Robust Geometric Forest Routing with Tunable Load Balancing

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

***Abstract*—Although geometric routing is proposed as a memory-efficient alternative to traditional lookup-based routing and forwarding algorithms, it still lacks: i) adequate mechanisms to trade stretch against load balancing, and ii) robustness to cope with network topology change.**

The main contribution of this paper involves the proposal of a family of routing schemes, called Forest Routing. These are based on the principles of geometric routing, adding flexibility in its load balancing characteristics. This is achieved by using an aggregation of greedy embeddings along with a configurable distance function. Incorporating link load information in the forwarding layer enables load balancing behavior while still attaining low path stretch. In addition, the proposed schemes are validated regarding their resilience towards network failures.

I. Introduction

Geometric routing schemes are proposed as an alternative for lookup-based routing algorithms. Although they were initially designed for Unit Disk Graphs (UDGs)[1] [1], their application to scale-free[2] complex networks has been demonstrated [2]. This form of routing makes use of a *graph embedding*, the assignment of coordinates in a mathematical space to every network vertex. This embedding, together with an appropriate distance function, forms the core of geometric routing, allowing packets to be transmitted along a distance-decreasing path towards their destination.

The main advantage of geometric routing is its low state complexity. A node only requires information about its neighbors, rather than being dependent on the state of the whole network. In contrast to more traditional routing schemes based on lookup tables, geometric routing thus restricts the required router memory overhead. A large disadvantage, however, is their lack of load balancing characteristics, which is essential in avoiding traffic congestion in large-scale networks. Lookup-based schemes can easily add this by incorporating multiple alternative routes in their lookup tables. How load balancing characteristics and stretch[3] can be combined and traded off in geometric routing is still an open research question.

[1]A graph $G = (V, E)$ is a unit-disk graph if $\forall u, v \in V : v \in N(u) \Leftrightarrow \delta(u, v) \leq 1$ when 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 (the number of hops along the path) divided by the shortest path length between its source and destination nodes.

In this work we explore the possibilities in combining low stretch with load balancing behavior. Our main contribution is a family of routing schemes called Forest Routing (FR). These algorithms are capable of adapting their routing behavior to varying traffic intensities by using a generalized distance function incorporating link load information. Additionally, in the absence of network failures they attain a 100% success ratio, while having a high resiliency to node and link failures. Although designed for, Forest Routing is not restricted to complex scale-free networks.

II. Related work

In general, load balancing mechanisms can be divided into two classes: *active* and *passive* techniques. Passive load balancing draws from the inherent structure of the routing algorithm to spread traffic equally over the network. Active load balancing, on the other hand, makes use of live traffic information to steer traffic away from hotspots. As a result, active load balancing techniques are able to adjust forwarding to a changing traffic matrix.

Many load balancing techniques used in geometric routing are designed for Wireless Sensor Networks (WSNs) and aim at lowering the number of overloaded nodes to slow down battery depletion. As such, most load balancing research is focused on node load balancing rather than link load balancing. Another issue is the focus on UDGs, which act as a model for WSNs, rather than more general topologies. Zeng et al. [3] use a routing scheme in which information about a node's energy level is used in the forwarding layer. In Curve-Based Greedy Routing [4] traffic is guided by a B-spline which is calculated by the source node and stored in the packet headers. By using a selection mechanism based on the node distances to this B-spline, and energy levels, there is more forwarding decision freedom than there is in basic geometric routing mechanisms.

LBLSP [5] uses a non-Euclidean routing scheme based on curves around obstacles, and targets wireless networks. Traffic hotspots are modelled as virtual objects that are avoided by the routing algorithm. How this can be implemented is, however, not specified. Curveball Routing [6] and Circular Sailing Routing [7] focus on passive load balancing. These routing schemes make use of a stereographic projection of 2D Euclidean coordinates onto a sphere. The authors report that a common problem in UDG networks is traffic congestion at

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

the center of the network under a uniform traffic matrix. Due to the lack of a sphere center, these systems naturally avoid this center hotspot.

Another way of balancing traffic passively is using multiple spanning tree-based embeddings [8]. By using multiple trees, the root hotspot effect, which is a common problem in geometric routing with tree-based embeddings [9], is avoided. However, as the authors target wireless networks, they do not incorporate link load balancing. The tests are also limited to small unit-disk networks. Virtual Polar Coordinate Routing [9] makes use of smart routing to attain passive load balancing. Smart routing is based on a spanning tree-based embedding and requires the network to be a UDG. Furthermore, the spanning tree has to be ordered according to an embedding into the 2D Euclidean plane. This makes it difficult to apply the routing scheme to non-UDG networks.

Other work focuses on geometric routing in scale-free networks rather than UDG networks, however, none of it truly aims for load balancing behavior. A hyperbolic embedding based on a hyperbolic tiling in the Poincaré disk model was originally proposed by Kleinberg [10]. Hyperbolic geometric routing is able to achieve high success ratios in scale-free complex networks [11], [12]. A recent work on routing complex networks is the Practical Isometric Embedding Protocol [13] in which Herzen et al. focus especially on the scalability of their routing algorithm.

III. Theoretical foundation

This section sets forth the theoretical foundation on which the Forest Routing (FR) algorithms are built. As mentioned in the introduction, geometric routing is based on the concept of a *graph embedding*. This is a mapping between a mathematical space and the vertices of a graph, formally defined by the following definition.

Definition 1: Let S be a set and δ a *metric function* (see Definition 3) 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 $\forall u, v \in V : u \neq v \Leftrightarrow f(u) \neq f(v)$. [14]

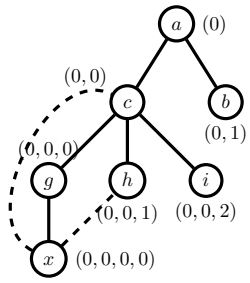
In order to guarantee a 100% routing delivery success rate, most geometric routing algorithms employ a *greedy graph embedding*, defined by the following definition.

