Greedy Embedding, Routing and Content Addressing for Darknets

Andreas Höfer and Stefanie Roos and Thorsten Strufe
Peer-to-Peer Networks Group
Technische Universität Darmstadt
{a.hoefer, roos, strufe}@cs.tu-darmstadt.de

***Abstract*—To achieve anonymous and censorship-resistant overlay communication, darknets restrict overlay links to trusted parties. Efficient data retrieval in such a restricted topology requires a decentralized addressing scheme. We propose a greedy embedding algorithm, which is used to realize efficient routing and content addressing for darknets. The embedding guarantees success of greedy routing using compact address representations. Evaluation on trust graphs obtained from PGP's web of trust shows that our embedding enables much more efficient routing than existing darknet embeddings. Though, content addressing based on the embedding exhibits unbalanced load.**

I. Introduction

Anonymous, censorship-resistant communication allows to publish opinions and ideas in the Internet without fear of retribution. This is especially important in societies that limit the basic right of freedom of speech.

An approach to realize anonymous, censorship-resistant publication and data sharing is the darknet paradigm. Prominent examples of overlay communication systems following this paradigm are the Darknet mode of Freenet [1], [2] or GNUnet [3].

In a darknet, identity-revealing direct network communication only occurs between nodes with a mutual trust relationship outside the system. Communication between untrusted nodes has to be done indirectly via friend-to-friend links. In other words, the network topology is a subgraph of the social graph resulting from the trust relationships between the participants. Compared to popular anonymous communication systems like Tor [4], the restriction of identity disclosure to trusted participants aims at providing additional protection against network attacks, identification and prosecution.

There are various approaches for content distribution and routing in darknets. The simplest use flooding, e.g. Turtle [5], sometimes enhanced with probabilistic forwarding as in OneSwarm [6]. Flooding scales linearly in the network size and is therefore problematic in large-scale networks. The R5N algorithm [3] of GNUnet combines random walks and a recursive variant of Kademlia routing. It requires a high number of replications. MCON [7] and X-Vine [8] both construct a virtual DHT overlay on top of the friend-to-friend topology connecting virtual DHT neighbors via tunnels over (multiple) friend-to-friend links. This approach requires keeping and maintaining a lot of state information.

An alternative to realize an efficient darknet routing is to use greedy routing on top of an addressing scheme as it is done successfully in standard DHTs. Darknets, however, are topology-restricted, in contrast to DHTs, which allow arbitrary connections between peers. Therefore, the addressing scheme has to adapt to the network topology via an network embedding procedure which assigns each node an address. These addresses can then be used by a local routing algorithm like greedy routing.

In more precise terms, a *network embedding* is a map from the set of nodes of a network into a metric space, assigning each node a point (address) in the metric space. A *greedy embedding* guarantees that greedy routing works in the embedded graph for any source-destination node pair, i.e. for every source-destination node pair holds that the source has a neighbor which is closer to the destination in the metric space [9] (first defined by Papadimitriou et al. [10]).

Greedy embeddings have been proposed in the context of wireless adhoc / sensor network routing, e.g. [10], [11], [12], and Inter-AS routing [13]. In our work they are applied to the problem of darknet routing with focus on the following requirements: The embedding algorithm needs to be *decentralized*, relying only on neighborhood information. *Scalability* is important for realizing systems with a large number of participants. This translates into requiring first of all *short routing paths*, but also *low address description complexity*, *low message complexity*, and a *low runtime*. Furthermore, the embedding should enable *content addressing*. This means that data items and nodes share an address space, each data item is mapped to a node which is responsible for this data item and the data item can be found by routing to its address. The *load* resulting from content addressing should be *balanced*, i.e. each node should be responsible for roughly the same number of data items.

These requirements are a subset of all requirements we deem necessary to be adhered to by a comprehensive darknet embedding approach. Main issues, which have to be handled in future work, are network dynamics, robustness against failures and attack resilience.

In this paper, we present Prefix Embedding, a simple, scalable, decentralized, greedy embedding algorithm for darknets, which improves upon existing greedy embeddings in terms of the coding length of the address representations.

We show by simulations that the algorithm achieves routing lengths close to the shortest paths, considerably shorter than any state-of-the-art darknet embedding algorithm. We point out how the algorithm can be modified for content addressing, at slight performance costs. The load distribution of the content addressing scheme is evaluated.

In Section II, the related work on embeddings is presented, with a focus on their applicability for darknets. Afterwards, Prefix Embedding is described in Section III and analyzed theoretically. Evaluations with regard to routing performance and load balancing are done in Section IV. Section V concludes and points out directions for future work.

II. Related Work

Heuristic embeddings, which only approximate greedy embeddings, are introduced in the first part of this section. Greedy embedding techniques stemming from different application areas are discussed in the second part.

A. Heuristic Embeddings

Several iterative embedding algorithms for darknets solely based on local interactions between the nodes have been devised, like Swapping [14], LMC [15], and spectral graph drawing [16].

