Geometric Routing on Flat Names for ICN

Yanbin Sun, Yu Zhang, Hongli Zhang, Binxing Fang
School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, China
Email:{sunyanbin, zhangyu, zhanghongli, fangbinxing}@nis.hit.edu.cn

Xiaojiang Du
Department of Computer and Information Sciences,
Temple University, Philadelphia, PA 19122, USA
Email: xjdu@temple.edu

***Abstract*—This paper presents Griffin, a scheme of geometric routing on flat names to conduct massive content distribution and retrieval. A tree-based metric space $\mathcal{T}$ is proposed according to the concept of hierarchical division of symbol space. In Griffin, the network topology is embedded into the $\mathcal{T}$-space, and content names are mapped to the $\mathcal{T}$-space. Content publication and retrieval are supported by geometric routing in the $\mathcal{T}$-space. Different from previous embedding schemes, Griffin constructs the $\mathcal{T}$-space according to the network topology before embedding. In contrast to prior name resolution schemes, Griffin operates directly on the network topology without establishing an overlay. The correctness of Griffin is proved by the greediness of geometric routing. The experiments by simulation demonstrate that Griffin is efficient and scalable.**

I. Introduction

Applications on the Internet have shifted from host-centric to content-centric in recent years. *How to conduct massive content distribution and retrieval in the Internet* has become a big challenge. Information Centric Networking (ICN), which focuses on contents (what) instead of hosts (where), is considered a promising clean-slate Internet architecture design. In ICN, a content is accessed by its unique name decoupled from its location, its owner or anything other than the content itself. In some ICN designs, such as DONA [1], PSIRP/PURSUIT [2], and NetInf[3], the content namespace is *flat*, i.e., there is no topological/structural pattern.

To publish and retrieve a content by its flat name, a typical design adopts *name-based routing via name resolution*. The mapping information between content names and content locations are registered and resolved in a name resolution system. The name resolution system is usually deployed as an overlay, such as hierarchical DHT (Distributed Hash Table) [4], [5], [6]. To retrieve a content, a consumer first resolves the content name to a location by name resolution, and then fetches the content by IP-like address-based routing.

However, the above design has two limitations. First, an overlay-based name resolution system may lead to high path stretches in resolving names, as there is inconsistency between the underlying network topology and the overlay topology. Second, address-based routing protocols, such as distance-vector and link-state routing protocols, require each router should store the routing information for every destination, from which the routing scalability may suffer. Although geographic DHT [7], [8], [9] can reduce the stretches and the routing information by embedding the network topology into a geographic/virtual coordinate space, routing may fail due to packets trapped in local minima, because the embedding of topology does not satisfy the greediness of geometric routing [10]. It is unclear that, for any network topology and any flat namespace, whether there is a universal metric space so that the topology can be greedily embedded and the namespace can be well resolved without overlay. We hypothesize the answer is negative.

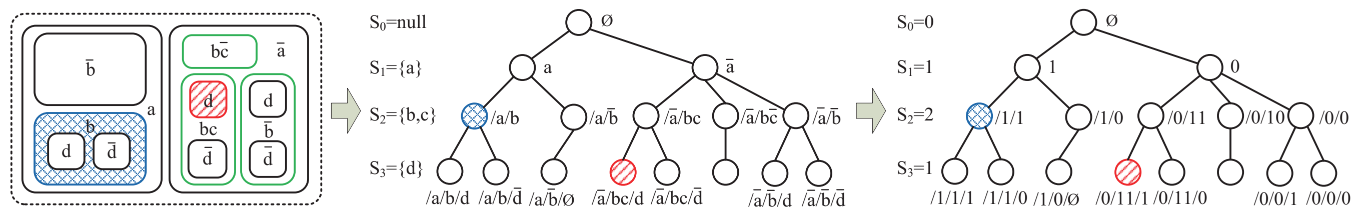
This paper presents *Griffin*, a scheme of Geometric RoutIng on Flat names for ICN without overlay. Instead of searching for such (probably non-existent) universal metric space, our main idea is to construct a tree-based metric space, $\mathcal{T}$-space, tailored for each network topology. The concept of $\mathcal{T}$-space is inspired by the hierarchical division of symbol space, while each instance of the $\mathcal{T}$-space is derived from the spanning tree of topology. The $\mathcal{T}$-space can not only naturally support the greedy embedding of topology for underlying geometric routing, but also be used as the key space of DHT for name resolution/registration. Since for any node, the neighbors of the node in the DHT topology are exactly the neighbors of the node in the network topology, the DHT topology is consistent with the underlying topology, which avoids using an overlay. We will show the correctness of Griffin by proving that there is no local minima during geometric routing. Griffin has high scalability, low latency and balanced distribution of resolution entries, which is demonstrated by simulation.

