αRoute: Routing on Names

Reaz Ahmed, Md. Faizul Bari, Shihabur Rahman Chowdhury, *Student Member, IEEE*, Md. Golam Rabbani, Raouf Boutaba, *Fellow, IEEE*, and Bertrand Mathieu

***Abstract*—One of the crucial building blocks for Information Centric Networking (ICN) is a name based routing scheme that can route directly on content names instead of IP addresses. However, moving the address space from IP addresses to content names brings the scalability issues to a whole new level, due to two reasons. First, name aggregation is not as trivial a task as the IP address aggregation in BGP routing. Second, the number of addressable contents in the Internet is several orders of magnitude higher than the number of IP addresses. With the current size of the Internet, name based, anycast routing is very challenging specially when routing efficiency is of prime importance. We propose a name-based routing scheme αRoute for ICN that offers efficient bandwidth usage, guaranteed content lookup and scalable routing table size. αRoute consists of two components: an alphanumeric Distributed Hash Table (DHT) and an overlay to underlay (Internet topology) mapping algorithm. Simulation results show that αRoute performs significantly better than Content Centric Network (CCN) in terms of network bandwidth usage, lookup latency and load balancing.**

***Index Terms*—Distributed hash table, information centric networks, next generation networking, overlay networks.**

I. Introduction

TODAY'S Internet exists as an interconnection of thousands of Autonomous Systems (ASs) from around the globe. In the current Internet paradigm, end hosts provide information and services, while the network core acts as a passive entity for information exchange. This philosophy places more emphasis on a content's location (i.e., end hosts) than the content itself. The end-users on the contrary, care more about the information they get from the Internet and place less or no emphasis on its location. This mismatch in the current Internet philosophy and users' demand along with the passiveness of the Internet core is causing problems like extraneous bandwidth usage for information access, difficulty in information relocation, and conflict of interest between information providers and information carriers. These issues are motivating researchers to look for an alternate Internet architecture. The emergence of Information Centric Networking (ICN) is one such alternative.

Manuscript received March 25, 2015; revised September 23, 2015 and October 29, 2015; accepted November 06, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor D. Medhi.

R. Ahmed, M. F. Bari, S. R. Chowdhury, and R. Boutaba are with the School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: r5ahmed@uwaterloo.ca).

M. G. Rabbani is with Amazon.com, Inc., Vancouver, BC V6B 0M3, Canada.

B. Mathieu is with the Orange Labs, 22300 Lannion, France.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2506617

ICN has recently received significant attention in the research community. ICN philosophy prioritizes a content ("what") over its location ("where"). Contents can be replicated at different locations. A content request is forwarded to a nearby copy of the content and served from that location. To realize this separation of a content from its location, a name based anycast routing mechanism is essential. However, a number of crucial issues and challenges related to name based routing are yet to be addressed in order to successfully realize a content oriented networking model for the future Internet. One major issue among them is routing scalability. The biggest Internet routing table today contains around 4×10^5 Border Gateway Protocol (BGP) [1] routes for covering about 3.8×10^9 IPv4 addresses and 6×10^8 hosts. This 10^4 scaling factor between IPv4 addresses and BGP routes is achieved by prefix based routing and route aggregation. However, the number of addressable ICN contents is expected to be several orders of magnitude higher that IPv4 address space. Google has indexed approximately 10^{12} URLs [2], which would impose 7 orders of magnitude scalability requirement on a name-based routing scheme analogous to BGP.

The routing scalability issue in ICN is also related to how contents are named and how inter-AS and intra-AS routing protocols process these names. Even if an inter-AS ICN routing protocol covers only the top-level domains as prefixes, it will need to carry approximately 2×10^8 unique prefix routes [3], as no aggregation is possible at this level. So, ICN requires Internet routers to maintain an extremely large amount of routing state, which does not seem to be possible with existing technology [4]. However, in reality the scalability requirement will be much higher than that in BGP due to the following reasons: (i) content names are not as aggregatable as IP addresses, (ii) unlike IP addresses, names with same prefixes may not be advertised from nearby network locations, (iii) routing cannot depend on topological prefix binding as content retrieval should be location independent, (iv) restricting the content name to some specialized format limits the usability of the system, and finally, (v) supporting content replication and mobility reduces the degree of route aggregation that, as we will need to maintain multiple routes in the routing table for the same content.

In this paper we address the routing scalability issue for ICN. We propose αRoute (for alphabet-based routing), a name-based overlay routing scheme, which is scalable and offers content lookup guarantee (Section III). Both the routing table size and the number of hops for content lookup in αRoute are logarithmically bounded by network size. For Internet inter-domain routing using αRoute, we propose a distributed overlay-to-underlay mapping scheme that enables near shortest path routing in underlay (AS-network) by preserving the adjacency relations in the overlay graph (Section IV). We

also provide mathematical bounds on the routing scalability of αRoute and the operating range of our mapping scheme (Section V). We also provide qualitative comparison of our approach with CCN [5] in Section VIII. Finally, we conclude and outline future research directions in Section IX.