Swapping: The swapping algorithm has been applied to friend-to-friend embedding in the darknet mode of Freenet [2]. Swapping is based on the idea that the darknet topology is assumed to resemble a Kleinberg small-world topology [17] in one dimension. Starting from a random address assignment the embedding is iteratively adapted in form of a Metropolis-Hastings algorithm [14]. An iteration of the algorithm consists of pairing two nodes by a random walk and swapping their addresses with a probability depending on the current addresses in their direct neighborhood.

LMC: Swapping is extremely vulnerable to misbehaving overlay participants spreading incorrect address information [18], [15]. In the Local Markov Chain (LMC) algorithm, nodes choose an address uniformly at random rather than exchanging address information via a random walk. The new address is then accepted with a probability determined by the addresses of direct neighbors according to the Metropolis-Hastings algorithm.

Decentralized Spectral Graph Drawing: Dell'Amico proposes a decentralized version of the spectral graph drawing algorithm by Koren [19] for embedding small-world networks. The embedding problem in one dimension is formulated as minimizing a stress function representing euclidean distances between adjacent nodes under constraints ensuring that the embedding does not collapse into a single point. This problem is solved by an iterative eigenvector computation. Dell'Amico's embedding algorithm executes a variant of this eigenvector computation in a decentralized manner and for multiple dimensions in parallel. A single iteration of the eigenvector computation can be interpreted geometrically as moving each node to the midpoint between its old coordinates and the centroid of its neighbors.

Further Heuristic Embeddings: Several heuristic embedding algorithms for greedy-like routing have been proposed in the context of wireless adhoc / sensor networks. The NoGeo [20] and GSpring algorithms [21] model edges between nodes as ideal springs and use a spring-force model to compute an embedding into the euclidean plane. The resulting embeddings are not guaranteed to be greedy. Furthermore, both approaches use variants of flooding to determine perimeter nodes on which the embedding is based.

In the context of general routing, heuristic embeddings have been employed as well. Ban et al. [22] apply variants of multidimensional scaling to model the node distances in different classes of networks (e.g. citation networks, Internet-AS networks). The resulting embedding is not greedy and the computation relies on landmark nodes. FPC [23] uses a spring-force approach and applies it to a two-hop neighborhood around each node to embed scale-free networks (attracting forces act between one-hop neighbors and repelling forces act between two-hop neighbors). As the resulting embedding is not greedy, additional source routing techniques are used to guarantee routing functionality.

The main disadvantage of heuristic embedding algorithms is that the resulting embedding is in general not greedy. For guaranteeing success, the routing algorithm has to be modified. However, routing paths might be considerably longer than in case of greedy embeddings. Another drawback of these approaches is that the number of necessary iterations of the embedding algorithm is high and can often only be determined empirically.

B. Greedy Embeddings

Most greedy embedding procedures (e.g. [24], [11], [12], [25], [13], [26]) construct a greedy embedding for a spanning subgraph (typically a spanning tree) of the network. Adding edges does not destroy the greediness property, which is why a greedy embedding of a spanning subgraph is a greedy embedding for the full graph as well [11].

If one constructs a greedy embedding along a spanning tree, message delivery is guaranteed along the spanning tree edges. The additional edges can be used as short-cuts. While these constructions guarantee to find a path, they do not guarantee to find the shortest path. But an upper bound for the routing stretch can be given as twice the spanning tree depth, as this is the length of the longest possible greedy path in the spanning tree [13].

In the following the embedding algorithms are classified according to the metric spaces used for the embeddings.

Hyperbolic Embeddings: Kleinberg [11] proposed the first greedy embedding algorithm for arbitrary graphs into the hyperbolic plane. Given the maximum degree d of a rooted spanning tree the algorithm computes the embedding by mapping the spanning tree to an infinite tree of degree

d which is the dual tree of a plane tiling by d-gons. Although a node address from the hyperbolic plane consists of just two coordinates the description complexity is linear in the network size n as the description complexity of each coordinate is linear [25], [13].

For the hyperbolic greedy embedding algorithm by Cvetkovski et al. [12], knowledge of the maximum degree in the spanning tree is not necessary. Still the coordinates have linear description complexity [13].

The problem of linear description complexity is tackled by Maymounkov [9] and Eppstein et al. [27]. Maymounkov computes greedy embeddings into three-dimensional hyperbolic spaces using polylogarithmic description complexity. Eppstein et al. embed into the hyperbolic plane using logarithmic description complexity. Whereas the algorithms due to Kleinberg and Cvetkovski et al. can be computed in a decentralized manner, the algorithms by Maymounkov and Eppstein et al. rely on heavy-path decompositions of the embedded graphs, which make them hard to decentralize.

Euclidean Embeddings: A greedy embedding into high-dimensional euclidean spaces is proposed by Westphal and Pei [25]. Their algorithm makes use of the Johnson-Lindenstrauss Lemma and random projections. The greediness of the embedding is guaranteed with high probability if the embedding space has sufficiently high dimension.

A similar approach is sketched by Maymounkov in [9].

