H-Pastry: An Inter-domain Topology Aware Overlay for the Support of Name-Resolution Services in the Future Internet

Nikos Fotiou*, Konstantinos V. Katsaros†, George Xylomenos‡, George C. Polyzos§
*‡§Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business
Athens 10434, Greece
fotiou@aueb.gr, xgeorge@aueb.gr, polyzos@aueb.gr
‡Department of Electronic & Electrical Engineering
University College London
WC1E 7JE, Torrington Place, London, UK
k.katsaros@ucl.ac.uk

***Abstract*—Overlay networks are widely used for locating and disseminating information by means of custom routing and forwarding on top of an underlying network. Distributed Hash Table (DHT) based overlays in particular, provide good scalability and load balancing properties. However, these come at the cost of inefficient routing, caused by the lack of adaptation to the underlying network, as DHTs often overlook physical network proximity, administrative boundaries and/or inter-domain routing policies. In this paper we show how to construct a DHT-based overlay network that takes all these aspects into account, so as to ease the global deployment of Future Internet architectures which require large-scale name resolution, such as Information-Centric Networking (ICN) and the Internet of Things (IoT). Based on the Pastry distributed object location and routing substrate and the Canon paradigm for multi-level DHTs, we developed H-Pastry, an overlay DHT scheme that harvests the scalability and load balancing features of DHTs, while also adapting to the underlying network topology, administrative structure and routing policies. We evaluate the performance characteristics of the proposed scheme through an extensive set of detailed simulations over realistic inter-network topologies. Our results show that H-Pastry substantially improves routing by reducing both overlay path stretch (by up to 55%) and routing policy violations (by up to 70%), compared to the Canonical (multi-level) Chord DHT. In addition, the design of H-Pastry keeps traffic within administrative boundaries as far as possible, reducing inter-domain hops by up to 27% compared to Pastry, while also creating excellent opportunities for the support of caching and multicast.**

***Index Terms*—Information-centric networking, Internet of Things, Name resolution, inter-domain, overlay, hierarchical, DHT, Pastry**

I. Introduction

Overlay networks are virtual networks formed by servers operating above the existing inter-networking infrastructure i.e., their functionality is based on higher layers of the protocol stack than those supported by the network layer, and they can be operated by third-parties [1]. Among several overlay network designs, *Distributed Hash Tables* (DHTs) have attracted considerable attention due to important structural advantages, such as their logarithmic scalability and their inherent load balancing capabilities. These characteristics have been considered especially advantageous for the support of emerging Future Internet architectures. In the context of the relatively recent Information-Centric Networking paradigm [2], the network is responsible for locating and delivering the information objects requested by end-hosts, thus necessitating the availability of a scalable name resolution service e.g., [3], [4], [5], [6], [7], [8]. Considering that (a) the current number of unique web pages indexed by Google is greater than 1 trillion [9] and that (b) billions [10], [11] of devices (mobile phones, sensors, home appliances etc.) will be offering additional content to future networks, one should expect that in the context of ICN, any name resolution approach will have to handle unique information objects (IOs) in the order of 10^{13}. Some studies raise this estimate even further to 10^{15} [12]. But even beyond the ICN paradigm, the current expectations for even up to 50 billions of interconnected devices, i.e., sensors and actuators, in the envisioned Internet of Things/Everything (IoT/IoE), call for lookup services of corresponding scalability [11], [13].

Even though DHTs present the desired scalability properties for such a task, the resulting overlay routing is, in principle, solely based on the logical organization of the overlay nodes, often neglecting their location in the physical network. This translates to routing schemes that ignore one or more of the following aspects: (i) physical network proximity, (ii) administrative domain boundaries, (iii) inter-domain routing policies. Several approaches, such as Pastry [14], take physical network proximity into account by incorporating proximity metrics (e.g., hop count, RTT) in the overlay construction process. However, they do not consider administrative domain boundaries. The Canon paradigm takes DHT design one step further by enabling the construction of multi-level DHT-based overlay networks [15]. Canon enables the progressive merging of individual DHT constructions, assuming a hierarchical structure for the inter-domain topology. The resulting overlay networks, usually termed *Canonical*, not only achieve scalability and efficient load balancing, they also facilitate the deployment of fault isolation and security mechanisms, effective caching and bandwidth utilization, hierarchical storage, and hierarchical access control [15].

Nevertheless, even Canonical DHT designs present significant inefficiencies. The Canon DHT merge process was illustrated based on the Chord DHT [16], yielding Crescendo [15], along with sketches of the Canonical versions of several other DHT's, except Pastry. All these designs inherently neglect physical network proximity. To adress this issue, the Crescendo (Prox.) variant offers *proximity adaptation* based on the creation of groups of densely connected nodes sharing the same prefix [15], [17]. However, proximity adaptation is not integrated in the overlay construction process and the corresponding overhead in terms of signaling and routing state has not been investigated. Furthermore, the Canon approach to multi-level DHTs was tailored to a strictly hierarchical domain-level network structure, which limits its applicability in the case of the Internet: as reported in [18], the increasing number of peering relationships as well as multihoming, result in an inter-domain graph that is not strictly hierarchical. These inter-domain relationships are not reflected in the overlay structure, leading to routing policy violations (as further illustrated in Section IV). This issue was addressed in [19] in a completely different networking context, where the DHT construction completely ignores the underlying routing substrate, resulting in highly stretched paths.

As a result, overlay paths in currently available DHT designs tend to be considerably longer than their underlay counterparts, often unnecessarily crossing administrative domain boundaries and/or violating established inter-domain routing policies. These facts actually constitute the basis of the polemic of ISPs against overlays and "justify" their efforts in cutting off overlay traffic whenever possible (e.g. through firewalls, deep packet inspection techniques etc.) [20]. Such inefficiences become of paramount importance when the supported services represent core network functionalities. In the context of ICN architectures and/or the envisioned IoT/IoE, access to any type of information or device is expected to lead to increased volumes of resolution traffic[1] magnifying the impact of inefficient overlay paths on both the quality of experience (QoE) at the edge (e.g., name-resolution delays) and the overall utilization of resources inside the network. In this respect, the need for a DHT design that presents both the highly desired scalability and load balancing properties, and efficient overlay paths that adapt to the structure of the underlying inter-domain network topology, becomes more than apparent.

To this end, in this paper, we present *Hierarchical Pastry* (H-Pastry), a multi-level DHT scheme that aims to bring together the benefits of the Pastry DHT and the Canon approach, adding also support for multihoming and peering relationships. We explore the advantages of the proposed design, based on a series of detailed packet-level simulations, paying particular attention to the structure of the underlying inter-domain network graph and the established inter-domain routing policies. To the best of our knowledge, this is the first work to investigate the performance of DHT-based overlay routing in this context. Our results show that, by taking network proximity into account H-Pastry yields shorter overlay paths, reducing the perceived path stretch by approximately 50% (on average) compared to Crescendo. At the same time, H-Pastry achieves a 67% reduction of inter-domain routing policy violations compared to Crescendo (on average), and reduces inter-domain hops by an average of 23% compared to Pastry, thus better respecting the underlying administrative domain boundaries. Some of the performance benefits of H-Pastry were previously investigated in [6], in the context of a direct comparison between lookup-by-name and route-by-name inter-domain name resolution systems for ICN. In this paper, we present the detailed design of H-Pastry and further demonstrate the resulting performance benefits with a comprehensive performance evaluation of H-Pastry in comparison to other DHTs.

In the following, we first provide background information for Pastry and Canon, introducing their main features and highlighting the addressed inefficiencies (Section II). We then proceed with a detailed description of the proposed design (Section III). In order to investigate the performance characteristics of H-Pastry, we engaged in extensive simulations, comparing H-Pastry against alternative approaches i.e., Pastry, Chord and Crescendo. The results of the performance evaluation are presented in Section IV. Finally, we provide a discussion on related approaches in Section V and conclude in Section VI.