The main limitation of Griffin is that it is not suitable for dynamic topologies, as the metric space may need to be reconstructed when the topology changes. We consider this limitation the tradeoff for geometric routing without overlay. We believe that Griffin is just an early step towards the practical application of geometric routing for ICN.

The organization of the rest of this paper is as follows. Section II discusses related work. Section III presents the detailed design of Griffin. Section IV evaluates the performance of Griffin. Section V concludes the paper.

II. Related work

Most routing schemes using name resolution in ICN adopted overlay networks. MDHT [4], SVMDHT [5] and HDHT[6] proposed hierarchical DHTs for name resolution by hierarchically dividing the topology into hierarchical resolution domains, and used IP-like routing for the underlay routing. Though the hierarchy provides topological embedding from the overlay to the underlay, the embedding is only for the resolution domain topology. Within each resolution domain,



(a) Hierarchical division based on symbols (b) Division tree based on symbols (c) General division tree

Fig. 1. An example of hierarchical division

the name resolution service is supported by the overlay DHT. αRoute [11] proposed a flat alphanumeric DHT based on a partitioning scheme, which is similar to the hierarchical division of symbol space of $\mathcal{T}$-space in Griffin. However, the DHT of αRoute is an overlay design, which needs an overlay-to-underlay mapping scheme to reduce the routing path lengths. Different from these schemes, Griffin adopts a DHT without overlay for name resolution.

GHT [7] is used for data-centric storage in sensor networks. It implements a DHT on a geographic coordinate space and uses GPSR (Greedy Perimeter Stateless Routing) for routing. However, GHT cannot guarantee greedy embedding and needs additional strategies to recover from the local minima. Subsequent researches, such as GEM [8] and T-DHT [9], adopted virtual coordinate spaces. They avoid the local minima by routing on the spanning tree of topology, which may bring long routing paths. Inspired by GHT, Griffin implements a DHT directly on the $\mathcal{T}$-space without overlay.

Some geometric routing schemes guarantee the greedy embedding of the topology, but do not directly support name-based routing. Hyperbolic routing [10] proposed a geometric routing scheme in a hyperbolic space. In that paper, an idea of DHT on the hyperbolic space is mentioned but without a detailed design. PIE [12] proposed a geometric routing scheme in l_p-norm space. Both PIE and Griffin adopt prefix-free coding for graph embedding.

Recently, some name-based routing schemes by geometric routing were proposed. Similar to Griffin, they adopted geometric routing and name mapping to implement underlay DHTs. MobiCCN [13] maps content names to coordinates by SHA-1 and uses the hyperbolic routing for name resolution. The coordinate length is $O(n)$ bits. Prefix-S [14] proposed the prefix-s embedding for geometric routing and the topology-aware hashing for name mapping. The coordinate length in Prefix-S is shorter than that in MobiCCN. But the graph embedding in Prefix-S is not greedy, and each node stores the whole network topology for the hashing computation. Griffin provides both scalable coordinates ($O(\log^2 n)$ bits) and greedy embedding. Moreover, the name mapping in Griffin only needs a small amount of topology information.

III. Design of Griffin

Given a network topology $\mathcal{G}$, Griffin constructs a tree-based metric space $\mathcal{T}$ according to $\mathcal{G}$. By greedily embedding $\mathcal{G}$ into $\mathcal{T}$ and mapping the namespace to $\mathcal{T}$, Griffin achieves geometric routing on flat names for content publication and retrieval.

We present the $\mathcal{T}$-space by first introduce a division $D_{\mathcal{S}}$ which hierarchically divides the symbol space and then describe the one-to-one mapping between $D_{\mathcal{S}}$ and $\mathcal{T}$. Later, we elaborate the scheme of greedy embedding $\mathcal{G}$ into $\mathcal{T}$ and the mapping scheme from the namespace to $\mathcal{T}$. After proving the correctness of geometric routing, we present how to publish and retrieve contents in Griffin.