II. Design Considerations

The inter-domain routing mechanism will play a major role towards the success of any Internet scale ICN deployment. In addition to packet forwarding, the routing mechanism should provide some additional features to improve overall system performance. In the following we enumerate a set of desirable features for an inter-domain ICN routing mechanism:

1) *Name-based, any-cast routing*: In ICN paradigm, a content is separated from its network location. A user desires a content, while the content provider is of secondary importance. Any-cast routing, i.e., serving a content from any valid replica of the content, is important in ICN context. For performance reasons, it is desirable to directly forward a query to a valid copy of the requested content. In today's Internet, this involves a few steps, e.g., searching a content using descriptive keywords (like in Google or Yahoo search), URL-to-IP lookup, and finally routing to the content server.
2) *Bandwidth efficiency*: The Internet AS topology is hierarchical. An upper level AS has much higher connectivity and capacity compared to a lower level AS. ASs have complicated routing policies driven by economics. In an inter-domain ICN deployment, routing should follow the shortest possible path adhering to AS level policy agreements and should scale with network size.
3) *Lookup guarantee*: Online content popularity is highly skewed [6], [7]. Caching can be used to efficiently lookup popular contents, while routing to less popular contents usually requires an indexing mechanism. The later involves two steps, index lookup and then routing to the actual content. A name-based routing mechanism should efficiently route queries to a target content regardless of its popularity. The routing mechanism should not require excessive time or bandwidth in determining if there is no content matching the query.
4) *Content and index placement freedom*: The network is content-aware in ICN paradigm. As a result the infrastructure providers, i.e. the ASs, should have the freedom of choosing the contents they will be caching and hosting based on their economic, security and performance criteria. Moreover, if an indexing mechanism is adopted for faster content lookup, an AS should have the freedom of selecting the indexes it is willing to host.
5) *In-network caching*: It is a core concept in ICN. Instead of serving contents from the hosts at network edge, ICN proposes to cache or replicate contents in routers. The routing mechanism should leverage this feature to direct a query to the replica, closest to a requester.
6) *Routing scalability*: The scalability of a routing mechanism is largely dependent on routing table size. In a name-based routing mechanism routing table entries are content indexes, which can be used for routing queries to a desired content or its replica. To ensure scalability, the routing table size should grow sub-linearly with the number of contents and network size.

The design implications to satisfy these features may conflict with each other. For example, content placement freedom can be achieved by allowing any content to be placed anywhere. This will cause the routing table to grow linearly with content population in order to ensure guaranteed lookup and bandwidth efficiency. Thus, it is difficult to develop a routing algorithm that simultaneously satisfies all of these desirable features. Instead, our aim is to develop a routing algorithm that satisfies each of these features as closely as possible.

Using a DHT-based structured overlay network, we can achieve all but two of the desirable features for inter-domain ICN routing: a) one-step (or direct) name-based routing and b) index placement freedom. DHT techniques place content indexes at designated nodes for guaranteeing content lookup in a small number of hops and using a small number of routing table entries—both scaling logarithmicaly with network size. For this reason, index placement freedom is not achievable using a DHT-based solution. In addition, a DHT requires two-step routing for content access: first, route to the indexing node to obtain content location(s) and then, route to the actual content. By adopting proper replication strategies we can achieve index placement freedom and one-step content routing to some extent. We explain these strategies in Section IV.

In order to obtain an efficient inter-domain routing mechanism, the DHT overlay has to match the underlying AS-topology as closely as possible. Physical links in the AS-topology are fixed and predefined. On the other hand, in an inter-domain DHT, two ASs with adjacent DHT IDs have to be connected through a logical link, which may map to a physical path in the AS-topology. Thus, we have to assign each AS an identifier respecting both the physical topology and the DHT constraints, which is known to be an NP-hard problem [8], [9]. In the following sections, we present a name-based DHT and a scheme for mapping the DHT to an AS topology.

III. αRoute: A Name-Based DHT

αRoute's design revolves around a logical partitioning tree, which is a core component in αRoute DHT. Each node in this logical tree is assigned a unique alphanumeric string, which determines the names that should be stored in the node. A child node extends its parent's prefix and partitions the names assigned to its parent. In order to embed this logical tree (i.e., the DHT overlay) over an AS network, we propose an embedding algorithm that selectively assigns some tree nodes to the ASs. The embedding process tries to map the logical node hierarchy on the AS network hierarchy, while retaining the one-to-one correspondence between the overlay routing links and physical inter-AS links. Each AS maintains the necessary rules for forwarding messages to other ASs based on the logical tree.

A DHT essentially maps a key to a value in a distributed manner. A DHT design involves two components: a) *partitioning*: segregating the entire key-space into subspaces and assigning each subspace to a physical node, and b) *routing*: a mechanism for locating any key in a bounded number of hops. Now, we present these two components of αRoute.