II. Background

A. Pastry

In the Pastry DHT, every node is assigned, uniformly and randomly, a unique identifier (ID). Each ID is a 128-bit number (though other lengths may be used), handled as a sequence of b-bit digits (b is a configuration parameter with a typical value of 4). Given a message and a key, Pastry routes the message to the node with the ID that is numerically closest to the key, in less than $log_{2^b} N$ steps, where N is the number of nodes in the DHT.

The routing state of each Pastry node is organized in two routing structures: the *Routing Table* and the *Leaf Set*. A Routing Table is organized in $\frac{128}{2^b}$ rows, with each row containing $(2^b - 1)$ entries. The entries at row i refer to nodes whose IDs share only the first i digits with the current node's ID (e.g., row 0 contains node IDs that do not share any digit with the current node's ID, row 1 contains node IDs whose first digit is the same as the current node's ID but the second is different, and so on). The Leaf Set contains $|L|$ entries for nodes whose IDs are the numerically closest to the present node's identifier. The set is split in two parts with $|L/2|$ entries for numerically smaller IDs and $|L/2|$ for numerically larger IDs. A typical value for $|L|$ is 2^b.

A Pastry node routes a message with a key K as follows: initially it examines if K falls within the range of the node IDs stored in the Leaf Set; if so, the message is forwarded to

[1]This is due to both the high volume of information objects and devices, and the lack of a host-centric mode of communication where requests are directly targeted to specific end-hosts (e.g., file requests from a specific web server), which nevertheless allows for the seamless support of in-network caching, multicast forwarding and mobility [21].

the node with the ID closest to K, which is the destination node. Otherwise, the Routing Table is used i.e., the message is forwarded to a node that shares a common prefix with K and this prefix is at least one digit longer than the common prefix of K and the current node's ID.

During routing state creation and maintenance, Pastry takes network locality into consideration i.e., among equally qualified candidate Routing Table entries, the one corresponding to the closest node in the network (with respect to the employed proximity metric, e.g., hop count, RTT, etc.) is selected. This feature yields Pastry's *short routes* property and constitutes one of the reasons for basing our work on Pastry. Additionally, according to Pastry's *route convergence* property, the distance traveled by two messages originating from two distinct nodes before their routes converge towards the same destination, tends to be approximately equal to the distance between the two source nodes in the proximity space [14], [22], [23]. This is a highly desirable feature for both mobility support [24] and caching services[21]. It should be noted that this property stems from Pastry's prefix based routing and is not available in any other DHT design.

B. Canon

The importance of adapting the DHT to the underlying network structure was identified by Ganesan *et al.* in [15], where the Canon DHT paradigm was proposed. Canon intervenes in the construction process of a DHT in order to adapt it to the hierarchical structure of the underlying inter-domain topology. At the lowest level of this hierarchical process, each domain creates its own DHT structure. At each higher level, the DHT structures of sibling domains are merged so that a single DHT is created. This process continues up to the highest level, where all individual DHT's are merged into a single overlay network. The purpose of this process is to satisfy the following requirements:

- *Requirement 1*: *Convergence of inter-domain paths*
  All messages originating from a domain A and targeting the same node located at a different domain should exit A through the same node.
- *Requirement 2*: *Locality of intra-domain paths*
  All messages originating from a domain A and targeting a node located in the same domain should never exit A.

In order to satisfy these requirements, the merging process is based on the incremental refinement of the overlay routing state while traversing the domain hierarchy towards the entire network structure. Specifically, a node *u* in a domain *A* creates and maintains a routing entry for a node *v* in a sibling domain *iff* *v*'s ID is closer to *u*'s ID than any other node ID in domain *A*.

Although Canon succeeds in fulfilling the aforementioned requirements, it does not address a series of important issues:

- The Canon design assumes a strictly hierarchical underlying domain-level network graph. Though this assumption adheres to the organizational or the administrative structure of several types of networks (e.g., university, corporate networks), it does not reflect more complex structures found in the Internet, such as multihoming and peering relationships (see Section II-C). Reports from CAIDA [25], research efforts in Internet cartography [26], [27] and large measurement studies such as [18] indicate that the Internet topology is evolving into a mesh graph dominated by peering relationships. Moreover, multihoming is a well established practice for traffic load balancing and reliability. Both types of inter-domain relationships introduce cycles in the tree-like domain level structure of the internetwork, thus limiting the direct applicability of the Canon paradigm. An extension was proposed in [19] to address this issue, based on the insertion of virtual domain nodes in the domain-level network graph. The solution was proposed in the context of a *clean slate* Internet architecture, assuming no availability of IP services. However, the applicability of this approach for the overlay level and, most importantly, its impact on the policy compliance of overlay routing have not been investigated.
- To the best of our knowledge, the only Canonical DHT design that attempts to consider physical network proximity is Crescendo (Prox.) [17]. In this Chord variant, nodes at the highest levels of the hierarchy are clustered based on a *T*-bit prefix of their identifier. Cluster members are densely connected so as to allow routing between them via a single hop, while links between clusters are selected based on proximity. Overlay routing takes place in two steps, first between clusters and then within clusters. However, this scheme does not directly reflect physical network proximity, as it mostly relies on the dense connectivity inside clusters to reduce the length of the overlay paths. Moreover, the size of the routing state and the corresponding signaling overhead required to support the dense overlay connectivity within clusters have not been investigated.
- The Canon design assumes that only the leaf domains of the hierarchical network graph contain nodes that take part in the overlay network; cases where internal, higher level domains also contain nodes that participate in the overlay network are not addressed. This is an important limitation when higher level domains represent distinct administrative areas of the network (as in the Internet inter-domain graph), rather than mere conceptual reflections of a single organization's hierarchical structure.

C. Inter-domain routing

Obviously, understanding inter-domain routing is a crucial factor in designing new Internet applications and/or inter-domain protocols. Inter-domain routing follows the routing relationships established through business agreements between *Autonomous Systems* (ASes). In general, provider ASes offer transit traffic services to customer ASes. As a result, a hierarchy of *provider-to-customer* relationships is formed recursively, in which customers are charged by their direct providers for connectivity and traffic transit services, direct providers are themselves customers of other provider ASes and so forth, up until the highest level of the hierarchy, where top level ASes are connected in a full mesh. ASes tend to establish connectivity with multiple providers for load and cost balancing reasons, as well as for redundancy; this practice is known as *multihoming*.

Apart from provider-to-customer relationships, ASes also establish *peering* relationships for the direct exchange of traffic.[2] The main incentive for establishing peering agreements is to avoid using the costly transit services offered by providers. In the example of Figure 1, peering domains 10 and 11 will directly exchange traffic instead of forwarding it to their ancestor domains 4 and 5 respectively.

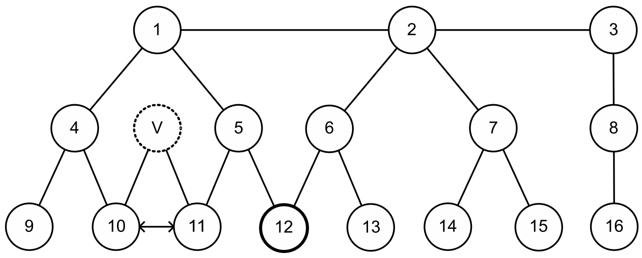

Fig. 1. Example inter-domain topology: each node represents a distinct administrative domain. Domain 12 is multihomed and a peering relationship is established between domains 10 and 11. Domain V is a virtual domain.