Maximum Norm Embeddings (Tree isometries): Linial et al. [24] propose an isometric embedding of distances in a rooted spanning tree into high dimensional spaces equipped with the maximum norm[1]. As any tree isometry is a greedy embedding of the tree, one can obtain a greedy embedding of an arbitrary graph by constructing an isometry of a spanning tree. To ensure that only $O(\log n)$ embedding dimensions are needed, the algorithm requires the computation of central nodes of the spanning (sub)trees encountered during the embedding, which makes it hard to decentralize.

The PIE embedding algorithm due to Herzen et al. [13] is based on an isometric embedding of the distances on a rooted spanning tree of the graph similar to the approach by Linial et al. The algorithm can be decentralized as it embeds the nodes traversing the spanning tree from the root to the leaves. Assuming a logarithmic depth of the spanning tree, the description complexity of the embedding is polylogarithmic.

Custom Metric Embeddings: Flury et al. [28] construct greedy embeddings with $O(\log n)$ routing stretch and polylogarithmic description complexity using a custom min-max metric. Their algorithm relies on the computation of a tree cover of the graph based on global topology information and is therefore not applicable to darknets.

Zhang et al. [26] compute greedy embeddings by enumerating nodes in a depth-first search (DFS) traversal of a spanning tree of the graph and assigning each node its DFS enumeration indices as coordinates. They define a custom semi-metric on the resulting coordinates for distance computations. The description complexity of the embedding depends linearly on the maximum node degree. Due to the DFS traversal, the embedding is not complete until all edges have been traversed sequentially.

In summary, most greedy embeddings can either not be applied in the darknet scenario or require a linearly growing description complexity. An exception is PIE, a decentralized embedding offering a polylogarithmic description complexity. For that reason, Prefix Embedding presented in Section III mainly adapts the idea of PIE to routing and content addressing in the darknet scenario.

III. Prefix Embedding

Prefix embedding follows the standard approach of embedding a spanning tree of the graph. More precisely, it is an isometric embedding of the hop count metric on the embedded spanning tree. It can be considered an adoption of the PIE algorithm [13] to unweighted graphs, reducing the size of the address representations. Furthermore, it simplifies realization of content addressing.

After introducing the algorithm in Section III-A, it is proven in III-B that the resulting embedding is indeed greedy. In Section III-C, the algorithm is analyzed with regard to its complexity properties. In Section III-D variants of the algorithm for fixed-size address spaces are given. We close this part with a description of our content addressing scheme.

A. Prefix Embedding Algorithm

The embedding algorithm works as follows:

- Choose a root node and assign an empty coordinate vector to the root.
- Compute a spanning tree starting from the root node (using breadth-first search) and traverse it on the fly.
- During the tree traversal, every node enumerates its children and assigns each child its coordinates. The coordinates of a child are the parent's coordinates concatenated with one additional coordinate corresponding to the child's index assigned during the enumeration.

The resulting embedding for a tree with depth 3 is displayed in Fig. 1.

B. Proof of Greediness

We use greedy routing with a custom distance metric based on prefix matching. The distance between two nodes s and t is defined as

$$dist(s,t) = |s| + |t| - 2 * |matchingprefix(s,t)|$$

where $|s|$ is defined as the length (number of coordinates) of the coordinate vector of s and $matchingprefix(s,t)$ is the common prefix of the coordinate vectors of s and t.

[1] An embedding is isometric when it preserves the node distances given by the network metric.

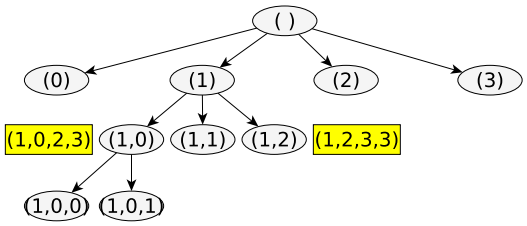


Figure 1. An example of embedding a tree of maximum degree 4, resulting in 2-bit coordinates. 8 bits are used for content addressing, meaning that every file address consists of four coordinates. The file with address $(1, 2, 3, 3)$ is stored at the leaf $(1, 2)$, whereas the file with address $(1, 0, 2, 3)$ is stored at the internal node $(1, 0)$, which has only two successors in the tree.

Similarly to [13], we show that using the above distance metric, Prefix Embedding produces not only a greedy, but an isometric embedding of the hop count metric on the tree. Consider the hop count distance $hop(s,t)$ from a source node s to a target node t. Then $hop(s,t)$ is the distance from the source to the least common ancestor node (lca) in the tree plus the distance from the lca to the target. So we can write: $hop(s,t) = hop(s,lca) + hop(lca,t)$. $hop(s,lca)$ equals $|s| - |lca|$ as lca is an ancestor of s in the tree and one coordinate is added per level of the tree. The address of the lca as least common ancestor of s and t is $matchingprefix(s,t)$. Therefore, $hop(s,lca) = |s| - |matchingprefix(s,t)|$ and analogously $hop(lca,t) = |t| - |matchingprefix(s,t)|$. In consequence, $hop(s,t) = hop(s,lca) + hop(lca,t) = |s| + |t| - 2 * |matchingprefix(s,t)| = dist(s,t)$.