A. Hierarchical division $D_{\mathcal{S}}$

To propose the $\mathcal{T}$-space, we first introduce a hierarchical division based on the symbol space. Note that there is no relationship between the symbol space and the content namespace.

The symbol space $\mathcal{S}$ includes all possible symbol strings. $\mathcal{S}$ can be divided into subspaces by specifying the absence or presence of each symbol according to a division parameter S_i (here is a symbol subset), which is called a division by the parameter. As shown in Figure 1(a), for the division by $\{a\}$, the subspace /*a* is composed of the symbol strings including at least one '*a*', and the subspace /$\bar{a}$ is composed of symbol strings without '*a*'. We call /*a* and /$\bar{a}$ the subspaces' IDs.

A hierarchical division $D_{\mathcal{S}}$ hierarchically divides $\mathcal{S}$ into non-overlapping subspaces via the ordered divisions by the division parameters (non-overlapping symbol subsets). We call a subspace after i times of division an ith-level subspace. The ID of an ith-level subspace is obtained by concatenating the ID of parent subspace with the ID in the ith-level division. The process of division can be described with a division tree (Figure1(b)) on which each node represent a subspace and the root represent the whole $\mathcal{S}$.

By representing the absence or presence of symbols with 0 or 1, a general division tree is obtained. As shown in Figure 1(c), the subspace ID of a node turns to be a hierarchical bit-string and the division parameter is the cardinality of corresponding symbol subset. The hierarchical string of a node cannot be the prefix of hierarchical strings of its siblings so that corresponding subspaces are non-overlapping. The general division tree can represent multiple hierarchical divisions of $\mathcal{S}$. If symbol strings can be transformed to different bit-strings, $\mathcal{S}$ can always be divided according to the general division tree. Note that the structure of the tree is determined by the hierarchical division according to the specific topology. Figure 1 shows an example of $D_{\mathcal{S}}$ with the general division tree, which conducts 3-level divisions on parameters 1, 2 and 1.

There exists a one-to-one mapping between the subspaces and the nodes. All leaf nodes are on the same level. The subspace of the node /1/0/ø is the same with the subspace of the node /1/0, but they are two subspaces in different levels. The subspace /1/0/ø denotes the 3rd-level division on S_3 divides the subspace /1/0 into one subspace.

B. Tree-based metric space $\mathcal{T}$

A tree-based metric space $\mathcal{T}$ corresponds to a division $D_{\mathcal{S}}$. $\mathcal{T}$ is a set of points with a distance definition (metric) between points. A point is identified by a coordinate. To build $\mathcal{T}$, let each subspace generated by $D_{\mathcal{S}}$ correspond to a point in $\mathcal{T}$. Thus each point corresponds to a node on the general division tree, so $\mathcal{T}$ can be visualized by the division tree. Let $T_{\mathcal{T}}$ be the tree of $\mathcal{T}$ corresponding to the division tree. Let the coordinate of a point be the ID of the corresponding subspace. The length of the coordinate is the level number of the ID.

The distance between any pair of points/coordinates in $\mathcal{T}$ is defined as the distance between the corresponding nodes by walking on $T_{\mathcal{T}}$. For any two points s, t in $\mathcal{T}$, to calculate the distance only based on the coordinates without the knowledge of the division tree, we define the distance function as

$$d(s,t) = l_s + l_t - 2l_p, \tag{1}$$

where l_s and l_t are the lengths of the coordinates of s and t, respectively, and l_p denotes the length of the longest common prefix between the coordinates of s and t. Two equal-length prefixes of coordinates are considered common, if two subspaces the prefixes represent are the same or one contains the other. For example, the lengths of /0/101/1 and /0/10/0 are both 3, and the distance between them is 2, as the corresponding subspace of /1/10 contains the corresponding subspace of /1/101.

Theorem 1: For a coordinate set H corresponding to the general division tree of $D_{\mathcal{S}}$ and the distance function $d(\cdot,\cdot)$ (Eq. 1) on H, define $\mathcal{T}$ as H with the metric $d(\cdot,\cdot)$. $\mathcal{T}$ is a metric space.