An important property of inter-domain routing is the so-called *valley-free* property according to which, the traffic that an AS receives from its provider(s) or peer(s) can only be forwarded to its customers. *Valley-free* (or *transit*) *policy violations* take place when an AS acts as a transit for traffic that originates from its provider(s) or peer(s) and is not destined to its customer(s)[3]. Such violations obviously play an important role as they translate to financial costs for network operators. As discussed in [28], overlay routing decisions inherently neglect the underlying routing policies, as they are performed at the application layer. While with overlay routing each hop in an overlay path does follow a policy compliant underlay path, as it relies on the underlying routing substrate, an end-to-end overlay path consisting of a sequence of such hops may result in transit policy violations.

III. H-Pastry design

In this section we provide a detailed description of H-Pastry. For our description, we represent the inter-domain hierarchy as a graph $G(V,E)$. V is the set of nodes in the graph, with each node representing a distinct administrative domain or AS[4]. E is the set of edges in the graph, with each edge representing the inter-domain link that connects the corresponding domains. We consider a domain hierarchy of H levels, with levels 0 and $H-1$ referring to the highest and lowest levels in G respectively. The level of a domain is defined as the length of the longest valley-free path from this domain to the root of G. Figure 2 shows a graph of 3 levels; domain 1 is at level 0, domains $2,3$ are at level 1, and domains $4,5,6,7$ are at level 2. Finally, we assume that a local Pastry *ring* i.e., a Pastry DHT instance, exists in every domain, regardless of its level.

[2]We omit relationships between ASes which belong to the same organization, as these have been reported to be very rare [26].

[3]A description of the various types of routing policy violations can be found in [28].

[4]Henceforth, we will use both terms interchangeably.

A. DHT formation

An H-Pastry DHT is formed by having all overlay nodes exchange routing state in a recursive manner. Assume an overlay node residing at a domain A at level l which has already joined the Pastry ring for domain A. First, the node will exchange state with the overlay nodes of A's descendant domains in the hierarchy. Then, the node will exchange its (now extended) state with the overlay nodes residing in A's ancestor domain at level $l-1$ *and* that domain's descendant domains (excluding the sub-tree rooted at A, which was covered in the previous step). This procedure is repeated by exchanging routing state with nodes residing at A's ancestor domain at level $l-2$ *and* that domain's descendant domains (excluding again the sub-tree rooted at A's parent domain at level $l-1$) and so forth. Figure 2 illustrates this procedure in a three level example topology, for domains residing at different levels of the hierarchy.

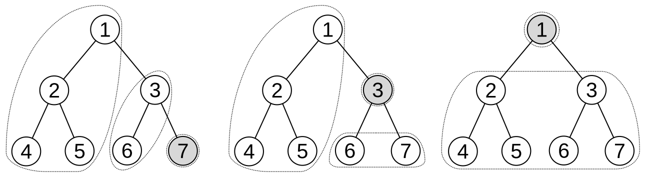

Fig. 2. DHT formation. An overlay node in domain 7 firstly exchanges state with nodes in domains 3 and 6, and then with nodes in domains $1,2,4$ and 5. An overlay node in domain 3 firstly exchanges state with nodes in domain 3's children domains, i.e., domains 6 and 7, and then with domains $1,2,4$ and 5. An overlay node in domain 1 will exchange state with its descendant domains $2,3,4,5,6$ and 7.

Multihoming in G is depicted by edges that connect a multihomed domain to multiple domains at higher levels. The state exchange procedure of H-Pastry does not need to be modified for multihoming. As an example, domain 12 in Figure 1 is multihomed, as it is connected to domains 5 and 6. During the state exchange process, an overlay node from domain 12 will first exchange its state with nodes in domains $5,6,11$ and 13, which correspond to domain 12's parent domains and their children, and then with nodes in the remaining domains.

Peering agreements must also be taken into account during DHT formation, in order for the overlay to take advantage of peering links. In the example of Figure 1, peering domains 10 and 11 directly exchange traffic instead of forwarding it to their ancestor domains 4 and 5 respectively. Ideally, messages exchanged by overlay nodes in peering domains should not traverse other domains, thus extending Canon's *Locality of intra-domain paths* property (see Section II-B) to peering links. Following the approach proposed in [19], we connect the peering domains to a virtual node introduced one level above them, thus making them multihomed. This node does not correspond to any network entity, it is simply used to guide the DHT formation process. Moreover this node is not linked to any other domain, in order to not affect the paths that do not use the peer link. As an example, in Figure 1, a virtual node V is introduced at level 1, linked to 10 and 11. Otherwise, overlay nodes exchange state as previously described. Note that a peering link between sibling domains (e.g., nodes 9 and 10 in Figure 1) does not require the use of this technique, as the nodes already have a (real) common parent domain.

B. *Routing state*

The routing state exchanged during the process described above is structured at each node in accordance to its domain's position in the hierarchy. Specifically, an overlay node located in a domain A at level l of the inter-domain hierarchy maintains $l+2$ sets of routing state, each composed of a Leaf Set and a Routing Table, organized as follows:

- A Leaf Set and a Routing Table for the intra-domain (at A) Pastry ring, referred to as the level $l+1$ routing state.
- A Leaf Set and a Routing Table for the Pastry state related to nodes residing at A's descendant domains, referred to as the level l routing state.
- l Leaf Set and Routing Table pairs for Pastry state related to nodes residing at the l levels of ancestor domains of A (and their corresponding children), referred to as the level $l-1$, $l-2$, ..., 0 routing state.

As an example, an overlay node located in domain 3 at level 1 of the topology depicted in Figure 2 maintains routing state in three Leaf Set and Routing Table sets: level 2 routing state refers to nodes in 3's domain (i.e., local Pastry ring), level 1 routing state refers to nodes in 3's descendant domains (i.e., domains 6 and 7), and level 0 routing state refers to nodes located in domain 3's ascendant domain (domain 1) and its children (domains $2, 4$ and 5). Note that the lowest level of routing state refers to the *local* domain, rather than to the descendant domains. This reflects the fact that, for routing purposes, nodes in the local domain are considered closer to the current node than nodes at descendant domains (see Section III-D).

Routing entries referring to nodes in the local domain, i.e. the level $l+1$ routing state, are handled as in regular Pastry (see Section II-A). Routing entries referring to nodes in other domains however, i.e. the level l to 0 routing state, are a subset of the routing state exchanged with those nodes, in order to avoid the maintenance of excessive routing state. Applying the Canon design principles (see SectionII-B), each node preserves only those routing entries that progressively (as we go up the domain hierarchy) refine the routing information on the identifier space. Specifically, at each level of the routing state hierarchy, a node maintains only routing entries refering to nodes closer to the node's ID than the routing entries maintained at lower levels. Hence, the routing state received from other nodes is *filtered* by applying the following procedure.

Let cID be the current node's identifier. l_i^- denotes the numerically closest smaller *node ID (i.e., routing entry) and l_i^+ denotes the numerically closest* larger *node ID (i.e., routing entry) in the level i Leaf Set. In order for an identifier to be maintained in the routing tables of level $i-1$, it should be numerically smaller than l_i^+ and larger than l_i^-.*

For example, assume an H-Pastry ring with parameters $b = 2$ and $|L| = 4$ that uses 8-bit identifiers and consider a node with $cID = 1A$, located in level 2, that has the following level 3 (local) Leaf Set: $05, 09, 2A, 32$. For this node $l_3^- = 09$ and $l_3^+ = 2A$. Among all the routing entries that will be received for the level 2 routing state, only those for which their node ID falls in the range $09 < ID < 2A$ will be preserved, that is, the level 2 routing state will consist only of entries pointing at nodes numerically closer to the current node than its nearest neighbors at level 3. As we show in Section IV-B5, this filtering procedure results in every overlay node maintaining less state, than the state that would be required if a single Pastry ring was formed by all nodes.