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 $\delta(f(v), f(w)) < \delta(f(u), f(w))$. [10]

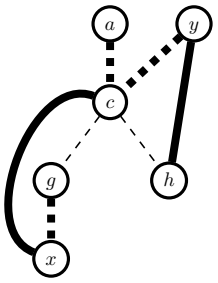
A *metric space* is the double formed by the space S and its complementary distance function δ, formally defined by the next definition. Geometric routing requires this metric space to send packets towards their target vertex along a distance-decreasing path.

Definition 3: Assume 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) $\delta(u, w) \leq \delta(u, v) + \delta(v, w)$,



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



(b) Tree T_1 (root a, dashed edges) and T_2 (root y, thick edges) minimizing tree redundancy, which is the overlap of edges of the different trees.

Figure 1. Graph embeddings illustrated: shortcuts and labeling.

then δ is called a metric, or a distance function, and the double (S, δ) is called a metric space.

The FR algorithms employ a spanning tree $T = (V, E')$ of the underlying graph $G = (V, E)$ to construct an embedding by making use of the vertex labeling procedure described by Korman et al. [15] and a metric representing the shortest path length in T, in number of hops, as described by Chávez et al. [16]. The embedding target space S is denoted as the *tree space* $\mathbb{T}$, 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. Now, we can interpret the assigned labels as coordinates in the space $\mathbb{T}$. We say that these coordinates form a greedy graph embedding [16], denoted as $\mathcal{T}$. As such, each vertex $v \in V$ corresponds to a point in $\mathbb{T}$ identified by the coordinates $\mathcal{T}(v)$.[4] An illustration of an embedded network can been found in Figure 1(a).

The distance function δ for $\mathbb{T}$ 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 returns the largest common prefix of two tuples; $|u|$ is the length of the coordinate tuple of vertex u. This leads to a distance function $\delta : \mathbb{T} \times \mathbb{T} \to \mathbb{R}^+$ that, combined with the space $\mathbb{T}$, forms the metric space[5] $(\mathbb{T}, \delta)$. Now, this metric space can be used according to the principles of geometric routing. This means that every vertex knows the coordinates of its neighbors and the coordinates of the target vertex are encoded in each packet header. As such, using this header, each node forwards packets along a distance-decreasing path to their destination.

Note that this type of routing is not the same as routing on the tree $T = (V, E')$ itself. Edges that are not part of T

[4]Rather than 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.

[5]The properties of a metric space can easily be proven.

may be used as *shortcut links*, which are links for which holds $e \in E \wedge e \notin E'$, in a graph $G = (V, E)$. Such links are shown in Figure 1(a) as dashed lines while edges in E' are shown as solid lines. The existence of these shortcut links allows for more freedom in the routing decision making. They can help in avoiding congestion at lower tree depths, which is a downside of geometric routing based on tree-based graph embeddings [9]. In general, congestion worsens as we go higher up in the tree, because a node will forward traffic to its parent in case no other neighor closer to the destination exists. Unless there are many shortcut links, this is likely the case. An illustration can be found in Figure 1(a). Say traffic is sent from x to b, the routing path will be $\langle x, c, a, b\rangle$. Hence, all traffic will traverse the root node a, congesting the links close to a. However, if a shortcut link exists, e.g., (c, b), some of the traffic could be offloaded to this link.

IV. Multi-embedding

In order to achieve load balancing, we use k spanning trees $T_i = (V, E'_i)$ with $\forall i \in \{0, 1, \ldots, k-1\} : E'_i \subseteq E$ to form k different greedy embeddings of $G = (V, E)$ into $\mathbb{T}$, a multi-embedding [8]. 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. Hence, δ_i represents the tree space distance for the i-th embedding $\mathcal{T}_i$. Using multiple embeddings suggest that we could alternate between them as a packet is sent to its destination. As such, for example, we can route a packet via geometric routing based on the coordinates in embedding $\mathcal{T}_1$, until it encounters a void[6], created by node or link failures. At this void we can switch to embedding $\mathcal{T}_2$, allowing the packet to reach its destination, which is why routing with multiple embeddings is more robust than with a single embedding. Similarly, a packet may be routed via a different embedding when this allows for an alternative less congested edge to achieve load balancing.

As more embeddings are available, the chances of a link or node failure disrupting all of them decreases. Let $G = (V, E)$ be a graph 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 a shortcut link (dashed lines in Figure 1(a)). When a shortcut link—or a leaf node—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.

To illustrate with an example: link (x, c) in Figure 1(b) is a critical parent-child link in embedding $\mathcal{T}_2$, while being a non-essential shortcut link in $\mathcal{T}_1$. The same holds for the network nodes: non-leaf node g in $\mathcal{T}_1$ is a leaf node in $\mathcal{T}_2$. If either (x, c) or g fails, routing in $\mathcal{T}_2$ and $\mathcal{T}_1$ respectively may fail because $\mathcal{T}_2$ or $\mathcal{T}_1$ loses its greedy property, but it is still guaranteed in the other embedding.

[6]A void is a node along a routing path by which the packet can no longer be send on, according to geometric routing, to the destination because no distance-decreasing neighbor exists [1].

Additionally, by using a multi-embedding it is possible to avoid traffic congestion at nodes and links residing at lower depths in the tree. The reason is two-fold:

1) Because of the existence of k different root nodes, the traffic bottleneck is now divided amongst them.
2) The chances of having shortcuts available increases, which leads to a less crowded root and an overall lower stretch.

As such, we attain passive load balancing. Active load balancing can be acquired by actively switching between the different embeddings by using link load information, which will be explained in Section V-B. The sacrifice made here is the increased storage requirements of the packet headers and the increased computational complexity of the forwarding layer. This latter can however be mitigated by the high parallelizability of the forwarding decision making procedure, namely, the different distances δ_i can be calculated in parallel.

To conclude, the benefits of using multiple embeddings over a single embedding are: i) passive load balancing behavior, ii) the possibility of active load balancing behavior, and iii) increased robustness towards routing failures. The exact embedding procedure, focusing on reducing the redundancy of the different spanning trees by making sure that their edges overlap as little as possible, can be found in [17].