C. Complexity Analysis

1) Address Description complexity: The number of coordinates of a node address is bounded by the tree depth and the magnitude of each coordinate is bounded by the maximum degree of the tree, so that at maximum $treedepth * \log maxdegree$ bits are needed for one node address. The maximum degree is bounded by the number of nodes, so the description complexity is $O(treedepth * \log n)$ bits. Assuming that darknets being social networks have the small-world property [29] the diameter is approximately in $O(\log n)$. Hence, the depth of the tree is in $O(\log n)$, and the description complexity is $O(\log^2 n)$ bits.

2) Message Complexity: Assuming that a spanning tree of the graph is precomputed and corresponding local information stored at each node, the total number of messages needed for an embedding of a graph is $n-1$, one message per edge of the spanning tree. If the spanning tree is computed on the fly during the embedding, $O(e)$ messages are needed for the embedding (e being the number of edges in the network), as a node has to probe its neighbors to learn if they are already embedded.

3) Runtime: The runtime considered as the number of time-units from the start of the embedding until all nodes are embedded is dominated by network delays, therefore only message delays are counted.

In prefix embedding, subtrees of any spanning tree node can be traversed in parallel. Therefore, the embedding is completed after the longest/slowest root-to-leave path of the tree has been traversed. The time (sum of delays) for traversing this root-to-leave path is in $O(treedepth)$ assuming uniform constant edge costs/message delays. Assuming a small-world network this is again in $O(\log n)$.

D. Prefix Embedding in Fixed-Sized Address Spaces

To achieve concise address representations and facilitate content addressing using a fixed-size address space the basic embedding algorithm needs to be adapted.

1) Bounded-Degree Embedding: In the following, let the *degree* of a node v in a rooted tree t denote the number of children of v in t. The first embedding variant for fixed-size address spaces requires a bound on the maximum degree in the spanning tree, which might mean that not all nodes can be included in the tree. Assume we have an address space with addresses of fixed length s bits (e.g. $s = 160$). We bound the maximum number of children of a node in the spanning tree to a power of two, 2^b, and therewith fix the number of bits b per coordinate (e.g. when using 5 bits per coordinate, a tree of depth 32 with 32 children per inner node fits into a 160 bit address space). In case a node has less than 2^b children, still b bits are used per coordinate.

2) Virtual Tree Embedding: An alternative, which allows more flexible node degrees in the spanning tree, is to represent one node as a set of several virtual nodes. A high-degree node in the spanning tree can be modeled with a tree of virtual nodes of low degree (cf. Fig. 2). In the following this low degree is set to 2, so that the virtual trees are binary trees. Each node is solely assigned the address of the root node of its virtual tree.

This scheme has the disadvantage of losing the isometric property of the embedding as several virtual nodes are represented by the same physical node. The distances given by the embedding are upper bounds on the number of nodes on the path between any source-destination pair. For example, in Fig. 2, there is a path of length 3 between $(0, 1, 1)$ and $(1, 0)$, but the distance is 5.

The binary trees of virtual nodes representing a physical node are constructed as balanced as possible, i.e. if a node has c children, the first $2*c - 2^{\lceil \log_2 c \rceil}$ receive an address with $\lceil \log_2 c \rceil$ additional coordinates, the others receive $\lceil \log_2 c \rceil - 1$ additional coordinates. The construction ensures that no child address is a prefix of any other child address.

As the resulting embedding is not strictly greedy, greedy routing is extended: In case there is no neighbor with a smaller distance to the target the message is forwarded to the parent in the spanning tree. To see that this guarantees routing success consider a source-destination pair (s,t). As in Section III-B the path from s to t in the spanning tree can be split into two subpaths from s to the least common

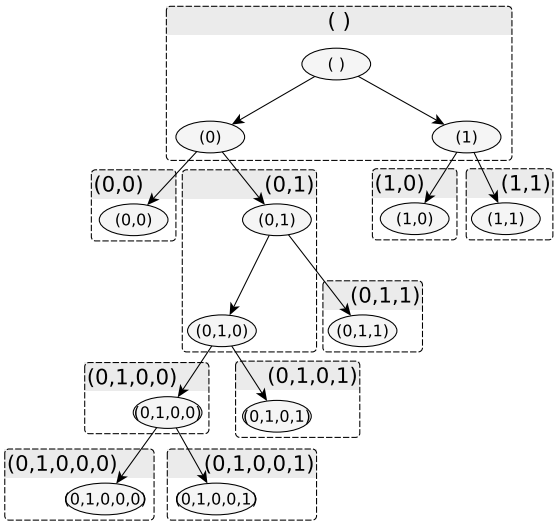


Figure 2. The tree from Fig. 1 using virtual trees. The squares represent the actual physical nodes, the ellipses the virtual nodes. Physical nodes with more than two children, i.e. the root and its second child, are represented by a virtual subtree of minimal size needed to address all children.

ancestor node, lca and from the lca to t. The path from s up the tree to the lca can be further split into the path from s to the child c of the lca and the one-hop path from c to the lca. Greedy routing up the tree on the sub-path from s to c succeeds as for each child-parent edge holds that they have the same common prefix with t but the parent has less coordinates. For the one hop from c to the lca, greedy routing might fail, as c might have a longer common prefix with t than its parent lca. This failure is handled by forwarding to the parent. On the sub-path down the tree from the lca to t greedy routing is again guaranteed to succeed as each for parent-child edge the parent address is a shorter prefix of t's address than the child address.