A. Partitioning

Strings (e.g., URLs or names) in the key space of a name-based DHT can be partitioned in different ways, including *lexicographical ordering*, *hashing* and *edit-distance*. Unfortunately, none of these partitioning strategies is suitable for name-based routing. For the first case, one can sort the strings into lexicographic or dictionary order and create partitions based on character prefixes. This approach will allow prefix matching only and may generate a large number of small-sized partitions depending on prefix length. Second for the hashing strategy, one can hash strings into numbers, and partition based on numeric ranges. But in this approach, string similarity can not be preserved—even one character difference will place two strings in separate partitions. Finally, one can use the edit-distance between strings to group similar strings into the same partition. Unfortunately, there exists no straight forward mechanism for creating partitions based on edit-distance.

Beyond simple name (string) matching, it may be beneficial to support routing based on content semantics. However, semantic-based partitioning is a much harder problem. A possible way to achieve semantic-based partitioning is to attach semantic tags with each content. In addition, standard keywords (semantics vocabulary) and possibly some ontology or hierarchical relation between the semantic tags will be needed to aid the matching.

αRoute builds upon a partitioning strategy based on the presence or absence of certain characters in a given string. This partitioning policy has three desirable characteristics: first, it places similar names in the same partition; second, it provides an upper-bound on the number of partitions; and third, it creates disjoint partitions. We create some partitioning sets (S_i) over an arbitrary alphabet $\mathcal{A}$ based on the expected frequency of each character. Then we create a logical partitioning tree by associating S_i with the i-th level of the tree. Finally, we expand each node at level i using the permutations of character-presence in S_i.

An example of this partitioning concept is shown in Fig. 1, where the logical tree has been expanded up to height three. The first, second and third level partitioning sets in this example are $S_1 = \{r, c\}$, $S_2 = \{e\}$, and $S_3 = \{k, t\}$, respectively. For this example, we assume that the presence or absence combinations of characters r and c yield the most balanced partitioning of the names in a given corpus. More explanation on the partitioning process follows. Now for level 1, we get four partitions rc, $r\bar{c}$, $\bar{r}c$ and $\bar{r}\bar{c}$ based on the characters in S_1. All strings containing the characters r and c will be partitioned under the node rc, while the strings that do not contain neither r nor c, will be placed under partition $\bar{r}\bar{c}$. Similarly, the strings that contain r but not c will belong to the node $r\bar{c}$, whereas the strings containing c but not r will belong to the node $\bar{r}c$. This partitioning strategy can be applied recursively on each node. For example, consider the shaded node (A) in Fig. 1, it corresponds to the partition $\bar{r}c - \bar{e} - k\bar{t}$. Node A will contain all strings (i.e., indexes) that has the characters c and k, but not r, e and t.

Here is an example of mapping a URL to a node. We treat a string (i.e., a content name or URL) as an unordered character set; e.g., the string "www.rocket.com" is treated as {w, r, o, c, k, e, t, c, m}. According to the tree in Fig. 1, "www.rocket.com" will be assigned to the left-most leaf node in the tree with prefix $rc - e - kt$.

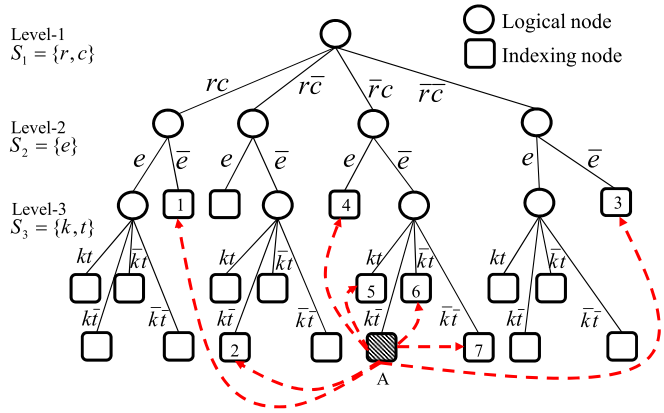


Fig. 1. An example partitioning tree.

It is worth mentioning that the partitioning sets (S_i) are precomputed and influences the distribution of indexing load across the network. To achieve proper load balancing, we need to ensure that for any node the frequency of each combination leading to its children is roughly equal. For example, in Fig. 1 the joint frequencies of rc, $r\bar{c}$, $\bar{r}c$ and $\bar{r}\bar{c}$ should be similar. To fulfill this requirement the partitioning set is generated by a three step process: (i) a large corpus of expected content names are parsed to compute character frequencies, (ii) all possible combinations of characters up to a fixed length ζ are generated, and (iii) the combination that produces the most balanced branching is selected. For example, let us assume that the tree is grown up to height 2 with the characters r, c, and e. Now, when we want to extend the tree height, we first compute the character frequency of the content names that are partitioned under the 8 nodes at height 2. Then we generate all possible character combinations in $\mathcal{A}\}$ (leaving out r, c, and e) of length at most ζ and select the combination that produces the most balanced branching at level 3, which is $\{k, t\}$ in Fig. 1. Generating all possible combinations and lengths is computationally expensive. Hence, we investigate combinations up to a pre-specified length ζ. The partitioning set can be precomputed over a name corpus and it can be expected that the partitioning sets $(\{S_i\})$ computed over a fairly large corpus will evenly distribute content names across the partitioning tree nodes in a deployment.