V. Routing variants

A. Greedy Forest Routing (GFR)

A straightforward way of routing with multiple embeddings is to allow each vertex to alternate freely between them, making use of their individual greediness. However, this naive forwarding mechanism is unreliable because it can introduce routing cycles. Routing along a distance-decreasing path in $\mathcal{T}_i$ may increase the distance in a different embedding $\mathcal{T}_j$. At a certain vertex along a packet's routing path, it may be sent back to its origin, resulting in a routing cycle. A cycle avoidance solution requires that each vertex along the routing path decreases the packet's minimum distance (over the k embeddings) to the destination. This way of working is similar to the TCGR mechanism [8]. For this reason a new distance function $\epsilon : \mathbb{T}^k \times \mathbb{T}^k \to \mathbb{R}^+$ is defined as

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

which replaces the original distance function δ [17]. The k embeddings into $\mathbb{T}$ can now be treated as a single k-dimensional embedding into $\mathbb{T}^k$. This allows us to adhere to the principles of geometric routing by using a semimetric space $(\mathbb{T}^k, \epsilon)$.[7] As such, each node u will forward packets to the neighbor with the lowest distance $\epsilon(u, d)$, with d the destination node. Therefore, it is a form of *greedy routing*, which is geometric routing in which a node always forwards to the neighbor leading to the largest decrease in distance. In case multiple neighbors have an equal ϵ-distance, a random choice is made among them.

[7]This double cannot be regarded as a true metric space as the triangle-inequality (property 3 of Definition 3) does not hold.

This mechanism is called *Greedy Forest Routing* (GFR), the greedy variant in the Forest Routing (FR) family. Though GFR focuses solely on attaining a low stretch, the embedding into $\mathbb{T}^k$ leads to increased passive load balancing when compared to a single embedding into $\mathbb{T}$.

B. Load Balanced Forest Routing (LBFR)

To supplement the passive load balancing behavior emerging from GFR, an active load balancing approach was developed called *Load Balanced Forest Routing* (LBFR). This system can be seen as a special case of the final HFR routing scheme. In LBFR, vertices $u \in V$ make use of traffic load information about their incident edges $e \in I(u)$. This information is used to select the neighbor v for which the edge (u, v) has the lowest load[8]. Solely using local link information is advantageous as it is scalable by nature and therefore fitting for a large-scale distributed setting. LBFR relaxes the greedy requirements of GFR by allowing routing alternately via different embeddings $\mathcal{T}_i$ independently. Because naive switching between embeddings may introduce cycles, routing is guided by an auxiliary function κ. This function acts as a routing restriction by requiring that its value decreases at each hop, similarly to how the δ-distance must decrease in standard geometric routing (or ϵ in GFR). All neighbors fulfilling this requirement are added to a set $S(u)$, the set of nodes that can be considered as next hops.

This function κ makes use of an additional function δ^* that outputs a k-tuple storing the minimal distance to the destination d attained by a packet so far along its routing path P_u, before arriving at the current node u, for each of the k embeddings. We denote the i-th element of δ^* as δ_i^* and the union of all possible paths in the network as $\mathcal{P}$. Assuming a packet has been routed along a path $P = \langle p_0, p_1, \ldots, p_n \rangle$ towards a destination vertex d, then κ is of the type $\mathcal{P} \times \mathbb{T}^k \to \mathbb{N}$ and is defined as

$$\kappa(P_{p_n}, d) = \sum_{i=0}^{k-1} \delta_i^* \left(P_{p_n}, d\right), \tag{4}$$

with $\delta^*(P_{p_n}, d)$ a function of the type $\mathcal{P} \times \mathbb{T}^k \to \mathbb{N}^k$ that is defined recursively $\forall i \in \{0, \ldots, k-1\}$ as

$$\delta_i^*(P_{p_0}, d) = \delta_i(p_0, d) \tag{5}$$

$$\forall n > 0 : \delta_i^*(P_{p_n}, d) = \min\{\delta_i^*(P_{p_{n-1}}, d), \delta_i(p_n, d)\}. \tag{6}$$

Herein P_u represents the path P until u has been reached, consisting of the vertices that a packet arriving at u has reached. Furthermore, p_0 is the source vertex of the path P. The minimum distances of each of the k embeddings is thus represented by an element $\delta_i^*(P_u, d)$. The LBFR system will enforce the restriction that κ has to decrease strictly monotonically along the routing path: $\kappa(P_{p_n}, d) < \kappa(P_{p_{n-1}}, d) < \ldots < \kappa(P_{p_0}, d)$. When forwarding, a node u will select those neighboring nodes which have a strictly decreasing κ-value and add them to the previously mentioned set $S(u)$. Next, u will select a vertex $v \in S(u)$ as the next hop for which the current traffic load of the link (u, v) is minimal compared to its other incident links $I(u)$. The LBFR forwarding procedure is described in Algorithm 1 and an example is given next.

[8]The representation of the load may be arbitrarily chosen but should be consistent for all network links.

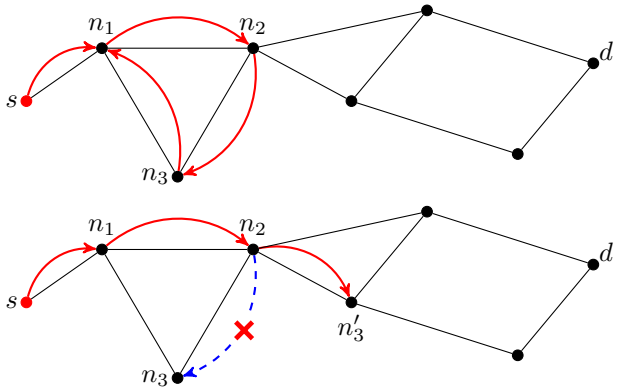


Figure 2. Top: Naive routing with multiple embeddings, a routing cycle is introduced. Bottom: LBFR avoids the introduction of routing cycles.

Example: To illustrate the LBFR system, an example is given for a multi-embedding consisting of three embeddings, thus $k = 3$, which is depicted in Figure 2 (top). This is the equivalent of saying that there is a single embedding into $\mathbb{T}^3$. Thus, every vertex has three sets of coordinates, one for each embedding. 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 embedding $\mathcal{T}_1$ and 7 in embedding $\mathcal{T}_2$. When routing naively the following scenario can occur:

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

In this scenario a cycle is introduced because although the packet is routed greedily in each embedding individually, this does not hold for their aggregation. When routing intelligently restricted by the κ-function, the following happens, see Figure 2 (bottom). At the source node, $\delta^*(\langle s \rangle, d) = \delta(s, d) = (3, 5, 7)$, thus $\kappa(\langle s \rangle, d) = 3 + 5 + 7 = 15$.