Proof: According to the definition of the metric space, for any $x, y, z \in H$, they should follow three properties: positive definiteness, symmetry and triangle inequality. The first two properties are obviously satisfied. We prove that x, y, z also satisfy triangle inequality.

Let l_x, l_y, and l_z denote the length of x, y, and z. Let l_1, l_2, l_3 denote the length of the longest common prefix of pairs (x,y), (y,z), (x,z). Then

$$d(x,y) + d(y,z) - d(x,z) = 2(l_y - (l_1 + l_2 - l_3))$$

Since $l_3 = min(l_1, l_2)$, then,

$$l_1 + l_2 - l_3 = max(l_1, l_2)$$

Since $max(l_1, l_2) \leq L_y$, then,

$$l_y - (l_1 + l_2 - l_3) \geq 0$$
$$d(x,y) + d(y,z) \geq d(x,z)$$

■

C. Greedy embedding of topology

Embedding the network topology $\mathcal{G}$ into the metric space $\mathcal{T}$ is to assign each node in $\mathcal{G}$ a coordinate in $\mathcal{T}$. The greedy embedding of $\mathcal{G}$ can be achieved by greedily embedding a spanning tree $T_{\mathcal{G}}$ of $\mathcal{G}$ into $\mathcal{T}$[10]. $T_{\mathcal{G}}$ can be obtained by using the STP protocol [15].

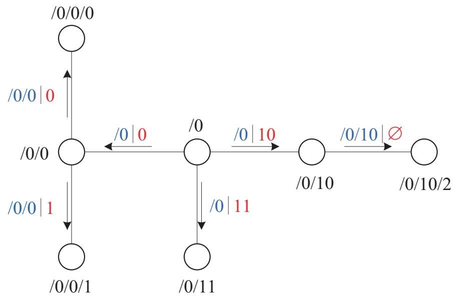


Fig. 2. An example of assigning hierarchical bit-strings

Instead of using a pre-defined $\mathcal{T}$, we construct $\mathcal{T}$ from $T_{\mathcal{G}}$ to improve the consistency between the topology and the metric space, and also to simplify the process of embedding. We use $T_{\mathcal{G}}$ as a subtree of $T_{\mathcal{T}}$. To construct the part of $T_{\mathcal{T}}$ other than $T_{\mathcal{G}}$, we need to extract division parameters from $T_{\mathcal{G}}$. Once $T_{\mathcal{T}}$ is obtained, each node on $T_{\mathcal{G}}$ is assigned a coordinate of $\mathcal{T}$. The greedy embedding of $\mathcal{G}$ is naturally achieved.

We first assign each node a hierarchical bit-string, i.e. a coordinate, and then obtain the division parameter of each level on $T_{\mathcal{G}}$.

Assigning the hierarchical bit-string is started from the root node in a top down manner. Given a node v in the ith level of $T_{\mathcal{G}}$ with a hierarchical bit-string b_v=/b_v^0/b_v^1/.../b_v^{i-1}. Let $C(v)$ be the set of children of v. For each child $c \in C(v)$, v computes a bit-string b_c^i by a prefix-free coding, e.g., Huffman coding, to avoid subspace overlapping. Thus, the child c gets its hierarchical bit-string by appending b_c^i to b_v. Figure 2 shows an example. The node with a coordinate /0 has three children, and it computes three bit-strings 11, 10, and 0 for the children. These bit-strings combined with /0 are sent to the children, respectively. After the children get their hierarchical bit-strings, they repeat the process. For the node /0/10, there is only one child, so the bit-string is null. For the convenience of expression, we use 2 to denote the null bit-string.

To get the division parameter, for any non-leaf node u in the ith level, let $|B_u^i|$ denote the length of the longest bit-string assigned to its children. The node u announces $|B_u^i|$ to the root node. Then the root node obtains the division parameter in the i+1th level: S_{i+1}=max$\{|B_u^i| \; |u$ is in the ith level$\}$, i.e., the maximum ID length for all i+1th-level divisions. After all division parameters are determined, the root node broadcasts these parameters to each node.