E. Content Addressing

To address a content item in a publication system based on prefix embedding, we assign bit strings to the data items and interpret them as tree coordinate vectors. The node with the longest matching prefix is then responsible for the data item. Formally, let F be the set of possible file identifiers, V the set of nodes. Then the function $r : F \to V$ maps a file address f to $v \in V$, so that $dist(v, r(f))$ is minimized.

It remains to map the bit strings to tree coordinate vectors. Using the variants of Prefix Embedding from Section III-D, the mapping is straightforward: Split the bit string in blocks of fixed size and map the blocks from left to right to single coordinates of a node address. The block size can be computed from the bound on the node degree (in case of bounded-degree embedding) or the predefined virtual node degree (in case of virtual tree embedding). Fig. 1 shows the mapping of two files with 8 bit addresses into a tree of depth 3 using 2 bit coordinates from a bounded-degree embedding.

One problem of such a content addressing scheme is load balancing as some nodes might be responsible for much larger slices of the address space than other nodes. In the following, the embedding variants are analyzed with regard to the number of bits each node is responsible for. Assume that file addresses consist of s bits and are distributed uniformly (e.g. due to generation by a hash function). Note that the level of a node is defined as the distance to the root.

If Prefix Embedding is used with a fixed number b of bits per level of the spanning tree, a node u on level L with c children in the tree is responsible for $2^{s-(L+1)\cdot b} \cdot (2^b - c)$ possible file addresses, i.e., all addresses for which u's address is a prefix, but not the address of any child of u. This means u is responsible for $s-(L+1)\cdot b+\log_2(2^b-c)$ bits.

In case of virtual tree embedding, a node with a degree of at least 2 in the spanning tree balances its children in such a way that it is not responsible for any part of the address space. Only leaves and nodes of degree 1 are responsible for parts of the address space. In case of a leaf node, the node is responsible for all addresses which have the node address as prefix, that are 2^{s-L} addresses. In case of a node of tree degree 1, the node is responsible only for half of these, 2^{s-L-1} addresses or $s - L - 1$ bits.

IV. Evaluation

In this section, Prefix Embedding is evaluated with regard to routing performance and load balancing, using existing darknet embeddings for comparison.

A. Setup

For the evaluation, darknets are modeled as simple undirected graphs. Embedding and routing are done on static graphs without taking network dynamics into account. As network topologies for the evaluation, four snapshots of PGP's web of trust[2] are used, but due to space constraints, we focus on one snapshot from January 1st, 2011. The results for the other graphs are very similar.

In PGP's web of trust, an edge from PGP user A to B indicates that A signed B's public key. This can be interpreted as a trust relationship (namely that A trusts in B), so that PGP web of trust topologies were used previously for the evaluation of darknet routing algorithms in [14], [16]. As darknets are based on mutual trust, we remove trust relationships without reciprocation from the raw web of trust graph. Afterwards, the giant component of the graph is extracted to ensure a connected topology. The resulting graph has 33317 nodes, an average degree of 8.3, a median degree of 2, maximal degree of 992, an average shortest path length of 6.15, and a diameter of 25.

All results are averaged over 30 runs, the Graph-Theoretic Network Analyzer (GTNA)[3] is used as framework for the evaluation.

[2]http://www.lysator.liu.se/~jc/wotsap/wots2/

[3]http://www.p2p.tu-darmstadt.de/research/gtna/

Setup Heuristic Embeddings: For comparison, three heuristic darknet embeddings are used: Sandberg's swapping algorithm, LMC and Dell'Amico's spectral graph drawing, as described in Section II-A. All algorithms are round-based: In each iteration of swapping and LMC, a random node is chosen to execute the address adaptation algorithm, whereas in each iteration of spectral graph drawing the addresses of all nodes are adapted. The parameters for the algorithms are taken from the respective papers: In case of swapping and LMC, $6000n$ iterations are used, where n is the number of nodes. The length of the random walks in the swapping algorithm is set to 6. For the spectral graph drawing algorithm 200 iterations and address spaces of 10 and 100 dimensions are used. Because of the non-greediness of the embeddings, routing is done with Freenet's distance-directed depth first search [1] using a time-to-live counter ttl of 200.

Setup Prefix Embedding: Prefix Embedding is evaluated without any degree constraints as well as with maximum degrees of $16, 32, 64, 128$ in the spanning tree, corresponding to $b = 4, 5, 6, 7$ bits per coordinate. The isometric embedding as well as the virtual tree embedding are evaluated.