- $\delta(n_1, d) = (3, 4, 7)$, s checks n_1 to route on $\mathcal{T}_1$, thus
$$\delta^*(\langle s, n_1 \rangle, d) = (\min\{3, 3\}, \min\{5, 4\}, \min\{7, 7\}) = (3, 4, 7),$$
which means that $\kappa(\langle s, n_1 \rangle, d) = 3+4+7 = 14$. Because $\kappa(\langle s, n_1 \rangle, d) < \kappa(\langle s \rangle, d)$ the next hop n_1 is accepted.
- $\delta(n_2, d) = (3, 6, 6)$, n_1 checks n_2 to route on $\mathcal{T}_2$, thus
$$\delta^*(\langle s, n_1, n_2 \rangle, d) = (\min\{3, 3\}, \min\{4, 6\}, \min\{7, 6\}) = (3, 4, 6),$$
which means that $\kappa(\langle s, n_1, n_2 \rangle, d) = 3 + 4 + 6 = 13$. Because $\kappa(\langle s, n_1, n_2 \rangle, d) < \kappa(\langle s, n_1 \rangle, d)$ the next hop n_2 is accepted.

Algorithm 1: LBFR: forwarding decision making

in : A graph $G = (V, E)$; a vertex $v \in V$ receiving a packet `packet` from vertex $u \in V$; v knows the coordinates of its neighbors $N(v) \subseteq V$ and the current load of its incident edges $I(v) \subseteq E$; all vertices have been embedded into $\mathcal{T}_i$ for $0 \leq i < k$.
out: packet is forwarded to a vertex $n \in N(v)$

```
δ*(u) ← packet.getMinDistances()
δ*(v), S(v) ← ∅ and κ(v) ← 0
foreach i ∈ {0, 1, ..., k − 1} do
    δ*_i(v) ← min{δ*_i(u), δ_i(v, d)}
    κ(v) ← κ(v) + δ*_i(v)
end
foreach i ∈ {0, 1, ..., k − 1} do
    foreach w ∈ N(v) do
        if κ(w) < κ(v) then // calculation of
        κ(w) based on δ*(v) is not shown
            S(v) ← S(v) ∪ {w}
        end
    end
end
minLoad ← +∞ and R(v) ← ∅
foreach w ∈ S(v) do
    if L(v, w) < minLoad then // L(v, w) is the
    load of link (u,w)
        R(v) ← {w}
        minLoad ← L(v, w)
    else if L(v, w) = minLoad then
        R(v) ← R(v) ∪ {w}
    end
end
packet.setMinDistances(δ*(v))
send packet to random vertex in R(v)
```

- $\delta(n_3, d) = (3, 5, 7)$, n_2 checks n_3 to route on $\mathcal{T}_1$, thus

$$\delta^*(\langle s, n_1, n_2, n_3\rangle, d) = (\min\{3,3\}, \min\{4,5\}, \min\{6,7\}) = (3, 4, 6),$$

  which means that $\kappa(\langle s, n_1, n_2, n_3\rangle, d) = 3 + 4 + 6 = 13$. Because $\kappa(\langle s, n_1, n_2, n_3\rangle, d) = \kappa(\langle s, n_1, n_2\rangle, d)$ the next node n_3 is rejected as a potential next hop.
- A node n_3' with $\delta(n_3', d) = (2, 5, 7)$ would be accepted by n_2 because

$$\delta^*(\langle s, n_1, n_2, n_3'\rangle, d) = (\min\{3,2\}, \min\{4,5\}, \min\{6,7\}) = (2, 4, 6),$$

  which means that $\kappa(\langle s, n_1, n_2, n_3'\rangle, d) = 2 + 4 + 6 = 12$, which is lower than $\kappa(\langle s, n_1, n_2\rangle, d)$.

To ensure that LBFR correctly routes traffic between all source-destination combinations, we introduce the following theorems.

Theorem 1: Let $G = (V, E)$ be a graph with k embeddings $\mathcal{T}_i$ for $0 \leq i < k$ into the metric space $(\mathbb{T}, \delta)$. Let d be the destination node. Then, for every path $P \in \mathcal{P}$ (in G) with a last element $v \in V$, for which d has not yet been reached, thus $d \notin P$, the set of neighbors $S(v)$ for which the value of the κ-function strictly decreases is not empty.

Proof: Assume a packet arriving at a vertex v by following a path $P = \langle \ldots, u, v\rangle$. Assume this packet has to be forwarded to a destination vertex d and that $d \notin P$. This means that $S(u) \neq \emptyset$. Therefore $\kappa(v) < \kappa(u)$. Because of the definition of κ as the sum defined by Eq. (4): $\exists i \in \{0, 1, \ldots, k-1\} : \delta_i^*(P_v, d) < \delta_i^*(P_u, d)$. Combining the definition of δ^* in Eq. (6) with the definition of the min-function gives $\delta_i(v, d) < \delta_i^*(P_u, d)$ and $\delta_i^*(P_u, d) \leq \delta_i(u, d)$. Therefore, again because of Eq. (6), $\delta_i^*(P_v, d) = \delta_i(v, d)$. Also, $\delta_i(v, d) < \delta_i(u, d)$, which means that the distance towards d in embedding $\mathcal{T}_i$ has decreased. Because $\mathcal{T}_i$ is a greedy embedding and because of Definition 2: $\exists w \in N(v) : \delta_i(w, d) < \delta_i(v, d)$. Thus, because of Eq. (6), $\delta_i^*(P_v^\frown\langle w\rangle, d) = \delta(w, d)$, and therefore, $\delta_i^*(P_v^\frown\langle w\rangle, d) < \delta_i^*(P_v, d)$. Combining this with Eq. (4) leads to $\kappa(P_v^\frown\langle w\rangle, d) < \kappa(P_v, d)$ based on the fact that δ_i^* never increases along a routing path (due to the min-function in its definition, see above). From this follows: $S(v) \neq \emptyset$. As such, any element from $S(v)$ is a suitable next vertex without violating the LBFR restrictions.

