success ratio, until it converges towards 100% as $k \to +\infty$. This is consistent with more embeddings allowing for more alternative paths by which packets are naturally re-routed to their destination.

The effect of the different graph embedding procedures on the success ratio is shown in Figure 7. It can be noticed that redundant mode (RM) is more robust to failures than breadth-first mode (BFM) while breadth-first redundant mode (BFRM) lies somewhere in between. This indicates the correlation of tree redundancy (see Definition 4) with fault-tolerance. The τ-ratios of the different embedding modes can be seen in Figure 4 for varying values of k. This result is important as the gain in fault-tolerance does not require any alteration of the routing procedure, it is entirely dependant on the initial embedding procedure.

B. Stretch

The fault-tolerant behavior of GFR should not lead to excessively long routing paths. Therefore, in this section GFR is analyzed regarding its average stretch $\bar{\rho}$. We define the stretch of a generated path between a source node s and a destination node d as the path length, in number of hops, divided by the length of the shortest path between s and d.

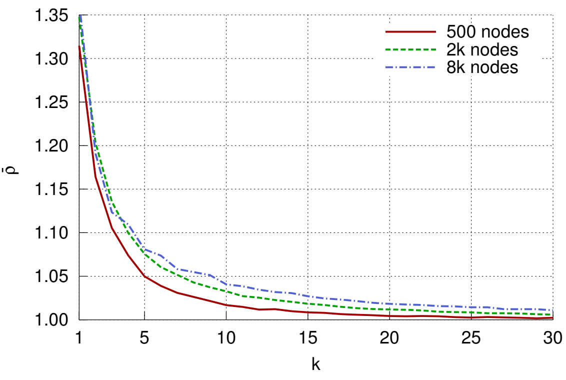


Figure 8. GFR with BFRM embedding generation: average stretch $\bar{\rho}$ in function of the dimension k of the embedding space $\mathbb{T}^k$ for differently sized scale-free graphs.

In Figure 8 the effect of changing the number of embeddings k (or the dimension of the space $\mathbb{T}^k$) on the average stretch $\bar{\rho}$ is depicted. For each k value, 15 different runs were executed in which 10^5 random source-destination pairs were generated between which traffic was simulated. The experiment shows asymptotic behaviour $\bar{\rho} \to 1$ as $k \to +\infty$. This means that as k increases, GFR starts to approximate shortest path routing. A logical explanation is that the availability of more embeddings allows for more routing freedom. As a result, there is an increased chance that a combination of embeddings will lead to a short path between two nodes. Therefore attaining a lower stretch by increasing k goes hand in hand with higher fault-tolerance.

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

VIII. Conclusion

In this paper a theoretical framework was built which serves as the foundation of a geometric routing scheme called Greedy Forest Routing (GFR). GFR performs greedy routing based on a multi-dimensional embedding that has been constructed with minimal tree redundancy. It has been empirically shown to combine a low path stretch with robust fault-tolerance behavior, especially when compared to a single-dimensional embedding. GFR enables natural traffic redirection without using any form of path protection or restoration mechanism. It is capable of guaranteeing success ratios as high as 97% at link failure rates of 30% in scale-free networks.

Furthermore, due to the algorithm's local routing decision making procedure no routing tables are needed, making it highly scalable regarding memory requirements and robust towards network growth. Future work will encompass the addition of load balancing characteristics to the routing decision engine and estimating an appropriate embedding dimension for unknown networks.

Acknowledgment

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. This work is partly funded by the European Commission through the EULER project (grant 258307), part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7). This project was partly funded by the UGent BOF/GOA project "Autonomic Networked Multimedia Systems".

References

[1] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 243–254.

[2] M. Boguñá, F. Papadopoulos, and D. Krioukov, "Sustaining the Internet with hyperbolic mapping," *Nature Communications*, vol. 1, no. 62, Oct 2010.

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

[4] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "Link failure recovery technique for greedy routing in the hyperbolic plane," *Comput. Commun.*, vol. 36, no. 6, pp. 698–707, Mar. 2013.

[5] ——, "Single failure resiliency in greedy routing," in *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference*, March 2013, pp. 306–313.

[6] A. Cvetkovski and M. Crovella, "Hyperbolic embedding and routing for dynamic graphs," in *INFOCOM 2009, IEEE*, 2009, pp. 1647–1655.

[7] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz, "Disjoint multipath routing using colored trees," *Comput. Netw.*, vol. 51, no. 8, pp. 2163–2180, Jun. 2007.

[8] 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. Netw.*, vol. 7, no. 5, pp. 641–652, Oct. 1999.

[9] G. Enyedi, G. Retvari, and A. Csaszar, "On finding maximally redundant trees in strictly linear time," in *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, July 2009, pp. 206–211.

[10] M. Tang, H. Chen, G. Zhang, and J. Yang, "Tree cover based geographic routing with guaranteed delivery," in *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1–5.

[11] E. Chávez, N. Mitton, and H. Tejeda, "Routing in wireless networks with position trees," in *Ad-Hoc, Mobile, and Wireless Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4686, pp. 32–45.

[12] A. Korman, D. Peleg, and Y. Rodeh, "Labeling schemes for dynamic tree networks," in *STACS 2002*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2285, pp. 76–87.

[13] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.