For the load balancing evaluation, file addresses are assumed to have 160 bits, corresponding to common hash lengths. Note that content addressing in case of unrestricted degrees is only possible if virtual trees are used.

The basic embedding algorithm works with any spanning tree. But the bounded-degree variant of the algorithm (Section III-D) requires a bounded-degree spanning tree. There is no guarantee that such a tree exists in the network and even if it exists, its computation is NP-hard. For this reason we evaluate three simple tree construction schemes, which do not guarantee that the computed tree spans the graph:

1) *RND*: execute a breadth-first search (BFS) from a root chosen uniformly at random, add maximally 2^b children randomly, but deterministically, from the set of neighbors that are not part of the tree.
2) *HD*: like *RND*, but choose the root uniformly at random from the 1% of nodes with the highest degree.
3) *FAT*: choose the root uniformly at random from the 1% of nodes with the highest degree, in case a node has more than 2^b neighbors that are not part of the tree, choose the 2^b with the highest degree.

For isometric Prefix Embedding, greedy routing is used whereas for virtual tree embedding extended greedy routing as described in Section III-D2 is used.

Setup - Metrics: The metrics for routing performance and load balancing are computed as follows. The routing performance is sampled by routing from each node in the address space to 5 distinct nodes chosen uniformly at random from all nodes with an address. The routing is considered failed if the source is not part of the address space or the target is not found in ttl hops. The average routing length is taken as the average over all queries, as is the cumulative hop distribution, i.e. for each x, the fraction of queries needing at most x hops was sampled. Variances are small, and are not given for a clearer presentation.

To evaluate the load balancing, we measure the distribution of the sizes of the address space slices assigned to single nodes by the embedding, i.e. for each x the fraction of nodes responsible for x bits of the address space is sampled.

B. Routing Performance

Expectations: Prefix Embedding is guaranteed to have a routing success rate of 1 when the degree is unrestricted. In case of a degree restriction, the success rate corresponds to the fraction of successfully embedded nodes, which is expected to increase with the maximal degree. The success rate of the heuristic embeddings is expected to be considerably below 1, because non-greedy embeddings result in very long routes in some cases, where the address information is misleading.

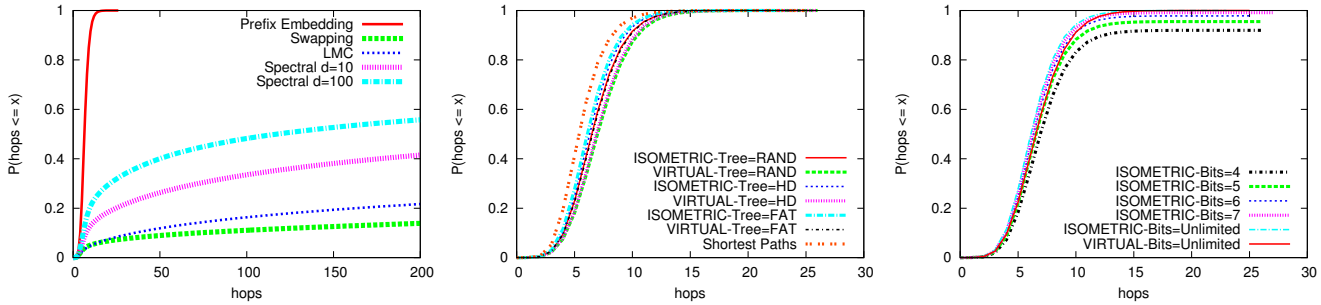
The routing paths are expected to be much shorter in case of a greedy embedding, as there are no loops and backtracking parts. An increase of the maximal degree as well as the usage of FAT trees might decrease the length of the path slightly, since the tree depth is bound to decrease.

Using virtual trees is expected to lead to a longer path length, because the tree embedding is no longer isometric, which might mislead the routing algorithm.

Results: The main purpose of a good embedding is that a routing algorithm should find short paths between all nodes. Comparing Prefix Embedding with other darknet embeddings shows that it is significantly better than heuristic embeddings. The maximal path length (Fig. 3a) for Prefix Embedding using RND trees and no restrictions on the degree is 26, whereas for the heuristic embeddings only about 56% (Spectral $d = 100$), 42% (Spectral $d = 10$), 22% (LMC), and 14% (Swapping) are successful within 200 hops.

The influence of the different spanning tree construction algorithms (with unlimited degree) can be seen from Fig. 3b. FAT trees lead to a reduced average path length of about 6.84 steps in contrast to 7.03 steps for HD and 7.23 steps for RND trees, whereas the average shortest path length is 6.15. The routing performance decreases when virtual trees are introduced. Still, the average routing path length is between 7.28 for FAT trees and 7.71 for RND trees, meaning it is about 1.5 hops above the optimal solution.