This also holds for a source vertex s: $S(s) \neq \emptyset$. Because of Eq. (5), in s, every value δ_i^* is equal to the distance δ_i. Therefore, any vertex for which the distance towards the destination decreases—and such a vertex exists because each $\mathcal{T}_i$ is a greedy embedding—leads a decrease in κ. ■

Theorem 2: The path followed by a packet routed on a graph $G = (V, E)$ by LBFR is never a cycle.

Proof: Assume d is the destination vertex and a packet has traveled along $P = \langle \ldots, u, v, \ldots, w\rangle$, arriving at $w \in N(v)$. When arriving at v for the first time, $\delta_i^*(P_v, d) \leq \delta_i(v, d)$ because of Eq. (6). Since the values of δ^* can never increase due to the definition of the min-function, upon calculating κ for the second time for vertex v (this time from w): $\nexists i \in \{0, 1, \ldots, k-1\} : \delta_i(v, d) < \delta_i^*(P_w, d)$ because if there would exists such an i, δ^* would already have been updated to this value the first time the packet arrived at v. Thus, the κ-value cannot decrease the second time v is reached. Therefore, no vertex appears twice along the path followed by a packet routed according to LBFR which enforces κ to be strictly monotonically decreasing along a routing path. ■

Theorem 3: A packet routed according to the principles of LBFR on a graph $G = (V, E)$ will arrive at its destination.

Proof: Because κ outputs values in $\mathbb{N}$ and it is strictly monotonically decreasing, while the initial κ-value at the source vertex is finite (assuming $|V|$ is finite), it will become 0 after the traversal of a finite number of vertices, unless it is routed along a cycle or it encounters a void. These two cases are impossible due to Theorems 1 and 2. When for a vertex $v \in V$ and a destination d holds $\kappa(P_v, d) = 0$, this means that $\forall i \in \{0, 1, \ldots, k-1\} : \exists w \in P_v : \delta_i(w, d) = 0$. Thus the destination was reached along the routing path because of property 1 in Definition 3 which defines a metric space. ■
According to these theorems, LBFR is always able to route a

packet to its destination without encountering cycles or voids.

C. Hybrid Forest Routing (HFR)

In terms of stretch and load balancing, GFR and LBFR are two opposites: GFR attains low stretch, but has no load balancing technique, while LBFR achieves load balancing, but pays no attention to stretch. We combine the best of both worlds into one algorithm called *Hybrid Forest Routing* (HFR). HFR makes a trade-off between stretch and load balancing by replacing the GFR distance function ϵ by a cost function that combines link load information with the ϵ-distance to the destination. This cost function $C : V^3 \to \mathbb{R}^+$ is defined as

$$C(u, n, d) = \gamma \cdot \hat{L}(u, n) + (1 - \gamma) \cdot \epsilon(n, d) \tag{7}$$

for $n \in N(u)$, with the ϵ-function defined by Eq. (3). The function $\hat{L}(u, v)$ represents the normalized traffic load of the edge between u and v. This is the traffic load of link (u, v) divided by the average load of all the node's incident links $I(u)$. This normalized load is defined as

$$\hat{L}(u, n) = \frac{d_G(u) \cdot L(u, n)}{\sum_{v \in N(u)} L(u, v)}, \tag{8}$$

with $d_G(u)$ the degree of vertex u. The factor $\gamma \in [0, 1]$ is a weight factor which allows scaling between greedy and load balanced routing. As can be seen, HFR also uses the semimetric space $(\mathbb{T}^k, \epsilon)$, but because the cost function is an extension of the ϵ-function, HFR is not greedy routing. Even more, it is not necessarily distance-decreasing in ϵ.

To guarantee packet delivery, the κ-function from LBFR is used to steer packets towards their destination, relying on the LBFR theorems from the previous section. HFR attains strong load balancing while keeping the stretch down, which is shown in the following section.

GFR and LBFR can now be seen as two special instances of HFR on opposite sides of the spectrum. On the one hand, when $\gamma = 1$, the LBFR mechanism is recreated. On the other hand, when $\gamma = 0$, the HFR reverts to GFR. The HFR forwarding procedure can be created by replacing $L(v, w)$ in Algorithm 1 on lines 16, 18 and 19, with $C(v, w, d)$ as defined in Eq. (7).

VI. Results and Discussion

To test our routing algorithms we have built a routing simulator on top of the open source graph manipulation framework Gephi [18]. This simulator is capable of generating a high number of parallel routing paths in a multi-threaded environment. Due their size and time requirements, all experiments were run on a High Performance Computer. In this simulation framework, each edge is assigned a weight w_e, initially set to 0, which represents the traffic load. Random source-destination node pairs are generated with a given frequency during the simulation, corresponding to a uniform traffic matrix. Between these pairs traffic is simulated by generating routing paths. Whenever a routing path is generated, the weights of its links are increased. For example if traffic f is sent through link e, having a current weight w_e, its new weight becomes $w_e + f$.

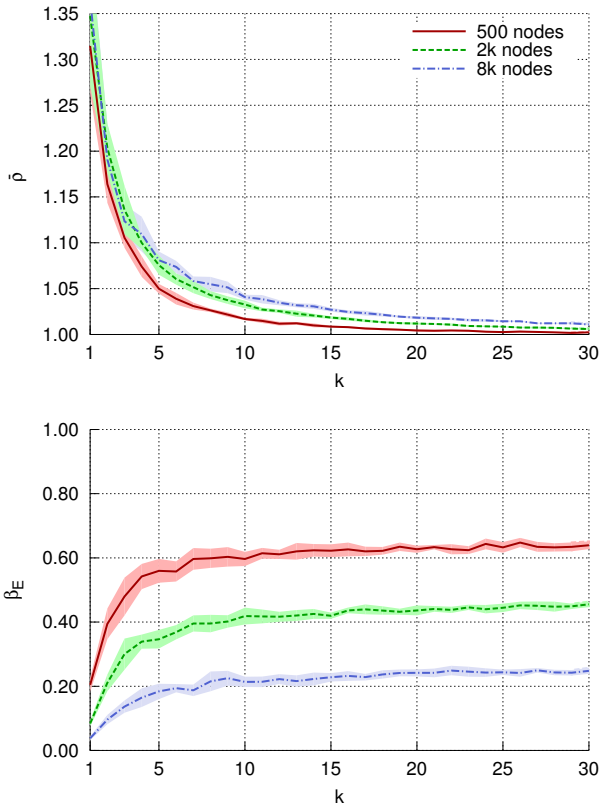


Figure 3. GFR: average stretch $\bar{\rho}$ (top) and link load balancing metric β_E (bottom) in function of the dimension k of the embedding space $\mathbb{T}^k$. The shaded background represents the average, plus and minus the standard deviation.