C. *Join procedure*

The H-Pastry node joining procedure assures that a new overlay node will obtain all the required routing state. As in regular Pastry, the *Joining Node* (JN) begins the join procedure using a *Bootstrap Node* (BN). The BN should belong to the same AS as the JN, but if the JN is the first node in its AS to join, then the BN can reside at a different AS, which should be close to the JN's AS with respect to Pastry's proximity metric. The JN issues a JOIN message to the BN as in regular Pastry, targeted to JN's ID i.e., the message shall be delivered to the H-Pastry node which is numerically closest to the JN, using the HPastry routing procedure (discussed in the next subsection). During the join procedure the JN receives routing state for all levels of the hierarchy. This state is sent by all the intermediate nodes that forward the JOIN message to its destination and it is filtered as explained above so that only entries that are within the boundaries of each level are kept. The join procedure completes with the exchange of state between the JN and the nodes that were stored in its Routing Tables, as dictated by the regular Pastry join procedure.

D. *Routing procedure*

Based on the routing information collected during the Join procedure, routing in H-Pastry consists of a sequence of steps that follow the hierarchy of the inter-network, which is reflected in the multiple levels of routing state maintaned by each node. In order to route a message towards a target key K, the receiving node must first determine the level of the routing state that should be used. This is accomplished by Algorithm 1.

The rationale behind this algorithm can best be explained with reference to Figure 3, which illustrates the routing state for a node located in a second level domain, i.e. $l = 1$, since the root is at level 0. This node has three levels of routing state: level 2 that corresponds its own domain, level 1 that corresponds its children, and level 0 that corresponds to its ancestor domains and their descendants in other subtrees. The numerically closest nodes at levels $2, 1$ and 0 are l_2^- and l_2^+, l_1^- and l_1^+ and l_0^- and l_0^+, respectively. The algorithm finds the *numerically lowest* level of routing state for which the corresponding l^- or l^+ entries are closer to the target key than the l^- or l^+ entries of any higher level. Specifically, when the current node has to route a message for a target key K,

Algorithm 1 Level of routing state selection

```
1: $l$ = Current node's domain level
2: $cID$ = Current node's ID
3: $K$ = target key
4: for $i = 0 \to l$ do
5:     $l_{i+1}^{-/2} = (l_{i+1}^{-} + cID)/2$
6:     $l_{i+1}^{+/2} = (cID + l_{i+1}^{+})/2$
7:     if $l_{i+1}^{-/2} \leq K < l_{i+1}^{+/2}$ then
8:         return $i$
9:     end if
10: end for
11: return $l + 1$
```

the above algorithm first examines if K is located in the area between the innermost set of dashed lines i.e., between $l_1^{-/2}$ and $l_1^{+/2}$. If it is located there, it uses the level 0 routing state; otherwise, it examines the area between the second innermost set of dashed lines i.e., between $l_2^{-/2}$ and $l_2^{+/2}$, expanding the part of the identifier space searched. If K is located there, it uses the level 1 routing state; otherwise, it uses the level 2 routing state.

When the appropriate level of routing state is found, the standard Pastry procedure is followed: initially the node examines if K falls within the range of the node IDs stored in the corresponding Leaf Set; if so, the message is forwarded to the node with the ID closest to K, which is also the destination node. If K does not fall within the range of the node IDs stored in the Leaf Set, the corresponding Routing Table is used; the message is forwarded to a node that shares a common prefix with K and this prefix is at least one digit longer than the common prefix of K and current node's ID.

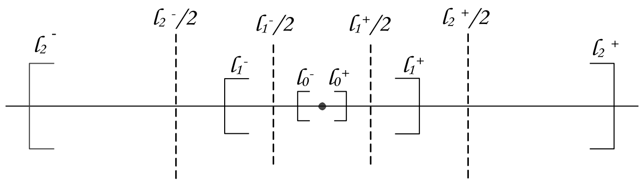

Fig. 3. H-Pastry ID space partitioning for a 3-level hierarchy.

This scheme guarantees that both of the design requirements of Canon are met (see Section II-B). According to the *convergence of inter-domain paths* property, all messages sent from within a domain towards the same key must exit the originating domain via the same node. Indeed, when a node uses a level $i, i < l + 1$ (i.e., non-local) Routing Table, then the above routing procedure guarantees that no other node in the local domain has an ID closer to the target than the current node, otherwise the message would have been forwarded based on the local routing table. As a result, all messages towards a non-local node, always exit the domain via the same node.

According to the *Locality of intra-domain paths* property, a message to a local node will never exit the local domain. Indeed, a message exits a domain with the above routing procedure only if an appropriate entry is found in one of the level $i, i < l + 1$ (i.e., non-local) Routing Tables. However, if the target key resides at the local domain, and it is not the current node, its ID will reside outside the $[l_{l+1}^{-/2}, l_{l+1}^{+/2}]$ interval i.e., an interval that is not handled by any of the level $i, i < l + 1$ Routing Tables. It follows then, that a message towards a target at the local domain, will never exit that domain.

E. Deployment

A critical question related to the deployment of H-Pastry is how can a node learn the network topology graph. This information is crucial for the routing state creation process during which a node should decide on the level of routing state a specific entry should belong to. For example, in Figure 2, the routing entry for a node in domain 7 would belong to level 1 for a node at domain 3, while it would belong to level 0 for a node at domain 1. A rule of thumb that can be used is the following: an entry belongs to the $l - u$ level of state, where l is the current node level, if u is the length of the uphill part of the path that connects the current node and the node that corresponds to the target entry. As an example consider a node in domain 4 of Figure 2 that wants to decide the level of an entry that corresponds to a node in domain 3. The path that connects these two nodes is $4, 2, 1, 3$, where the uphill part of the path is $4, 2, 1$ and its length is 2, hence this entry belongs to level $2 - 2 = 0$. This means that, in order to properly classify routing information, H-Pastry nodes must be able to discover the domain at which the downhill part of a path begins.

This can be achieved by allowing each domain X to be aware of its *up-graph* G_X [29] i.e., a complete list of X's ancestors and their peers in G. Then, an H-Pastry node can discover the domain at which the downhill part of a path begins by simply comparing its own up-graph with that of the target node. This requires the target node's up-graph to also be available; this can be accomplished by including such information in the routing state exchange mechanism.

To this end, the NIRA TIPP protocol can be used by an overlay node to discover its up-graph [29]. A global lookup service is established in [29] for this reason, enabling H-Pastry nodes to get access to their up-graph. However, including this information in the routing state exchange messages of H-Pastry would possibly expose sensitive information regarding the established routing agreements and policies, hindering the adoption of H-Pastry. In order to limit the exposure of such information, we consider the use of Bloom filters [30] to summarize up-graphs in routing state exchange messages. The overlay routing entry for a node Y is extended to also include a Bloom filter BF_Y denoting G_Y. Upon the reception of a routing entry, an overlay node X checks each of the links in G_X against BF_Y, thus determining the level of the Routing Table to be used. Further details of this mechanism (e.g., size of BF and associated overheads, false positive rate) are deferred to future work[5].

It should be noted here that, as we show in Section IV, even if a node has *partial* knowledge of the network graph,

[5]Obviously, this mechanism necessitates the existence of an up-graph lookup service, thus introducing an additional deployment overhead compared to other DHT schemes, albeit along with a series of qualitative and quantitative performance benefits, as shown in Section IV.

H-Pastry still yields significant advantages. This applies even if a node is only able to distinguish which nodes belong to its domain and which do not (i.e., it can only distinguish between local and global nodes).

Another point of consideration with respect to deployment, is the proximity metric(s) that H-Pastry should use. RTT and hop count are two straightforward metrics that can be measured relatively easily by each node using existing tools. ISPs may also provide services that allow using of other, more accurate metrics. For example, an ISP may deploy the Application-Layer Traffic Optimization (ALTO) service [31] which provides nodes with information that helps them perform better peer selection. Such information may include link capacity, node load, cost of a link, and so on.