Fig. 3c displays the effect of degree restriction on the number of embedded nodes and the path length distribution for FAT trees. For 4 bit coordinates, corresponding to a maximal degree of 16, only 92% of the nodes are embedded, for 5, 6 and 7 bit coordinates, it is about 96%, 98% and 99%, respectively. Restricting the degree in the tree leads to longer routes, since the tree height is increased. Comparing the routing length of the isometric embedding with virtual trees (for FAT trees), using virtual trees has a slightly lower



(a) Heuristic Embeddings (b) Spanning Tree (c) Degree Restrictions

Figure 3. Routing Performance: a) compares Prefix Embedding with unlimited degree with heuristic embeddings, b) shows the influence of various spanning tree constructions on both with and without virtual trees and c) shows the influence of restricting the degree

average path length of 7.28 than the 4 bit case (7.43 hops), but takes on average longer than the isometric embedding with 5 bit (7.16), 6 bit (7.02), and 7 bit coordinates (6.94). For random spanning trees the success rates are slightly higher, whereas the paths are slightly longer, as can be seen in Fig. 3b.

C. Load Balancing

Expectations: The load distribution is not uniform, since the number of bits a node is responsible for depends on its tree level and the number of children it has: A node having the maximal degree in the tree, is not responsible for any file address whereas a root node having less than half of the maximal degree is responsible for more than half of the address space.

Virtual trees are bound to lead to a more uniform load distribution, since the nodes on the first levels are not responsible for any addresses, assuming they have at least two children. Only leaf nodes and nodes with degree one are responsible for parts of the address space in virtual tree embeddings. If the spanning trees are relatively balanced, most of these nodes should be deeper in the tree, meaning they are responsible for less bits. However, the fraction F_0 of nodes not responsible for any file address is expected to be higher in virtual tree embedding compared to isometric embedding.

Results: Though having a worse routing performance, virtual trees offer a better load balancing. Given the embedding algorithm (ALG), the number of bits per coordinate (b), and the spanning tree algorithm (ST), Table I displays the maximum, mean, median, and non-zero minimum number of bits a node is responsible for, as well as the fraction F_0. Note that content addressing without degree restrictions ($b = -$ in Table I) is only possible when using virtual trees. For comparison, the virtual tree embedding is also evaluated using degree restrictions. As example, Table I shows the case $b = 6$, respectively a bound of 64 for the degree in the tree.

In Table I one can see that when a random node is selected as a root, this node is on average responsible for nearly the whole address space. The maximum load is only slightly

Table I
DISTRIBUTION OF SIZES OF ADDRESS SPACE SLICES: NUMBER OF BITS

ALG	ST	b	max	mean	median	min	F_0
ISOM	RND	6	159.82	119.84	123.39	57.20	0.02
ISOM	HD	6	154.00	130.84	135.99	69.60	0.02
ISOM	FAT	6	153.98	131.23	135.99	69.40	0.02
VIRT	RND	6	158.00	112.00	136.20	117.40	0.18
VIRT	RND	–	157.83	113.46	134.77	114.83	0.16
VIRT	HD	6	154.00	114.74	139.70	121.83	0.18
VIRT	HD	–	153.17	116.60	138.70	119.50	0.16
VIRT	FAT	6	153.60	114.66	139.60	119.07	0.18
VIRT	FAT	–	153.33	115.12	136.17	113.90	0.16

decreased in case of virtual tree embeddings, for which the node with the maximum address space slice is on average responsible for a quarter of the address space (158 bits). The results clearly indicate that a low degree node high up in the tree leads to a highly unbalanced load distribution. This is improved considerably by using higher degree nodes as root, but FAT trees do not further reduce the maximum load compared to HD trees. The later is presumably a result from the very low number of high degree nodes, some children of the root have not the required degree and are hence responsible for large address space slices. Using virtual trees without a fixed number of bits per level, the maximum load is slightly reduced but still on average one node out of the 33317 nodes is responsible for more than 153 bits ($1/128$ of the address space) in case of FAT and HD trees and 157 bits ($1/8$ of the address space) in case of RND trees. As expected, for virtual tree embeddings the fraction F_0 is much higher than for isometric embeddings. Consequently, the average number of bits is lower, while the median and minimum are higher.

V. CONCLUSION AND FUTURE WORK

We introduced a darknet routing approach and content addressing scheme based on *Prefix Embedding*, a greedy network embedding technique. As greedy embedding Prefix Embedding guarantees success of greedy routing. It requires only modest address representation sizes ($O(\log^2 n)$ bits in case of networks with $O(\log n)$ diameter), a low number of messages and low runtime for the embedding.

In the evaluation using PGP's web of trust, greedy routing on networks embedded with heuristic embeddings exhibits low success rates and very long routing paths. On the contrary, Prefix Embedding guarantees routing success and routing paths are very short compared to the heuristic paths and close to the shortest paths in the network.

Prefix embedding and the spanning tree embeddings in general impose a hierarchy over the nodes and introduce the need to select a root node. This is a drawback compared to heuristic embeddings, which do not assign special importance to any node.