Since all leaf nodes of $T_{\mathcal{T}}$ are in the same level, $T_{\mathcal{T}}$ is obtained by completing $T_{\mathcal{G}}$ with the following method: If the depth of $T_{\mathcal{G}}$ is k, then each leaf node of $T_{\mathcal{G}}$ on the ith level ($i \neq k$) obtains $2^{S_{i+1}}$ new children. These children is also assigned hierarchical bit-strings by the parent node. The process is repeated until all leaf nodes are in the same level. Thus, each node of $T_{\mathcal{T}}$ obtains a coordinate. Since $T_{\mathcal{G}}$ is a sub-tree of $T_{\mathcal{T}}$, the embedding is trivial and greedy.

D. Namespace mapping

To map a content name to a coordinate in $\mathcal{T}$ is to assign the name to a node on $T_{\mathcal{T}}$. We determine the branch of the node on $T_{\mathcal{T}}$ first and then the depth of the node on $T_{\mathcal{T}}$.

A branch on the tree is determined by a leaf node. A content name n is first hashed to a bit-string $h(n)$ with a consistent hash function. If the length of $h(n)$ is long enough to denotes a subspace corresponding to a leaf node on $T_{\mathcal{T}}$, the branch which n is mapped to is determined. For example, if $h(n)$ is 0100..., then the leaf node /0/10/0 is obtained by the division in Figure 1(c). If $h(n)$ is 0001..., then the name n obtains /0/00/1 which is not a node of $T_{\mathcal{T}}$. Since /0/00/1 is equal to /0/0/1, /0/00/1 belongs to the branch determined by the leaf node /0/0/1.

To determine the depth of the node, we first assign each level of $T_{\mathcal{T}}$ an interval. For better load balancing, we adopt some parameters of T_G so that the distribution of coordinate lengths is consistent with the node number of each level of T_G. For the ith level, the interval I_i is $\prod_{j=0}^{i} 2^{\lceil \log |C_j| \rceil}$, where $|C_j|$ is the maximum number of siblings in the jth level of $T_{\mathcal{G}}$. For each content name n, we calculate a value $p = hash(n)\%\{\sum_i I_i\}$. Beginning from the root, if p falls in the interval of the kth level of $T_{\mathcal{T}}$, then the depth is k.

E. Geometric routing

Based on the greedy embedding of topology, geometric routing does greedy forwarding with local information. Each node only stores the coordinates of neighbors. When a packet reaches a node other than the destination, the node greedily selects the neighbor nearest to the destination as the next hop if the neighbor is nearer to the destination than the node. The process is repeated until the packet reaches the destination. The greedy embedding of topology guarantees that such next hop can always be found if the destination is a node on the topology [16].

In Griffin, the destination is not only a node on the topology but also a point (content coordinate) in $\mathcal{T}$. We show that geometric routing can always find a unique node which is the nearest to the destination by proving the following theorem. Thus, the geometric routing on flat names can avoid local minima and each node can find a next hop to the unique node.

Theorem 2: Given a graph $\mathcal{G}(V, E)$ and its corresponding metric space $\mathcal{T}(H, d)$. $\exists f : V \rightarrow H$, where f is the greedy embedding. For any $p \in H$, there is a unique node $s \in V$ such that $d(f(s), p) < d(f(t), p)$ for any $t \in N(s)$, where $N(s)$ is the neighbor list of s.

Proof: If p is a node coordinate, then $f(s) = p$. Obviously, the unique node s exists. If p is a virtual coordinate in $\mathcal{T}$, we assume that there are two nodes s, a that are nearer to p than their neighbors, respectively.

According to the greedy embedding of the spanning tree $T_{\mathcal{G}}$ of $\mathcal{G}$, $T_{\mathcal{G}}$ is a sub-tree of $T_{\mathcal{T}}$. For all nodes on $T_{\mathcal{G}}$, the node s is the nearest ancestor node of p on $T_{\mathcal{T}}$. Otherwise, the parent or one of the children of s is nearer to p than s, which is inconsistent with the assumption. Similarly, the node a is also the nearest ancestor node of p on $T_{\mathcal{T}}$. Since the nearest ancestor node of p is unique, we obtain $s = a$, which contradicts to the assumption. Thus, the node s is unique. ■