F. Node failures

H-Pastry inherits Pastry's mechanisms for handling node failures. A node failure is normally detected during forwarding, although a failure can also take place during the join procedure. Message forwarding is not interrupted due to a node failure, unless if $|L/2|$ nodes with adjacent IDs simultaneously fail, which is highly unlikely considering the uniform randomness of node IDs [14]. In most cases, an alternative routing entry can be used to continue the forwarding of a message. In the event of a node failure, H-Pastry follows the routing state maintenance procedure of Pastry i.e., a node chooses the entry in its Routing Tables that is numerically closest to that of the failing node and contacts the corresponding node. That node then provides an alternative routing entry. For local routing information, H-Pastry is obviously identical to regular Pastry. For routing entries in higher level Routing Tables, alternatives entries are requested from all nodes in all routing tables whose IDs are numerically close to the failing node; the received alternative entries are filtered, and the most appropriate is kept. In the event of a concurrent failure of $|L/2|$ nodes with adjacent IDs, all belonging to a level i Leaf Set, the forwarding of a message will not be interrupted as it will be based on level $i-1$ routing information. Thus, routing will only fail in the event of a maximum of $H \cdot |L/2|$ concurrent failures of nodes grouped in H sets of $|L/2|$ adjacent IDs each. Therefore, by hierarchically structuring routing information, H-Pastry is expected to be more robust under node failures compared to regular Pastry.

IV. Performance Evaluation

The performance evaluation of H-Pastry is based on a detailed simulation model[6] developed for the OverSim Simulation framework [32]. In the following, we compare H-Pastry against Crescendo [33], regular Pastry and Chord [16].

A. Methodology

As this work focuses on the routing efficiency of an overlay construction, it becomes crucial to evaluate it based on an accurate model of the underlying network topology. The exact Internet topology remains undisclosed to the research community by the network operators, as their internal connectivity reflects their business operation and is therefore considered confidential. Research studies however converge on the number of the ASes, which is estimated to be more than 40.000 [25]. The identification of peering links on the other hand, remains an open issue. Data traces produced by [25] and [26] diverge on the size of identified peering links, while the study in [27] argues that the methodology followed fails to reveal up to 90% of existing peering links.

Evaluating a new application or protocol using a realistic topology of 40.000 ASes (and approximately 200.000 annotated links) is a technical challenge on its own. To overcome technical constraints, we used the algorithm described in [34] to scale down the measured Internet topologies, thus creating realistic synthetic topologies which i) are of manageable size for evaluation purposes and ii) maintain the same characteristics and business relationships as the measured graphs. In the following, we present results for a 400 domain topology, unless otherwise stated. We vary the size of the overlay node population in order to understand the impact of the overlay network size on the examined performance metrics. The size of the overlay network varies from 1125 to 4499 nodes. In all scenarios, the overlay nodes are uniformly dispersed across the domains, with each node being attached to the corresponding domain's router. In each repetition of the same scenario, the overlay nodes are distributed again, and then we randomly select 200.000 pairs of overlay nodes and route a message between them using the overlay routing fabric. The results we present are the average of five repetitions for each experiment[7]. We note that the randomness of the simulations is limited by the algorithm of [34], which yields a single domain graph instance for a selected topology size.

Our evaluation is based on a series of performance metrics aiming to capture several important aspects of H-Pastry and demonstrate its overall advantages over Pastry, Chord and Crescendo. We first focus on the routing properties of the considered routing schemes with respect to physical network proximity, administrative domain boundaries and inter-domain routing policies. Our goal is to assess the impact of our design choices on the perceived performance of the overlay network and quantify the corresponding benefits.

We then investigate the scalability characteristics of H-Pastry with respect to the size of routing state accumulated. We also consider the effect of enforcing a limit on the maximum number of routing state levels ($l_{max} \leq H-1$). In this case, once the maximum number of levels has been reached, the top-level (with respect to the AS-level hierarchy) routing state set will include routing entries for all the remaining overlay nodes. The comparison between H-Pastry and Crescendo is performed for $l_{max} = 2$, due to Crescendo's implementation restrictions. For H-Pastry we also present results for $l_{max} = 6$, which is the maximum number of levels in the considered topology. Finally, we examine how H-Pastry reacts to node churn, i.e. the removal and addition of overlay nodes over time, focusing

[6]Our complete source code is available under an open source license at http://mm.aueb.gr .

[7]Each figure further presents the 99% confidence intervals for the measured values, although in most cases they are too small to discern.

on the fraction of messages that are dropped (and have to be retransmitted) as the overlay routing state converges.

B. Results

1) Stretch: Figure 4 shows the *stretch* of the delivery paths in our setup. We define stretch as the ratio of the number of underlay hops required for a message to reach its destination based on overlay routing, over the corresponding hop count required by an identical message to reach the same target following the shortest underlay path. Stretch expresses the penalty paid for using the overlay routing fabric, essentially depicting the degree by which an overlay routing scheme takes into account the underlying physical network proximity of the nodes.

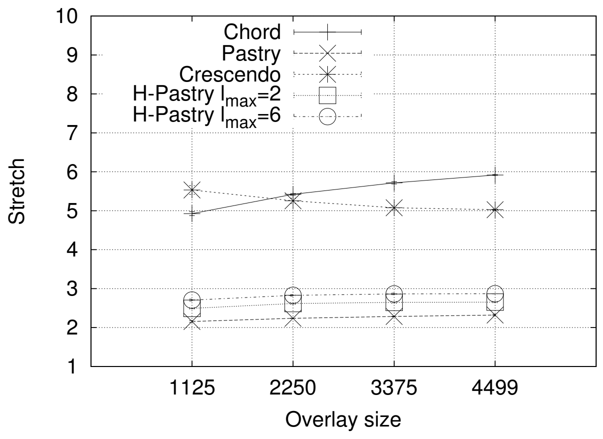

Fig. 4. Routing stretch.

We can see that both Pastry and H-Pastry exhibit similar behavior, outperforming both Chord and Crescendo. H-Pastry reduces stretch by 52% and 50% on average (i.e., across the overlay sizes considered) compared to Chord and Crescendo, respectively, reaching a stretch value of 2.65 for the largest overlay size. This substantial decrease is attributed to the employment of the physical network proximity metric during the creation and maintenance of routing information both in Pastry and H-Pastry. Pastry exhibits a slightly lower stretch value than H-Pastry, due to the structural characteristics of H-Pastry: Pastry's proximity metric may yield shorter routes, which are not followed by H-Pastry due to the restrictions imposed by administrative domain boundaries and inter-domain routing policies. This is also the reason why H-Pastry with 2 levels works better than H-Pastry with 6 levels. This is not the case with Chord and Crescendo, as regular Chord is completely network topology oblivious, therefore Canon's restrictions in Crescendo actually prevent overlay paths from unnecessarily crossing the entire inter-network, thus reducing stretch with larger overlays.

2) Intra-domain routing: Figure 5 illustrates the average length (measured in number of underlay hops) of paths connecting overlay nodes residing in *the same* domain. H-Pastry outperforms all other considered protocols, yielding on average 55% shorter intra-domain paths than Pastry. This is due to the *Locality of intra-domain paths* property of Canonical

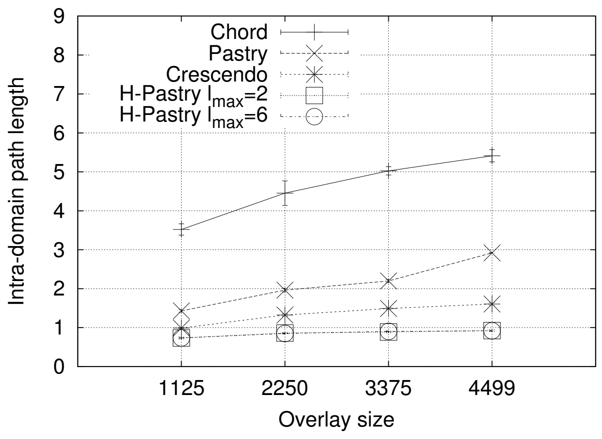

Fig. 5. Intra-domain path lengths.