Content addressing on top of the embedding exhibits unbalanced load. Appropriately constructed spanning trees and virtual tree embeddings mitigate the imbalances somewhat but they remain an issue. Strategies to handle these imbalances are to be analyzed in future work. Furthermore, our simple load model needs to be made more realistic.

Having seen that our approach offers very goods routing performance in a static environment, it remains to analyze it in dynamic, possibly malicious environments. The maintenance of the embedding under network dynamics caused by churn needs to be considered. This is especially urgent for greedy embeddings as they rely on data structures spanning the whole network like spanning trees. We plan to investigate measures for handling network dynamics and evaluate variants of prefix embedding using event-based simulation. Attack resilience is another important problem for most of the embedding approaches, greedy or not. Embeddings often allow an attacker to influence the addresses of nodes in his neighborhood. Hence, methods for detecting attacks and restricting an attacker's influence are needed. One of the main goals of a darknet is protecting the participants against prosecution by hiding their identity. This raises the question how much information about participating nodes an attacker can gain from observing and maybe manipulating an embedding and its messages.

References

[1] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with freenet," *IEEE Internet Computing*, vol. 6, no. 1, pp. 40–49, 2002.

[2] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel, "Private communication through a network of trusted connections: The dark freenet," http://freenetproject.org/papers.html, 2010.

[3] N. S. Evans and C. Grothoff, "R5N: Randomized recursive routing for restricted-route networks," in *International Conference on Network and System Security (NSS 2011)*, 2011.

[4] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *USENIX Security Symposium*, 2004.

[5] B. C. Popescu, B. Crispo, and A. S. Tanenbaum, "Safe and private data sharing with turtle: Friends team-up and beat the system," in *Security Protocols, 12th International Workshop*. Springer, 2006.

[6] T. Isdal, M. Piatek, A. Krishnamurthy, and T. E. Anderson, "Privacy-preserving p2p data sharing with oneswarm," in *SIGCOMM '10*, 2010.

[7] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim, "Membership-concealing overlay networks," in *CCS '09*, 2009.

[8] P. Mittal, M. Caesar, and N. Borisov, "X-vine: Secure and pseudonymous routing using social networks," in *Network & Distributed System Security Symposium (NDSS '12)*, 2012.

[9] P. Maymounkov, "Greedy embeddings, trees, and euclidean vs. lobachevsky geometry," https://www.pdos.lcs.mit.edu/~petar/papers/maymounkov-greedy-prelim.pdf, 2006.

[10] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 3–14, 2005.

[11] R. Kleinberg, "Geographic routing using hyperbolic space," in *INFOCOM 2007*, 2007.

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

[13] J. Herzen, C. Westphal, and P. Thiran, "Scalable routing easy as pie: A practical isometric embedding protocol," in *ICNP 2011*, 2011.

[14] O. Sandberg, "Distributed routing in small-world networks." in *Workshop on Algorithm Engineering and Experiments (ALENEX '06)*, 2006.

[15] B. Schiller, S. Roos, A. Höfer, and T. Strufe, "Attack resistant network embeddings for darknets," in *Symposium on Reliable Distributed Systems Workshops (SRDSW)*, 2011.

[16] M. Dell'Amico, "Mapping small worlds," in *P2P 2007*, 2007.

[17] J. M. Kleinberg, "The small-world phenomenon: an algorithmic perspective," in *STOC '00*, 2000.

[18] N. S. Evans, C. GauthierDickey, and C. Grothoff, "Routing in the dark: Pitch black," in *Annual Computer Security Applications Conference (ACSAC 2007)*, 2007.

[19] Y. Koren, "On spectral graph drawing," in *Computing and Combinatorics (COCOON 2003)*, 2003.

[20] A. Rao, C. H. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *MOBICOM 2003*, 2003.

[21] B. Leong, B. Liskov, and R. Morris, "Greedy virtual coordinates for geographic routing," in *ICNP 2007*, 2007.

[22] X. Ban, J. Gao, and A. van de Rijt, "Navigation in real-world complex networks through embedding in latent spaces," in *Workshop on Algorithm Engineering and Experiments (ALENEX'10)*, 2010.

[23] Y. Wang, G. Xie, and M.-A. Kaafar, "FPC: A self-organized greedy routing in scale-free networks," in *Symposium on Computers and Communications (ISCC '12)*. IEEE, 2012.

[24] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215–245, 1995.

[25] C. Westphal and G. Pei, "Scalable routing via greedy embedding," in *INFOCOM 2009*, 2009.

[26] H. Zhang and S. Govindaiah, "Greedy routing via embedding graphs onto semi-metric spaces," in *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management (FAW-AAIM 2011)*. Springer, 2011.

[27] D. Eppstein and M. T. Goodrich, "Succinct greedy geometric routing using hyperbolic geometry," *IEEE Trans. Computers*, vol. 60, no. 11, pp. 1571–1580, 2011.

[28] R. Flury, S. V. Pemmaraju, and R. Wattenhofer, "Greedy routing with bounded stretch," in *INFOCOM 2009*, 2009.

[29] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.