In Algorithm 1, we present the pseudo code for building the αRoute tree. It takes two inputs: desired number of nodes in the tree (n) and the partitioning sets $\{S_i\}$. In line 1 and 2 the root node of the tree is initialized. The while loop at line 4 iterates until the tree is grown to the desired size n. In line 5, a tree node with minimum height and maximum load is selected for being extended in the current iteration. In line 6, the *GetPatterns* function returns all possible character present-absent (similar to binary 0-1 sequence) sequences based on the partitioning set. The for loop from line 6 to 10 traverses over the patterns returned by the *GetPatterns* functions and creates child nodes by extending their parent's pattern with the current pattern. In the loop from line 12 to 14 the routing links for each leaf node are generated

Algorithm 1 BUILDALPHATREE $(n, \{S_i\})$

Require: Desired tree size n and partitioning sets $\{S_i\}$.
Ensure: Builds the αRoute partition tree.

```
root ← new TreeNode()
root.[height, prefix] ← [0, ∅]
nodeCount ← 1
while nodeCount < n do
    node ← node with min. height and max. load
    for each pattern, p ∈ GetPatterns(S_{node.height}) do
        child ← new TreeNode(node)
        child.prefix ← node.prefix + p
        nodeCount ← nodeCount + 1
    end for
end while
for each node ∈ GetLeafNodes(root) do
    node.links ← routing links based on (1)
end for
```

and saved in the $node.links$ data structure. These routing links represent the overlay links in the DHT, which can be computed by using (1). In this strategy, tree height will grow with network size and the maximum tree height is determined by the alphabet $\mathcal{A}$.

B. Routing

Our routing mechanism has two components: routing table and message forwarding mechanism. Ideally the routing table should be logarithmic on network size, while the forwarding mechanism should ensure shortest path routing using local information only. In this section we present an overlay routing mechanism which achieves both of these goals. In the next section we will present a mapping algorithm to achieve these goals in an underlay network.

Routing Table: Each node in the logical partitioning tree can be identified by a *pattern* $s_1 s_2 \dots s_h$, where s_i is a character presence combination over the characters of S_i and h is the height of the tree. For example, in Fig. 1, there are three partitioning sets, $S_1 = \{r, c\}$, $S_2 = \{e\}$ and $S_3 = \{k, t\}$. The pattern assigned to the shaded node (A) in Fig. 1 is $\bar{r}c - \bar{e} - k\bar{t}$. The first level partition s_1 (constructed from the characters in S_1) for this node is $\bar{r}c$, the second level partition s_2 is $\bar{e}$ and the third level partition s_3 is $k\bar{t}$.

Each leaf node of the logical partitioning tree corresponds to an AS in the Internet. We can describe the routing table entries for the AS responsible for partition $s_1 s_2 \dots s_i \dots s_h$ as follows. For some level i, the AS's routing table will have $2^{|S_i|} - 1$ routing links corresponding to the partitions $s_1 s_2 \dots f_j(s_i) \dots s_h$, where $f_j(s_i)$ is a function that inverts the presence status of the jth character in s_i. In other words if s_i corresponds to the presence of the jth character then $f_j(s_i)$ will represent the absence of the jth character and vice versa. For example, if $s_i = k\bar{t}$, then $f_2(s_i) = kt$. In general, the routing table at each AS will have $\sum_{i=1}^{h}(2^{|S_i|} - 1)$ entries. We can express the routing table entries for any AS as follows:

$$\begin{aligned} &Routing - links(s_1 s_2 \dots s_i \dots s_h) \\ &= s_1 s_2 \dots f_j(s_i) \dots s_h \\ &\quad i \in [1 \dots h],\ j \in [1 \dots |s_i|] \end{aligned} \tag{1}$$

We can better explain the routing table entries with an example. Consider the shaded AS in Fig. 1 with pattern $\bar{r}c - \bar{e} - k\bar{t}$. This AS will have a total of 7 $(= (2^2 - 1) + (2^1 - 1) + (2^2 - 1))$ routing links to ASs marked with numbers 1 to 7 in the figure. The first three routing links are computed by taking the character presence combination over the characters in $S_1 = \{r, c\}$, which gives us $rc - \bar{e} - k\bar{t}$, $r\bar{c} - \bar{e} - k\bar{t}$ and $\bar{r}\bar{c} - \bar{e} - k\bar{t}$. Note that for the first and the third links, the tree has not been fully expanded to level 3, so the links will be pointing to nodes $rc - \bar{e}$ and $\bar{r}\bar{c} - \bar{e}$, respectively. For computing link 4, characters corresponding to partitioning sets S_1 and S_3 shall remain unchanged, while the character(s) in S_2 will be complemented and so on. Slightly different situation can arise if an AS has a shorter pattern than other ASs, e.g., the AS with pattern $rc - \bar{e}$; its first routing entry would be $r\bar{c} - \bar{e}$, which is an internal node. Here any AS having a pattern starting with $r\bar{c} - \bar{e}$ (i.e., any of $r\bar{c} - \bar{e} - kt$, $r\bar{c} - \bar{e} - k\bar{t}$, $r\bar{c} - \bar{e} - \bar{k}t$, $r\bar{c} - \bar{e} - \bar{k}\bar{t}$) can be considered as the routing link for $rc - \bar{e}$.