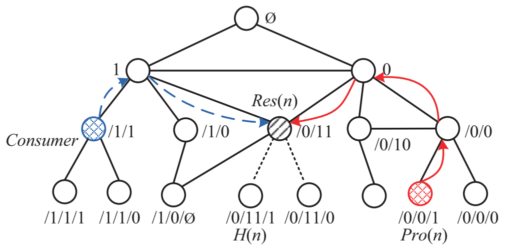


Fig. 3. Name registration and resolution

F. Content publication and retrieval

On the embedded topology, contents are published and retrieved via name registration and resolution, respectively. For a content with a name n, let $Pro(n)$ be its provider coordinate and let $H(n)$ be the coordinate of n in $\mathcal{T}$ by namespace mapping. The name resolution node $Res(n)$ responsible for n is the closest node to $H(n)$. The content provider publishes the content to $Res(n)$ by registering n with $Pro(n)$. A content consumer retrieves the content by first resolving n to $Pro(n)$ and then retrieving the content from $Pro(n)$. To register/resolve n, a registration/resolution packet is sent towards $H(n)$ by the provider/consumer and will reach $Res(n)$ via geometric routing.

Fig.3 shows an example of name registration and resolution. All circles are the points of $\mathcal{T}$ and the whole topology with solid lines as edges are greedily embedded into $\mathcal{T}$. To register (resolve) n, the provider (consumer) first maps n to a coordinate /0/11/1 which does not belong to the topology, then a registration (resolution) packet which contains the destination /0/11/1, the name n and the coordinate $Pro(n)$ ($H(consumer)$) is greedily forwarded. Each node that receives the packet, selects the best neighbor which is nearer to /0/11/1 than the node, and forwards the packet to the neighbor. When the packet reaches the node $Res(n)$, there is no neighbor nearer to /0/11/1 than $Res(n)$, then $Res(n)$ is the resolution node of n.

When $Res(n)$ receives a registration packet, a $\langle n, Pro(n) \rangle$ pair is stored in its resolution table as a resolution entry. When $Res(n)$ receives a resolution packet, $Res(n)$ resolves n to $Pro(n)$ /0/0/1 according to the corresponding resolution entry and returns $Pro(n)$ to the consumer, then the consumer sends a content request to $Pro(n)$. $Res(n)$ can also directly forward a content request to $Pro(n)$ with $H(consumer)$, then the provider returns the content to the consumer.

To reduce resolution latency and guarantee the robustness of name resolution, there may exist multiple replicas for a resolution entry in the network. A content name is mapped to multiple coordinates by using different hash functions, and then it is registered to corresponding nodes. The consumer can select the nearest coordinate for name resolution. When a resolution request failed, the consumer can re-send the request to another coordinate.

G. Load balancing

We propose two strategies to improve the load balance among resolution nodes:

(1) Coding with unequal probabilities (CUP). To make the number of registrations on a subtree of T_G proportional to the size of subtree, we adopt Huffman coding with unequal probabilities in greedy embedding. According to the name mapping scheme, a node with a shorter code covers a bigger subspace and will be responsible for more registrations. Therefore, when a node assigns bit-string to its children, the bit-string length of a child is inversely proportional to the size of the subtree rooted at the child.

(2) Registration forwarding (RF). If the number of registrations on a node s exceeds a threshold, upon receipt of a new registration packet, s can transfer the load to a neighbor t by forwarding the packet. The neighbor t is chosen deterministically according to the hash value of the content name. If t is also overloaded, the forwarded packet will be pushed back to s, and s should either store the registration or return a negative acknowledgement to the provider. The name resolution adopts the same process.

IV. Evaluation

We evaluated Griffin by simulation on various topologies. The real-world topology is obtained from CAIDA's AS-level topology dataset [17] in January 2012 with the average degree 4.2 and the scale factor λ 2.46. The synthetic topologies include random graphs with the average degree 4 and power-law graphs with $2 < \lambda < 3$ generated by the ER [18] model and the BA [19] model, respectively.

1. *Correctness*

The correctness of Griffin is proved by Theorem 2. Here, we validate the correctness of Griffin by checking whether the contents can be published and retrieved correctly. In a simulation on the AS graph, 5000 content names with different lengths are generated, and for each name, 500 nodes are randomly selected as providers to register the name. The experiment shows that all registration packets with the same content name from different providers are routed to the same resolution node. As shown in Fig.4, for each node, the number of resolution entries on the node is an integral multiple of 500, because each name is registered at the same node 500 times.