DHTs, which causes paths between nodes of the same domain to be confined inside domain boundaries. In the case of regular Pastry, routing is based on global routing information i.e., Routing Tables may contain entries for nodes residing on the same or different domains, making no distinction between the two cases. Therefore, paths connecting nodes in the same domain may leave the domain and pass through nodes at different domains. While Crescendo also improves upon plain Pastry due to its Canonical construction, the proximity awareness of H-Pastry allows it to reduce path length by 33% compared to Crescendo. Hence, H-Pastry incorporates both Pastry's *short routes* and Canon's *Locality of intra-domain paths* properties.

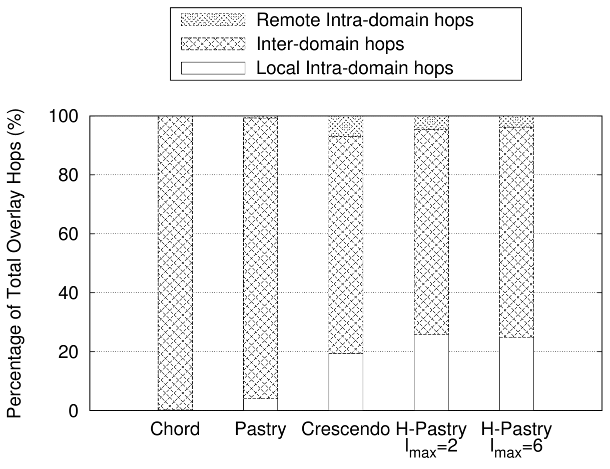

Fig. 6. Distribution of overlay hops

3) Conformance to administrative boundary restrictions: We next delve into the details of overlay routing, by categorizing each of the overlay hops in an overlay path into the following three categories:

- *Local intra-domain hops*: overlay hops within the sender's domain i.e., the number of overlay hops taken before the message (possibly) leaves the domain.
- *Inter-domain hops*: overlay hops across domain boundaries i.e., connecting nodes across different domains.
- *Remote intra-domain hops*: overlay hops between nodes

residing in the same domain, but other than the domain of the sender.

Our target is to assess the effectiveness of the Canonical design with respect to administrative domain boundaries. Figure 6 presents the distribution of overlay routing hops in these categories for each protocol for the largest overlay (4499 nodes); similar results were derived for smaller sizes. Compared to plain Pastry, H-Pastry significantly reduces the Inter-domain part of the resulting overlay paths, by up to 27%. At the same time, the Intra-domain part of the paths increases for both the Local and Remote domains. This is a direct concequence of both the *Locality of intra-domain paths* and the *Convergence of inter-domain paths* properties of the Canon paradigm, which constrain overlay routing within domain boundaries, avoiding unecessary inter-domain hops when possible. H-Pastry performs only slightly better than Crescendo since both use the same (canonical) method to isolate traffic within domains; recall, however, that the actual overlay paths with Crescendo are much longer than with H-Pastry, as shown above.

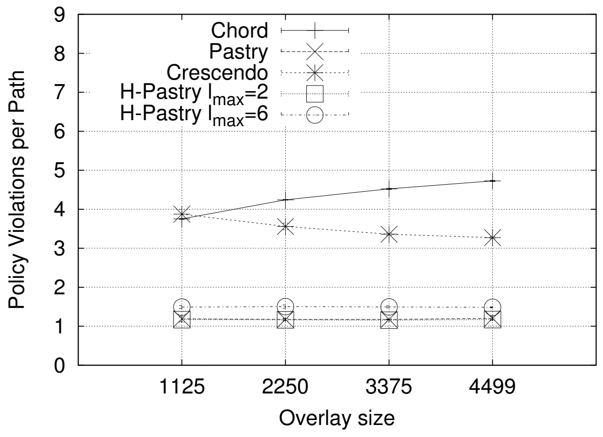

Fig. 7. Average number of policy violations per overlay path.

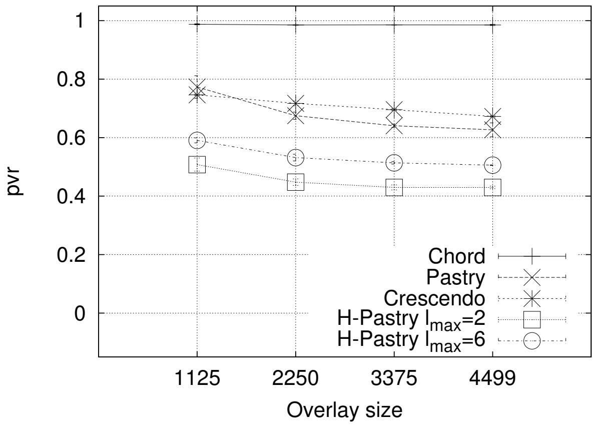

Fig. 8. Policy violation ratio (pvr).

4) Policy violations: We now investigate the compliance of H-Pastry to the routing policies established at the underlying network. In this work we only consider the case of transit (or valley-free) policy violations (see Section II-C). Figure 7 shows the average number of policy violations per overlay path. We see that Pastry and H-Pastry exhibit similar performance, both outperforming Chord and Crescendo. In particular, H-Pastry reduces policy violations by an average of 67% and 73% compared to Crescendo and Chord, respectively. Pastry performs slightly better in this metric because, as explained in Section IV-B1, its overlay paths are slightly shorter than those of H-Pastry, especially when H-Pastry operates with 6 levels, hence they are expected to lead to less violations.

To provide a more fair comparison between the schemes, we define the *policy violation ratio* metric as $pvr = \frac{v}{l-1}$, where l is the number of overlay routing hops in an overlay path, and v the number of policy violations taking place along the path. Essentially, this metric normalizes policy violations to the overlay path length. In the worst case, $l - 1$ policy violations can take place in a l hop overlay path, yielding $pvr = 1$. Figure 8 shows the pvr values for all schemes. Evidently, H-Pastry significantly lowers the *pvr* value, compared to all other considered schemes, including Pastry (33% average reduction). We can therefore conclude that H-Pastry's overlay paths are better adapted to routing policies, when taking into account the fact that H-Pastry results in longer overlay paths than Pastry, due to its adaptation to administrative boundaries.

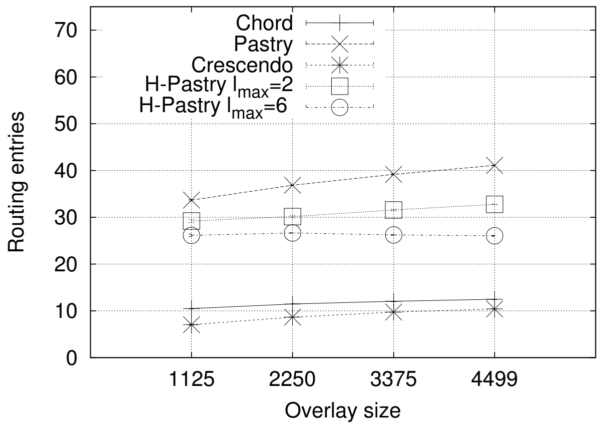

Fig. 9. Routing state size.

5) Routing state: Figure 9 shows the number of routing entries for each protocol considered. In the case of Chord and Crescendo we consider the total number of *fingers*, while in the case of Pastry and H-Pastry we consider the total number of Routing Table entries; we exclude the Leaf Sets, as the corresponding information is also maintained in the Routing Tables. The size of the routing state is clearly larger in the case of Pastry-based protocols, since Pastry by design maintains more pointers to certain areas of the identifier space compared to Chord. However, the overall routing state is not excessive. H-Pastry requires less routing state compared to regular Pastry, due to the layered organization of its routing state and the filtering of each level's routing information, as described in Section III. This is more evident when H-Pastry operates with 6 levels, since in that case routing information filtering is taking place to a greater extent.