Message Forwarding: We can define a simple message forwarding mechanism based on the above described routing table. A lookup string is converted to a set of characters corresponding to the partitioning set, S_i. The lookup request will be forwarded to the AS responsible for the queried characters in a multi-hop path. This path is obtained by gradually transforming the prefix of the current AS to the lookup pattern. We define a *prefix* of a *pattern* as a leftmost sub-pattern of any length, e.g., $s_1 s_2 \dots s_p$, where $p \leq h$. Following the previous example in Fig. 1, suppose the dark shaded node is looking for the AS responsible for string "rectangle", which is mapped to an AS with prefix $rc - e - \bar{k}t$. At the first step, node $\bar{r}c - \bar{e} - k\bar{t}$ will forward the query to AS with the prefix $rc - \bar{e}$ using routing link 1. The 2nd routing link of $rc - \bar{e}$ will have the prefix of any AS, say $rc - e - kt$, under AS $rc - e$. Thus the query will be forwarded to $rc - e - kt$, which will finally forward the query to $rc - e - \bar{k}t$.

It is worth noting that the partitioning tree, as in the example of Fig. 1, does not exist in terms of physical links, because not all of the internal nodes in the tree are assigned to an AS. Rather, the tree exists logically at each AS as prefix strings to the root. The overlay network is composed of the routing links (dashed lines in Fig. 1) between the indexing nodes.

C. Join Protocol

To join the network, a new AS, say X, has to know an existing AS in the system, say M, as a seed AS. Initially, X will forward its queries to M, and M will route the queries using the routing protocol discussed in Section III-B. X will query M for the neighbor, say M_1, with shortest prefix. Next, X will query M_1 for the neighbor with shortest prefix. In this way X will crawl the network and find a local minimum, i.e., a node with shorter prefix than all of its neighbors. In the case of a tie, X will choose the node with higher load, i.e., storing higher number of index records. Once the local minimum, say Y, is

found, X will request Y to increase its prefix by one step. If prefix of Y is $s_1 s_2 \ldots s_p$ and Y has $2^{|S_p|}$ siblings in level p then Y will increase its prefix to $s_1 s_2 \ldots s_{p+1}$ and X will become a new sibling of Y, otherwise X will become a sibling of Y at level p. Accordingly, X has to populate its routing table using the routing information at Y. It should be noted that the neighbors of X will be within two overlay hops of Y. As a result, the joining overhead is expected to be low.

IV. From Overlay to Underlay

In order to implement αRoute for ICN routing, we have to map the nodes in the αRoute overlay graph to the Internet AS topology. In this section we first explain the impact of Internet topology on the mapping process (Section IV-A), then we present the mapping algorithm (Section IV-B) followed by the lookup (Section IV-C) and caching (Section IV-D) mechanisms in the underlay network.

A. Topology Considerations for Mapping

The inter-domain AS network is based on the Border Gateway Protocol (BGP), while each AS controls its intra-domain routing protocol independently. Hence, it will be inappropriate to use αRoute for both inter- and intra-domain routing in the future information centric Internet. Instead, we assume that ASs will collaborate using αRoute for inter-domain routing, while for intra-domain routing, an AS may extend its αRoute prefix or it may use a separate intra-domain routing protocol. This will allow an AS to have complete control over its own domain, while collaborating with the other ASs.

Node degree distribution at the overlay (αRoute) and underlay (AS-topology) graphs has profound impact on the mapping process. According to Fig. 1, each indexing node of a uniformly grown partitioning tree should have similar number of routing links. In other words, the overlay graph is a nearly regular graph. On the contrary, it has been reported in [10] that the node degree distribution in the AS-topology exhibits a power law relationship with the number of ASs. We exploit this dissimilarity in node degree distribution during the mapping process. Recent studies [11] on Internet topology revealed that a small number (around 12 to 16) of high-degree ASs form an almost completely connected core. The rest of the ASs have multiple physical links to the core, which results into many triangles in the AS graph. It is also reported [12] that the inter-connect graph between the non-core ASs is sparse. For the mapping process, we treat the core ASs as Tier-1 AS, while the ASs directly connected to at least one Tier-1 AS are treated as Tier-2 ASs and so on. Hence, a Tier-2 AS, directly connected to multiple Tier-1 ASs, can route a lookup request to a core AS with an appropriate prefix, or it may use its peering links with other Tier-2 or lower tier ASs. This process recurs for the lower tier ASs as well.