2. *Path stretch*

The performance of geometric routing compared with the shortest path routing can be measured by path stretch. Based on theorem 4.3 in [12], it can be proved that the stretch upper bound in power law graphs with $2 < \lambda < 3$ is $O(\log n)$. Our results indicates that the actual stretch is much better than the theoretical expectation. On the AS graph, where over 70% of routing paths are the shortest path, the maximum stretch is no more than 3, and the average stretch is 1.084.

To evaluate the stretch as the network grows, we compare Griffin with a tree-based routing scheme on the ER and BA graphs of different sizes. The tree-based routing scheme routes packets along the shortest path on the spanning tree, which scales with the growth of the network. As shown in Fig.5, the average stretches of Griffin on the ER and BA graphs are below 1.7 and 1.2 respectively, which are lower than those of the tree-based routing scheme.

3. *Latency*

To measure the latency of name resolution, we registered 500 contents on the AS graph. For each content, we adopted m different hash functions to register a content name, respectively. Thus, there are m resolution replicas in the network for each content. In name resolution, the consumer chooses the nearest one from the m coordinates. As shown in Fig.6, the latency decreases with the increase of m, as there is a bigger chance of resolving at a nearby node. In the worst case, the number of hops of name resolution is no more than 7.

4. *Coordinate length*

The scalability of greedy embedding as the network grows can be evaluated by coordinate lengths. Here, we measure the coordinate length by the number of bits to describe a coordinate. It can be proved that the coordinate length by coding with equal probability is $O(\log^2 n)$ bits according to [12]. Fig.7 shows that the average and the maximum numbers of coordinate values on the BA graphs are much shorter and increase much slower than the theoretical upper bound.

5. *Node states and load balancing*

States maintained by each node include a neighbor list and a resolution table. As the size of the neighbor list is much smaller than the size of the resolution table, we focus on the distribution of resolution table size. In the simulation, 10^6 content names are registered by randomly chosen nodes on the AS graph. The threshold in the registration forwarding is set to 240 which is 2.5 times of average registration number.

Fig.8 shows the CDF of number of registrations with and without the load balancing strategies. Obviously, our strategies can improve the load balancing. Moreover, the standard deviation of registration numbers are reduced from 226.49 without the load balancing strategies to 52.12 with the load balancing strategies. According to the larger version of Fig.8, over 97% of registration numbers are under the pre-defined threshold by using CUP, and overloaded nodes are eliminated by using RF. Furthermore, Fig.9 shows that for most of nodes, the registration numbers are between 10 and 150.

V. Conclusion and Future work

This paper proposes Griffin, a scheme of geometric routing on flat names for ICN without overlay. By constructing a tree-based metric space tailored for each network topology, Griffin builds a DHT for flat names directly on the network topology without overlay. We proved the correctness of Griffin, i.e., geometric routing on flat names without local minima, in theory. Our analyses present that the upper bound of path stretch is $O(\log n)$, and each node only stores $O(c/n)$ resolution entries, where n is the network size and c is the number of contents. Our experiments show that Griffin provides low latency, scalable state size and balanced distribution of resolution entries. The average path stretch is independent of network size and is extremely low especially for Internet-like graphs.