The algorithms are tested on scale-free networks generated according to the Barabási-Albert model [19] with a degree distribution of $P(k) \propto k^{-2.2}$. The running time of the different experiments is largely dependent on the graph size, ranging from minutes to days.

A. Stretch and load balancing

The different routing algorithms, GFR, LBFR, and HFR, are analyzed regarding their average stretch $\bar{\rho}$ and link load balancing behavior. We define the stretch of a path between source node s and destination node d as the path length (number of hops) divided by the length of the shortest path between s and d. The load balancing behavior is measured by the β_E-metric [20], defined as

$$\beta_E = \frac{\left(\sum_{e \in E} w_e\right)^2}{|E| \sum_{e \in E} w_e^2}, \tag{9}$$

with w_e the weight of edge $e \in E$. Note that $\rho \geq 1$ (1: shortest path routing) and $\frac{1}{|E|} < \beta_E \leq 1$ ($\beta_E \approx 0$: no load balancing; $\beta_E = 1$: traffic is equal on all links).

In Figure 3 the effect of changing the number of embeddings k (or the dimension of the space $\mathbb{T}^k$) on the average stretch $\bar{\rho}$ (top) and the β_E-ratio (bottom) is plotted for GFR. The measurements for each k-value are averaged over 10 runs, and

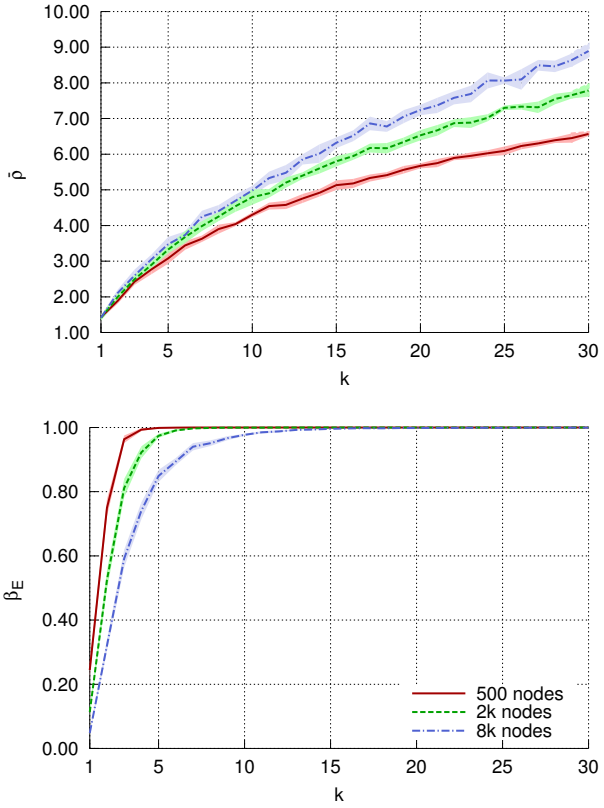


Figure 4. LBFR: average stretch $\bar{\rho}$ (top) and link load balancing metric β_E (bottom) in function of the dimension k of the embedding space $\mathbb{T}^k$. The shaded background represents the average, plus and minus the standard deviation.

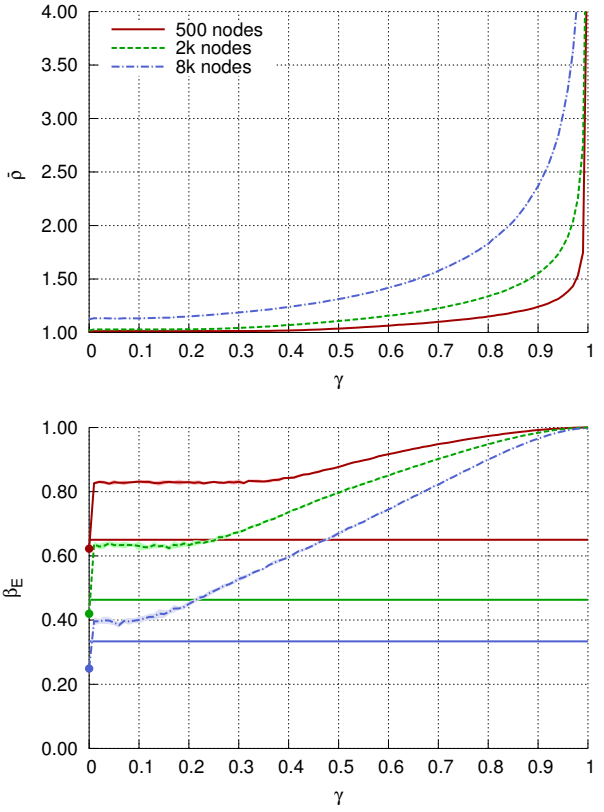


Figure 5. HFR: average stretch $\bar{\rho}$ (top) and link load balancing metric β_E (bottom) in function of a varying γ-value. The solid horizontal lines in the bottom figure represent the load balancing β_E-value of shortest path routing with lookup tables. The shaded background represents the average, plus and minus the standard deviation.

for each run 10^5 routes are generated. Asymptotic behavior of $\bar{\rho} \to 1$ can be observed as $k \to +\infty$. This can be explained by the availability of more embeddings, which allows more routing forwarding freedom. As a result, there is an increased chance that some combination of embeddings will lead to a short path between two nodes. As the number of vertices increases, k also needs to increase to maintain the same stretch because the relative number of embeddings per node or edge decreases. This is consistent with larger graphs having a later convergence towards $\bar{\rho} = 1$.