Fig. 2 depicts a conceptual overview of αRoute prefix distribution over the ASs. To exploit the heterogeneity in inter-AS connectivity, we assign short prefixes to the highly connected top tier ASs. A lower tier AS, on the other hand, extends a prefix of an upper tier AS. In contrast to the partitioning tree introduced in Fig. 1, selected logical nodes (partitions) at different levels are assigned to highly connected upper tier ASs. In addition to having the regular αRoute links (as presented by dashed arrows in Fig. 2), an upper tier AS will have physical links to the lower tier ASs that extend its prefix. Now, a lower tier AS will probably have more than one upper tier AS as its neighbor and it will be offered more than one name prefix. Our routing mechanism does not depend on the selection of a particular prefix in this case. The lower tier AS can choose a prefix from an upper-level AS based on some business policy (e.g., random or cheapest). In our implementation, we give preference to an upper-level AS having higher degree (connected to more ASs), but any selection policy will work with αRoute.

Algorithm 2 PERFORMNEIGHBOURMAPPING(ξ, p)

Require: Neighbor set ξ, and own prefix ρ.
Ensure: Neighbors in ξ are mapped to an extension of ρ

1: $S_{patterns} \leftarrow$ set of all unmapped patterns starting with ρ
2: **while** There are unmapped neighbours **do**
3: $nbr \leftarrow$ neighbor with highest number of mapped neighbors from ξ
4: $nbr.pattern \leftarrow$ select a pattern from $S_{patterns}$ that minimizes the hamming distance between nbr and its neighbors
5: Remove $nbr.pattern$ from $S_{patterns}$
6: Mark nbr as mapped
7: **end while**

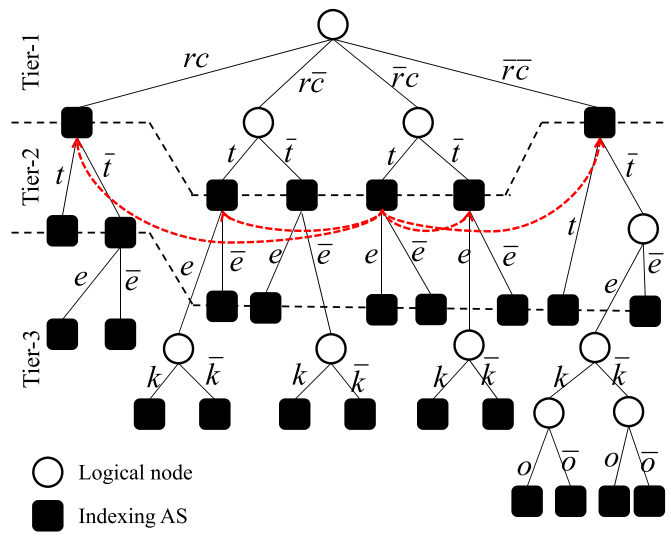


Fig. 2. Mapping a partitioning tree to AS-topology.

B. Mapping Algorithm

The mapping procedure is initiated by a centralized entity referred to as the Name Assignment Authority (NAA). We envision this entity to be like the Internet Corporation for Assigned Names and Numbers (ICANN) that manages the Top-Level Domains (TLDs) in DNS. The NAA chooses prefixes based on the pre-computed partitioning sets and assigns them to the Tier-1 ASs. The prefixes are selected in such a way that the expected name resolution related processing load at each Tier-1 AS remains proportional to its capacity.

It is worth mentioning that the involvement of NAA in αRoute will be much less intensive than that of ICANN in today's Internet for two reasons. First, NAA will assign namespaces to the Tier-1 ASs only. For the rest of the cases, an upper level AS will partition its own namespace among the lower level ASs based on their peering relations without any involvement from NAA. Second, compared to the number of TLDs maintained by ICANN, the number of Tier-1 ASs is much smaller. Thus, the involvement of NAA in the name assignment process will not be a bottleneck. In the next step, each Tier-1 AS executes Algorithm 2 to assign prefixes to Tier-2 ASs. Each AS periodically exchanges messages with its neighbors to collect information required for executing Algorithm 2. Each Tier-1 AS extends its own prefix to generate a set of patterns $S_{patterns}$ that are not yet mapped and starts with the same pattern as its own, e.g., if a Tier-1 AS is assigned prefix $r\bar{c}$ then its $S_{patterns}$ set contains all unmapped patterns starting with $r\bar{c}$. Next, the AS finds a neighbor (nbr) that has the highest number of mapped neighbors. nbr is then assigned a prefix in such a way that its distance (in the hamming space) from all its neighbors is minimized and the process goes on until all neighbors are mapped. Here, we choose to minimize hamming distance as it ensures a mapping that closely resembles the DHT link structure. This mechanism produces a mapping between the prefix space and AS network that will enable αRoute to achieve significantly better performance. After the Tier-1 ASs have executed this mapping process, the already mapped Tier-2 ASs map their neighbors using the same Algorithm. The process goes on for the ASs down the hierarchy in a similar manner until each AS is mapped to a prefix.