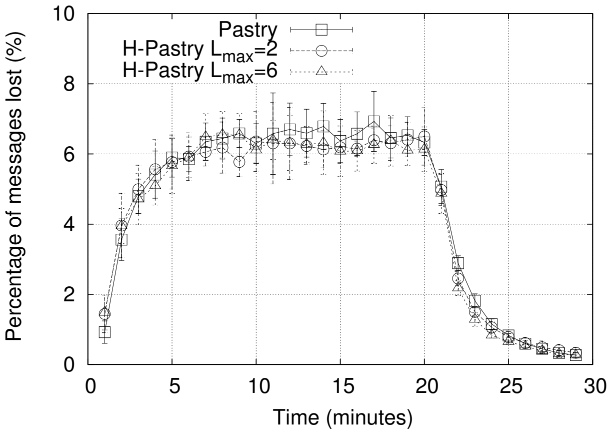

Fig. 10. Message losses due to churn.

6) Impact of churn: In order to study the impact of churn on H-Pastry, we adopted the methodology proposed by Rheat *et al.* [35]. Due to reasons related to the scalability of the simulator, we simulated a smaller topology for these experiments, composed of 100 domains. In each experiment, we consider a warming up period, during which 1000 overlay nodes form an (H-)Pastry ring. After the warming up period, randomly selected nodes fail, following a Poisson process characterized by inter-failure times of 2 sec. Every time a node fails, a new one joins the overlay. Each node continuously sends messages to randomly selected destinations in the identifier space, following a Poisson process with rate 0.1/sec. Nodes verify that a message has reached the next hop using a time-out period of 1.5 sec. When a time-out occurs the sending node considers the message lost and performs the routing state maintenance routine (see Section III-F). After the warming up period, we simulate a churn period of 20 min, followed by a 10 min period without churn. Every minute we measure the percentage of messages that did not arrive at their destination and we repeat each experiment 10 times.

Figure 10 shows the obtained results[8]. It is clear that even this high churn rate has a small impact on the message delivery ratio for both Pastry and H-Pastry. Namely, the percentage of messages lost due to node failures is consistently below 7%. H-Pastry behaves similarly to Pastry during both simulated periods, with a slightly lower message loss rate, within the margins of statistical confidence. Moreover, we also observe that, similarly to Pastry, H-Pastry rapidly recovers from the churn period, requiring only 2 sec to halve the observed message loss, and less than 5 sec to ensure message delivery.

V. Related work

The application of multi-level structures on DHTs has attracted the attention of many researchers who attempted to combine the scalability characteristics of both hierarchical designs and DHTs, so as to further improve overlay routing performance and/or achieve administrative autonomy. In the HIERAS [36] and Coral [37] schemes, overlay nodes are organized in a hierarchy of clusters based on the link latency/RTT between them i.e., the lower the cluster layer, the lower the average link latency between cluster nodes. A separate DHT ring is then created for each cluster, leading to nodes participating in several rings. In HIERAS, the clustering method is based on a distributed binning scheme, while a similar random sampling method is employed by Coral. In both cases, cluster formation and discovery incurs additional signaling overhead to the overlay construction process. Based on Pastry's joining process, H-Pastry also uses probing for the selection of nearby nodes, however it also takes into account the already retrieved topology information during the join process [14]. Moreover, HIERAS and Coral clusters are not disjoint, thus allowing for duplicate routing entries, while H-Pastry carefully splits the identifier space, with top level Routing Tables containing only entries not present in lower level tables.

This issue is also addressed by an alternative approach, in which the multilevel structure is achieved based on the existence of *superpeers* i.e., nodes responsible for the communication between disjoint clusters of nodes [38], [39]. However, this approach poses significant capacity and availability requirements for the superpeers and heavily affects the load balancing character of DHTs. The multilevel scheme presented in [40] balances between the superpeer approach and those that require all nodes to enter all rings of the hierarchy. Its drawback is that it relies on the Scribe multicast scheme [22] for the communication between independent rings, thus incurring additional signaling and state overhead. On the other hand, this solution allows operating different DHT schemes at each administrative domain.

SkipNet [41] is a DHT-based overlay network which provides controlled data placement (instead of the usual random placement). Dannewitz *et al.* [4] proposed a hierarchical version of SkipNet, which achieves low resolution delays. In their design they consider *resolution domains* organized into a global, tree-like hierarchy topologically embedded in the underlying network. Lower-level resolution domains are mapped to lower-level networks and higher-level resolution domains are mapped to higher-level networks. Higher-level resolution domains are built by combining and interconnecting all lower-level domains nodes of its subtree. This approach however, unrealistically assumes a full *k*-ary tree model for the underlying inter-domain topology structure.

The approach that is probably closest to the Canon design is Cyclone [42]. In Cyclone node identifiers consist of two parts, the *prefix*, which is used as the regular identifier of the node in the selected DHT scheme, and the *suffix* that denotes the cluster the node belongs to. Enforcing a strict numbering of clusters raises scalability concerns, as all nodes must coordinate for the sake of consistency, a requirement not present in H-Pastry. Moreover, Cyclone routing necessitates the traversal of the entire hierarchy of clusters in order to reach the target node, starting from the local cluster and proceeding to higher layers. In contrast, H-Pastry's Routing Tables structure allows a target key to be searched at several levels of the hierarchy, starting from higher level Routing Tables which contain finer grained information, thus possibly

[8]Similar results where derived for different time-out and failure rate values.

skipping several hops.

Apart from DHTs, other hierarchical overlay systems have also been considered by contemporary networking architectures. The Domain Name System (DNS) is probably the most popular hierarchical name lookup service. Although DNS is mainly used for resolving domain names, it can be extended to support other types of lookups. For example, Sevillia *et al.* [43] use DNS to store *content records*, which are then used by their ICN architecture. The main advantage of DNS is its widespread usage, therefore the incremental deployment and adoption of a DNS-based system is easier. On the other hand, it is considered inadequate for storing object-level information since it is susceptible to security attacks, such as Denial of Service (DoS) ones [44], with root servers having a disproportionally larger load than ones at lower levels of the hierarchy.

The Data-Oriented (and beyond) Network Architecture (DONA) [7] considers a hierarchical overlay network composed of Resolution Handlers (RHs). The deployment of the RHs strictly follows the AS-level structure of the Internet, allowing DONA to directly adapt its structure and operation to the underlying network. Content providers issue REGISTER messages to advertise their content to the closest RH which then propagates this information upwards in the hierarchy. Name lookup uses a similar propagation of requests (FIND messages). The CURLING [8] design follows a similar approach with DONA, in that it adapts to the inter-domain topology structure. However, in CURLING, resolution servers propagate registration and resolution requests only to their provider ASes. Both approaches suffer from scalability issues due to the substantial replication of the overall name resolution state and a heavily skewed distribution of the corresponding load throughout the network [45].

VI. Conclusions

In this paper we presented the detailed design of H-Pastry, a multi-level DHT scheme based on the Pastry DHT and the Canon multi-level paradigm. H-Pastry aims at combining the benefits of a multi-level DHT construction with the *short routes* and *route convergence* properties of Pastry. In addition, H-Pastry takes into account the inter-domain routing policies at the underlay level and the increasing prevalence of peering links that make the Internet less hierarchical. By taking all these aspects into consideration, H-Pastry yields significant improvements for overlay routing. Namely, H-Pastry yields considerably shorter overlay paths compared to Crescendo, it reflects the underlying administrative structure of the network by reducing inter-domain hops compared to both Pastry and Crescendo and it leads to a lower routing policy violation ratio than both Pastry and Crescendo. At the same time, H-Pastry requires less routing state than Pastry and is equally resilient to node churn. To the best of our knowledge, this is the first work to consider this rich set of aspects relating to the adaptation of a DHT design to the underlying network. This is of particular importance in view of the recent research interest in DHTs for the support of very large scale inter-domain name resolution in ICNs. Indeed, we have explored the suitability of H-Pastry for this purpose, with very promising results [6].