In Figure 3 (bottom) the link load balancing behavior is plotted. Though there is no active mechanism steering for traffic load balancing in GFR, load balancing improves as k increases. This may be due to the existence of multiple root nodes, splitting the root hotspot—a problem in single tree-based geometric routing schemes [9]—over k roots. Also, the existence of additional short paths between different source and destination nodes reduces the reliance on the root node to act as a transit hub for traffic between different parts of the network. This illustrates the passive load balancing effect of a multi-embedding. β_E decreases as the graph size increases, meaning that the passive load balancing behavior—remember that GFR focuses solely on stretch—weakens. However, the observed trend is similar for all sizes. Furthermore, it can be noticed that the convergence of β_E is delayed for larger graphs, indicating that larger graphs require more embedding.

Figure 4 shows the previous experiment applied to LBFR. In Figure 4 (top) a steep increase in stretch can be observed as k increases. Because more embeddings are available, there are more forwarding candidates that respect the κ-restriction (i.e., κ must be strictly monotonically decreasing along the routed path). Every node focuses entirely on balancing the load of its outgoing links, sacrificing the stretch while doing so.

In Figure 4 (bottom) the β_E-ratio is depicted. As k goes up, β_E converges to 1 very quickly, which indicates near-perfect load balancing behavior. Although the β_E-ratio indicates that the traffic is spread equally over all links, this does not mean that the sum of the traffic over all network links is at its lowest point. Because the stretch increases by a large factor, the total network traffic goes up as well. For this reason, a good practice could be to prioritize stretch in order to avoid overloading the network links. As with GFR, we see that the β_E-convergence happens at higher k-values for larger graphs.

Previously it was shown that GFR enables low stretch while LBFR is able to attain great load balancing behavior. Now HFR, which unites both GFR and LBFR, is evaluated. When the parameter γ in Eq. (7) is shifted towards 0, the

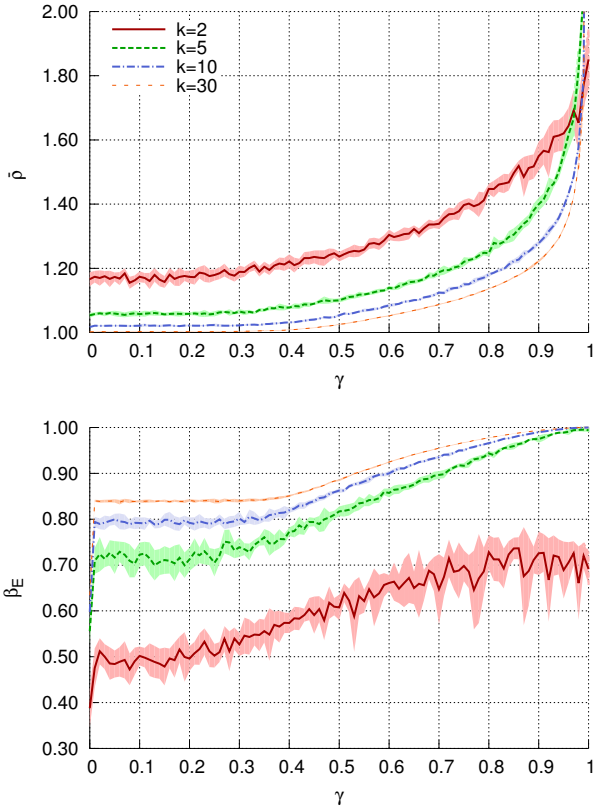


Figure 6. HFR: average stretch $\bar{\rho}$ (top) and link load balancing metric β_E (bottom) in function of a varying γ-value for different k-values on a scale-free graph with 500 nodes. The shaded background represents the average, plus and minus the standard deviation.

GFR system is recreated, while shifting γ to 1 results in LBFR. Therefore, in Figure 5 a sensitivity analysis of the γ-parameter is conducted for $k = 15$. For each value of γ, 10 runs consisting of 10^5 generated routing paths are executed. Figure 5 (top) shows that at low γ-values $\bar{\rho}$ becomes equally low as with GFR. Afterwards, $\bar{\rho}$ starts to incline as $\gamma \to 1$, consistent with HFR approximating LBFR.

The β_E-values in Figure 5 (bottom) indicate that shifting γ between its two extremes gives the expected results regarding load balancing. However, something interesting can be noticed at the right side of $\gamma = 0$. A step occurs such that β_E suddenly rises. Although, when inspecting Figure 5 (top) we see no such step in the $\bar{\rho}$-curve. This can be explained as follows. Upon a node's forwarding decision, many potential candidates will have an equal distance to the destination. So within this set it does not matter which node to forward to in terms of stretch. When taking into account load balancing, a huge improvement can be made by prioritizing those links with a low current load.

Figure 5 (bottom) also shows the load balancing behavior of routing with lookup-tables for the different scale-free graphs, which is depicted by the horizontal solid lines. It can be noticed that HFR offers much better load balancing with only a minor increase in stretch.

Figure 6 shows the influence of γ for different dimensions

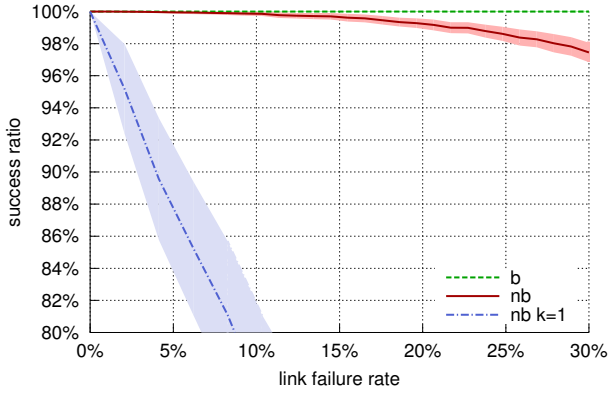


Figure 7. Fault-tolerance: the horizontal axis depicts the fraction of links removed from the total number of links that can be removed ($|E| - |V| + 1$) without disconnecting the graph $G = (V, E)$. HFR ($\gamma = 0.1$, $k = 15$) with (b) and without (nb) backup mechanism is tested, along with a single tree-based geometric routing algorithm (RTP [16]). The shaded background represents the average success ratio, plus and minus the standard deviation.

k on a scale-free graph with 500 nodes. For each γ-value, 10 runs are executed consisting of 10^5 generated routing paths, over which the average and standard deviation of ρ and β_E are taken. Here it is clearly noticeable that the effect of tuning γ increases with k. This can be explained by the higher number of embeddings, allowing for more emphasis on load balancing behavior. However, above a certain k-value, we experience diminishing returns as the trend of the γ-curve stabilizes.