In terms of mapping a logical link to the underlay network, this mapping strategy can produce three scenarios: *equal*, *compression*, and *expansion*. In most of the cases, a logical link will be mapped to a physical link, resulting into an *equal* or one-to-one mapping. Recall that, if two logical nodes in the overlay space have more than one mismatch (hamming distance) between their prefixes, this results in a multi-hop path between them in the overlay routing space. Therefore, when two physical neighbors have more than one mismatch in their assigned prefixes, a logical path between them in the overlay space is mapped with the physical link between them. In this case, traversing a physical link makes a jump in the overlay space while routing. This will essentially result into a *compression* of an overlay path into a physical link. Finally for a few cases, adjacent overlay nodes will be more than one hop away in the underlay, which will degrade the mapping performance due to the *expansion* of a logical link in the overlay graph to a physical path in the underlay AS-topology. In the experimental results section, we will provide quantitative measures for these three cases and their impact on routing performance.

C. Content Lookup

In αRoute, the content lookup process involves three steps: (i) the content name is converted to a pattern depending on the presence or absence of characters in the partitioning string, (ii) the AS indexing the particular pattern in looked up using the αRoute network, and (iii) the indexing then forwards the lookup request to the actual location where the content is stored. In Fig. 3, we demonstrate the content lookup process with an example, which is based on the partitioning tree in

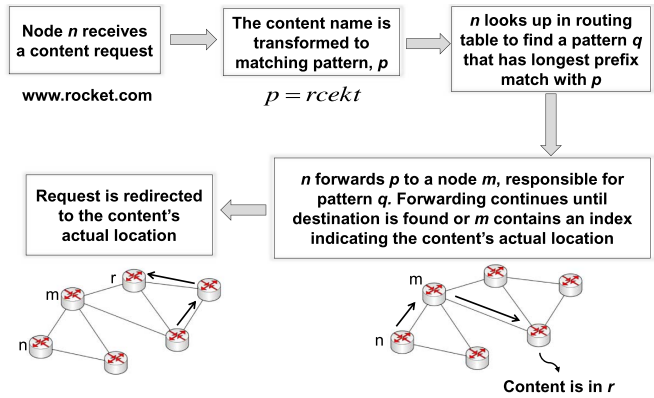


Fig. 3. Content lookup.

Fig. 1. Node n receives a lookup request for the content with name `www.rocket.com` according to the partitioning sets $\{S_i\}$. It converts the name to the pattern p = `rcekt`, then it looks up its routing table and finds pattern q with the longest prefix match with p. Routing is performed using the αRoute network, and it finds node m containing the index for content `www.rocket.com`. The index identifies node r as the actual location for the content. m then forwards the lookup request to r, which returns the content back to n.

Each AS has to maintain a routing table (as explained in Section IV-A) for routing messages to an AS responsible for any given pattern. For each logical routing link L_k (corresponding to the dashed lines in Fig. 2), the routing table will contain an entry like $\langle L_k, I_k, h_k \rangle$. Here, I_k is the inter-AS link that should be used for routing to the AS responsible for pattern L_k, and h_k is the number of ASs to be traversed for reaching L_k. With a good mapping algorithm, h_k will be 1 for most of the cases. In addition to the logical links, an AS will maintain separate routing entries like $\langle P_k, I_k, 1 \rangle$ for each physical neighbor. Here, P_k is the pattern of the neighbor AS reachable through the inter-AS link I_k.

Similar to BGP, αRoute supports policy-based routing. αRoute can be augmented with different policies during the route selection process. In the current implementation we adhere to the following policy. If a lookup request can be resolved using a peering link (usually free of cost) we route using that link. Otherwise, the request has to be forwarded to a provider AS, which usually incurs cost to the requesting AS.

D. Indexing and Caching

Restriction on index placement is a major disadvantage for any DHT approach. To enable efficient content lookup, we have to place a content's index at specific network location. In addition, it introduces a two-step routing: first, route to an index and then route to the content. We can mitigate both of these problems (i.e., index placement freedom and two-step lookup) by intelligently caching indexes and contents. Such caching policies will reduce expensive inter-AS traffic.

Index Caching: ASs may not agree to store any content's index for several reasons, including legal implications and high query traffic volume for a popular content. If the content is illegal or access restricted then this behavior of an AS is appropriate. But, for a popular content, such behavior can decrease the content's reachability. To minimize the volume of lookup traffic for a popular content, each AS can cache the indexes returned

by the outgoing lookup requests for resolving future lookup requests. This index caching strategy will effectively reduce the query traffic volume at the AS indexing a popular content. In addition, an AS can benefit from such index caching. The more index an AS has, the higher number of requests from its own users can be served without forwarding the requests to other ASs. Besides serving its own users, an AS with a large index cache can serve other ASs as well. Although the inter-AS relationships are extremely complex, in general the AS getting more incoming traffic has an upper-hand. Moreover, this index caching strategy will effectively reduce the query traffic volume at the AS indexing a popular content.

Index Replication: The indexing mechanism in αRoute can be easily modified to replicate each index to multiple ASs, and the routing algorithm is able to find at least one of the indices. Index replication can partly tackle misbehaving ASs (e.g., if an AS does not want to serve an index), although at the expense of increased index size per AS, and a corresponding increase in the index synchronization overhead.