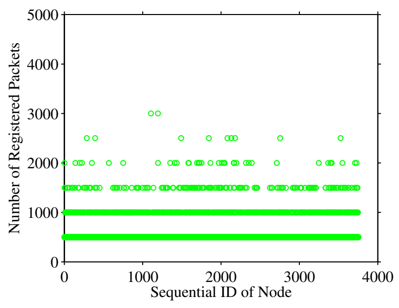


Fig. 4. The number of resolution entries

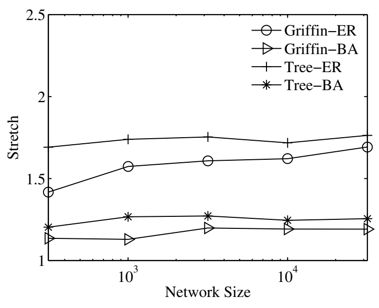


Fig. 5. Average path stretch

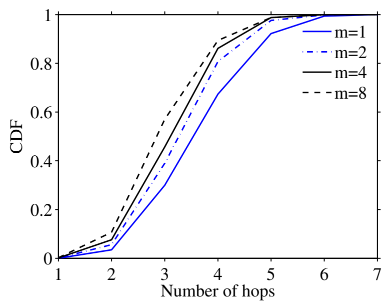


Fig. 6. Resolution latency

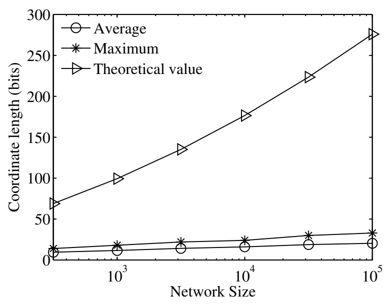


Fig. 7. Coordinate length

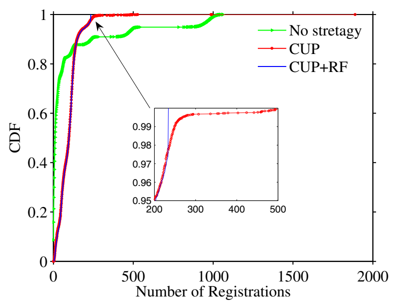


Fig. 8. CDF of registration numbers

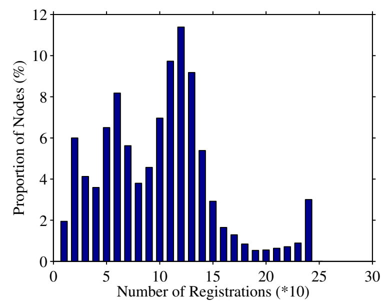


Fig. 9. Distribution of registration numbers

Currently, Griffin is not aware of other key components of ICN, such as anycast and in-network caching, and may not support dynamic topologies. Therefore, there are two interesting questions for our future work: one is how to support anycast and in-network caching in Griffin, the other is how to apply Griffin to dynamic topologies.

Acknowledgment

This research is supported in part by the National Key Basic Research Program of China (973 Program) under Grant No. 2011CB302605, 2013CB329602, and the National Natural Science Foundation of China under Grant No. 61202457, 61402149.

References

[1] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 181–192.

[2] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From psirp to pursuit," in *Broadband Communications, Networks, and Systems*. Springer, 2012, pp. 1–13.

[3] C. Dannewitz, "Netinf: An information-centric design for the future internet," in *Proceedings of the 3rd GI/ITG KuVS Workshop on The Future Internet*, 2009.

[4] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT: a hierarchical name resolution service for information-centric networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 7–12.

[5] H. Liu, X. De Foy, and D. Zhang, "A multi-level dht routing framework with aggregation," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 43–48.

[6] C. Dannewitz, M. DAmbrosio, and V. Vercellone, "Hierarchical DHT-based name resolution for information-centric networks," *Computer Communications*, 2013.

[7] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 78–87.

[8] 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*. ACM, 2003, pp. 76–88.

[9] O. Landsiedel, K. A. Lehmann, and K. Wehrle, "T-DHT: topology-based distributed hash tables," in *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2005, pp. 143–144.

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

[11] R. Ahmed, M. F. Bari, S. R. Chowdhury, M. G. Rabbani, R. Boutaba, and B. Mathieu, "αRoute: A Name Based Routing Scheme for Information Centric Networks," in *Proceedings of the 32nd IEEE International Conference on Computer Communications Mini-Conference*, 2013.

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

[13] L. Wang, O. Waltari, and J. Kangasharju, "Mobiccn: Mobility support with greedy routing in content-centric networks," in *Global Communications Conference, 2013 IEEE*. IEEE, 2013, pp. 2069–2075.

[14] S. Roos, L. Wang, T. Strufe, and J. Kangasharju, "Enhancing compact routing in ccn with prefix embedding and topology-aware hashing," in *Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture*. ACM, 2014, pp. 49–54.

[15] R. Perlman, "An algorithm for distributed computation of a spanning tree in an extended LAN," in *ACM SIGCOMM Computer Communication Review*, vol. 15, no. 4. ACM, 1985, pp. 44–53.

[16] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theoretical Computer Science*, vol. 344, no. 1, pp. 3–14, 2005.

[17] "The IPv4 Routed/24 AS Links Dataset-Jue, 2012," $http://www.caida.org/data/active//ipv4\_routed\_topology\_aslinks\_dataset.xml$.

[18] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hungar. Acad. Sci*, vol. 5, pp. 17–61, 1960.

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