B. Fault-tolerance

In this section the fault-tolerance of HFR is evaluated. We restrict ourselves to link failures, although, node failures lead to very similar results. When more embeddings are used, the chances of a link failure disrupting each of them lower. A link that is a critical parent-child link in one embedding may be a non-essential shortcut link in another one (see Section IV). Hence HFR should be more resilient to failures than routing with a single embedding.

In Figure 7 the routing success rate of the HFR system with $\gamma = 0.1$ and $k = 15$ is shown, alongside HFR combined with the backup routing system Gravity-Pressure (GP) routing [12], and the single tree-based geometric routing algorithm RTP [16], exercised on a scale-free graph with 500 nodes. Links are removed probabilistically such that link failures are spread out evenly over the network to avoid random failure concentration in a certain area. This shows that HFR can be easily equipped with a backup routing mechanism, allowing it to achieve a 100% success rate even in severe failure scenarios. However, even without the GP mechanism, HFR attains a success rate of over 97% when 30% of the removable links are deleted. This is a huge improvement over more basic tree-based geometric routing algorithms based on a single tree where the success ratio quickly declines as the number of failed links increases (a success ratio of less than 50% at a 30% link failure rate).

This inherent fault-tolerance can be explained by the availability of more embeddings which results in more alternate

pathways between source and destination. Therefore, it is less likely that a routing void will occur, a situation in which no new neighbor respecting the κ-restrictions can be found. From this can be concluded that HFR is very fault-tolerant by its very nature, which is essential in a large-scale network where link or node failures are common. However, in case failures are not resolved, the network and its embeddings will deteriorate, which requires coordinate recomputation to avoid large stretch increases. Estimating the exact computation time requires network complexity analysis as well as an analysis of the protocol implementation, which remains future work.

The choice of an appropriate value for k is a multifaceted question. On the one hand, k should be chosen high enough to achieve low stretch, high fault-tolerance, and high load balancing. On the other hand, it should be chosen low enough to avoid extensive routing overhead. The parameter γ should be set to a value corresponding to the desired stretch-load balancing trade-off, which depends on the use case, the graph size, the topology, etc. Although higher γ-values do not affect routing overhead, they do affect the total amount of traffic in the network, as mentioned earlier. Furthermore, k and γ both influence each other and should therefore be tuned in concert. One approach for fitting these parameters would be to inject test traffic into the network for different parameter combinations, while monitoring the routing performance. Parameter estimation, however, remains future work.

VII. Conclusion

In this work a theoretical framework is built which serves as a foundation for the developed family of geometric routing systems, called Forest Routing (FR). Combining a strictly greedy approach, Greedy Forest Routing (GFR), with a load balanced routing scheme, Load Balanced Forest Routing (LBFR), results in Hybrid Forest Routing (HFR).

In HFR, path stretch can be traded against load balancing behavior, two features until now not perceived to be compatible. Due to its local routing decision making procedure it is highly scalable regarding router memory requirements, making it robust towards network growth.

Furthermore, the HFR system has favorable characteristics such as inherent fault-tolerance and guaranteed packet delivery. It can deal even with a highly deteriorated network topology, and is as such able to guarantee success ratios as high as 97% at link failure rates of 30%.

Future work will encompass the incorporation of routing policies into Forest Routing, a detailed Forest Routing complexity analysis, and experiments by emulation.

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". This work was partly funded by Flamingo, an FP7 Network of Excellence project (318488) supported by the European Commission.

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, 2000, pp. 243–254.

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

[3] K. Zeng, K. Ren, W. Lou, and P. J. Moran, "Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply," *Wireless Networks*, vol. 15, no. 1, pp. 39–51, 2009.

[4] J. Zhang, Y.-p. Lin, M. Lin, P. Li, and S.-w. Zhou, "Curve-based greedy routing algorithm for sensor networks," in *Proceedings of the Third international conference on Networking and Mobile Computing*, ser. ICCNMC'05, 2005, pp. 1125–1133.

[5] N. Carlsson and D. L. Eager, "Non-Euclidian geographic routing in wireless networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 1173–1193, 2007.

[6] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica, "Balancing traffic load in wireless networks with curveball routing," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '07, 2007, pp. 170–179.

[7] F. Li, S. Chen, and Y. Wang, "Load balancing routing with bounded stretch," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, pp. 10:1–10:16, 2010.

[8] 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.

[9] J. Newsome and D. Song, "Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03, 2003, pp. 76–88.

[10] R. Kleinberg, "Geographic routing using hyperbolic space," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1902–1909.

[11] F. Papadopoulos, D. Krioukov, M. Boguñá, and A. Vahdat, "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces," in *Proceedings of the 29th Conference on Information Communications*, ser. INFOCOM'10, 2010, pp. 2973–2981.

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

[13] J. Herzen, C. Westphal, and P. Thiran, "Scalable routing easy as PIE: A practical isometric embedding protocol," in *Proceedings of the 19th annual IEEE International Conference on Network Protocols, ICNP*, 2011, pp. 49–58.

[14] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, ser. MobiCom '03, 2003, pp. 96–108.

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

[16] 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, 2007, vol. 4686, pp. 32–45.

[17] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet, "Fault-tolerant greedy forest routing for complex networks," in *RNDM'14 - 6th International Workshop on Reliable Networks Design and Modeling (RNDM 2014)*, Barcelona, Spain, Nov. 2014.

[18] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *ICWSM*, 2009, pp. 361–362.

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

[20] H. Velayos, V. Aleo, and G. Karlsson, "Load balancing in overlapping wireless LAN cells," in *Communications, 2004 IEEE International Conference on*, vol. 7, 2004, pp. 3833–3836.