References

References

[1] D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawski, “Overlay Networks and the Future of the Internet,” *Communication & Strategies*, vol. 63, pp. 109–129, 2006.

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, “A Survey of Information-Centric Networking Research,” *Communications Surveys Tutorials, IEEE*, vol. 16, no. 2, pp. 1024–1049, Second 2014.

[3] R. Jarno, M. Särelä, K. Visala, and J. Riihijärvi, “On name-based inter-domain routing,” *Computer Networks*, vol. 55, no. 4, pp. 975–986, March 2011.

[4] C. Dannewitz, M. D’Ambrosio, and V. Vercellone, “Hierarchical DHT-based name resolution for information-centric networks,” *Computer Communications*, vol. 36, no. 7, pp. 736–749, Apr. 2013.

[5] H. Liu, X. De Foy, and D. Zhang, “A multi-level DHT routing framework with aggregation,” in *Proc. of the ACM SIGCOMM Workshop on Information-centric networking (ICN)*, 2012, pp. 43–48.

[6] K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, “On inter-domain name resolution for information-centric networks,” in *Proc. of the IFIP TC 6 Conference on Networking*, 2012, pp. 13–26.

[7] 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 *Proc. of the ACM SIGCOMM Conference*, 2007, pp. 181–192.

[8] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. de Blas, F. Ramon-Salguero, L. Liang, S. Spirou, A. Beben, and E. Hadjioannou, “Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services,” *IEEE Communications Magazine*, vol. 49, no. 3, pp. 112 –120, march 2011.

[9] Google, *We knew the web was big, July 2008*. [Online]. Available: http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

[10] Cisco, *Cisco Visual Vetworking Index 2010-2015, June 2011*.

[11] D. Evans, “The Internet of Everything: How More Relevant and Valuable Connections Will Change the World,” White Paper, February 2012. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoE.pdf

[12] M. D’Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, “MDHT: a hierarchical name resolution service for information-centric networks,” in *Proc. of the ACM SIGCOMM Workshop on Information-centric networking (ICN)*, 2011, pp. 7–12.

[13] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, “Internet of things: Vision, applications and research challenges,” *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497 – 1516, 2012.

[14] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale Peer-to-Peer systems,” in *Proc. of the Middleware Conference*, 2001, pp. 329–350.

[15] P. Ganesan, K. Gummadi, and H. Garcia-Molina, “Canon in G Major: Designing DHTs with Hierarchical Structure,” in *Proc. of the International Conference on Distributed Computer Systems (ICDCS)*, 2004, pp. 263–272.

[16] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for Internet applications,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[17] P. Ganesan, K. Gummadi, and H. Garcia-Molina, “Canon in G Major: Designing DHTs with Hierarchical Structure,” Stanford University, Tech. Rep., 2003. [Online]. Available: http://dbpubs.stanford.edu/pub/2003-74

[18] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, “Internet inter-domain traffic,” in *Proc. of the ACM SIGCOMM Conference*, 2010, pp. 75–86.

[19] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, “ROFL: routing on flat labels,” in *Proc. of the ACM SIGCOMM Conference*, 2006, pp. 363–374.

[20] M. Piatek, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson, “Pitfalls for ISP-friendly P2P design,” in *Proc. of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2009.

[21] K. Katsaros, G. Xylomenos, and G. C. Polyzos, “MultiCache: An overlay architecture for information-centric networking,” *Computer Networks*, vol. 55, no. 4, pp. 936–947, March 2011.

[22] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, “SCRIBE: A large-scale and decentralized application-level multicast infrastructure,” *IEEE JSAC*, vol. 20, no. 8, pp. 100–110, 2002.

[23] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Microsoft Research, Tech. Rep. MSR-RT-2002-82, 2002.

[24] K. Katsaros, N. Fotiou, G. Polyzos, and G. Xylomenos, "Overlay Multicast Assisted Mobility for Future Publish/Subscribe Networks," in *Proc. of the ICT Mobile Summit*, 2009.

[25] CAIDA, July 2011. [Online]. Available: http://www.caida.org

[26] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang, "The (in)completeness of the observed Internet AS-level structure," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 109–122, February 2010.

[27] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao, "Where the sidewalk ends: extending the Internet as graph using traceroutes from P2P users," in *Proc. of the ACM CoNEXT*, 2009, pp. 217–228.

[28] S. Seetharaman and M. Ammar, "Inter-domain policy violations in multi-hop overlay routes: Analysis and mitigation," *Computer Networks*, vol. 53, no. 1, pp. 60–80, January 2009.

[29] X. Yang, D. Clark, and A. W. Berger, "NIRA: a new inter-domain routing architecture," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 775–788, August 2007.

[30] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[31] S. Kiesel, S. Previdi, M. Stiemerling, R. Woundy, and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements," RFC 6708, September 2012.

[32] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proc. of the IEEE GI Symposium*, 2007, pp. 79–84.

[33] G. Nomikos, "Implementation of Hierarchical Chord (Crescendo)," Master's thesis, Mobile Multimedia Laboratory (MMLab), Athens University of Economics and Business, Athens, Greece, 2009.

[34] X. Dimitropoulos, D. Krioukov, A. Vahdat, and G. Riley, "Graph annotations in modeling complex network topologies," *ACM Transactions on Modeling and Computer Simulation*, vol. 19, no. 4, pp. 17:1–17:29, November 2009.

[35] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proc. of the USENIX Annual Technical Conference (ATEC)*, 2004, pp. 10–10.

[36] Z. Xu, R. Min, and Y. Hu, "HIERAS: a DHT based hierarchical P2P routing algorithm," in *Proc. of the IEEE ICPP*, 2003, pp. 187–194.

[37] M. Freedman and D. Mazieres, "Sloppy hashing and self-organizing clusters," in *Proc. of the International Conference on Peer-to-Peer Systems (IPTPS)*, 2003, pp. 45–55.

[38] L. Garcés-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller, "Hierarchical peer-to-peer systems," *Parallel Processing Letters*, vol. 13, no. 4, pp. 643–657, 2003.

[39] S. Zoels, Z. Despotovic, and W. Kellerer, "On hierarchical DHT systems - An analytical approach for optimal designs," *Computer Communications*, vol. 31, no. 3, pp. 576–590, February 2008.

[40] A. Mislove and P. Druschel, "Providing administrative control and autonomy in structured peer-to-peer overlays," in *Proc. of the International Conference on Peer-to-Peer Systems (IPTPS)*, 2004, pp. 162–172.

[41] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "Skipnet: A scalable overlay network with practical locality properties," in *Proc. of the USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003, pp. 9–9.

[42] M. S. Artigas, P. G. Lopez, J. P. Ahullo, and A. F. Gomez Skarmeta, "Cyclone: A Novel Design Schema for Hierarchical DHTs," in *Proc. of the IEEE Peer to Peer Computing Conference (P2P)*, 2005, pp. 49–56.

[43] S. Sevilla, P. Mahadevan, and J. Garcia-Luna-Aceves, "iDNS: Enabling Information Centric Networking through the DNS," in *Proc. of the IEEE INFOCOM Workshop on Name Oriented Mobility*, 2014, pp. 476–481.

[44] K. Schomp, M. Allman, and M. Rabinovich, "DNS resolvers considered harmful," in *Proc. of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2014, pp. 1–7.

[45] X. Vasilakos, K. Katsaros, and G. Xylomenos, "Cloud computing for global name-resolution in information-centric networks," in *Proc. of the Network Cloud Computing and Applications Symposium (NCCA)*, 2012, pp. 88–94.