Content Caching: As previously reported [7], [13]–[15], content popularity in the Internet and hence lookup rate follows the Power law distribution. We can use this property to improve response time by caching popular contents at the AS storing the content's index. This will allow us to access a content in one DHT lookup. However, a number of challenges need to be addressed before in-network caching can be realized with αRoute. First, a content owner may not allow the indexing AS (or other ASs) to cache and serve its content due to financial and legal reasons. We can overcome this issue by enforcing the policy that only a content owner can update the list of content locations stored at the index node for that content. Second challenge in realizing in-network caching would be determining content popularity. A global popularity model would require the ASs to coordinate among themselves to rank the contents. A number of coordination models are discussed in [16], [17] for determining global content popularity for in-network caching in ICN. However, coordinated caching at the AS level can become very expensive because of the increase in inter-AS traffic. Therefore, a uncoordinated caching strategy that considers content popularity local to an AS is more preferred for αRoute. For instance, we can strike a balance between inter- and intra-AS traffic exchange for serving local demand by adapting existing uncoordinated caching techniques from the literature such as *Myopic caching* [16]. Finally, another challenge in in-network caching would be to ensure that the caches do not create any hot spot in the ASs. To avoid such hot spots, we can probabilistically cache a content along its distribution path, similar to the proposal in [18]. Once a content is cached based on some popularity model, its index can be updated by adding a link to the cached copy. A user can lookup the indexing AS to find a list of ASs caching the desired content and access the content from a nearby AS.

For a large content, the index lookup overhead will be insignificant compared to the bandwidth that we can save by serving the requester from a nearby replica of the content.

V. Analysis

A. Bounds on αRoute

Routing table size (RT_{size}) and the number of hops required to reach a target node in the overlay from any given node are important performance measures for αRoute. Here we define loose upper bounds for these quantities. Consider a network of N routing nodes. Assume that there are ℓ partitioning sets in the overlay graph, and the minimum and maximum sizes for the partitioning sets are b_1 and b_2, respectively. We can compute an upper-bound on RT_{size} as follows:

$$RT_{size} \leq \sum_{i=1}^{\ell} \left(2^{|S_i|} - 1\right) \leq \ell(2^{b_2} - 1)$$

Now we find the bound for hop-distance between any pair of nodes in the overlay. We can observe that at each routing hop the search space is reduced by $1/|S_i|$ and the route converges when there is only one node in the search-space. Hence, we can bound the routing hops, say g, as follows:

$$\frac{N}{\prod_{i=1}^{g} 2^{|S_i|}} = 1$$
$$\Rightarrow \frac{1}{b_2}\log_2 N \leq g \leq \frac{1}{b_1}\log_2 N$$

We can trade off RT_{size} for hop distance; increasing the size of partitioning sets $|S_i|$ will result in a shorter, fatter tree with shorter overlay paths but increased RT_{size}.

B. Bounds on Mapping

The effectiveness of our mapping strategy is inversely proportional to the number of path expansions, i.e., the cases where an overlay edge is mapped to a multi-hop underlay path. We use *expansion ratio* (η) to measure this quantity. η can be defined as the expected value of the ratio of the length of mapped underlay path and the length of corresponding overlay path. The lower the expansion ratio is, the higher the effectiveness of a mapping will be. Suppose a mapping algorithm $\mathcal{M}$ maps an overlay path P_{ov} to an underlay path $\mathcal{M}(P_{ov})$, where $\mathcal{M}(P_{ov})$ is the concatenation of the underlay paths corresponding to the edges along P_{ov}. Let $|P|$ denote the length of path P. Hence, we can define $\eta_{\mathcal{M}}$ as follows:

$$\eta_{\mathcal{M}} = E\left[\frac{|\mathcal{M}(P_{ov})|}{|P_{ov}|}\right] \tag{2}$$

Lemma 1: The expansion ratio $(\eta_{\mathcal{M}})$ of a mapping strategy, say $\mathcal{M}$, that tries to preserve the adjacency relations in overlay graph will not be greater than the expansion ratio $(\eta_{\mathcal{R}})$ of a mapping strategy, say $\mathcal{R}$, that maps overlay nodes to randomly chosen underlay nodes without considering the adjacency relations in overlay graph.

Proof: Suppose, $\mathcal{R}$ maps the overlay nodes x, y and z to underlay nodes x', y' and z', respectively, such that the followings hold: (a) x and y are neighbors, (b) x' and z' are neighbors, and (c) none of the adjacency relations are preserved for y and z. If $\mathcal{M}$ takes the adjacency relation into consideration, and remaps y to z' and z to y', then path expansion will decrease for one edge (as $x \rightarrow y$ mapped to $x' \rightarrow z'$) and $\eta_{\mathcal{M}}$ will become smaller than $\eta_{\mathcal{R}}$. By repeating this process, mapping $\mathcal{M}$ can reduce $\eta_{\mathcal{M}}$ for every instance of x, y and z. If no such x, y and z exist then $\eta_{\mathcal{M}}$ will remain equal to $\eta_{\mathcal{R}}$. ■

From Lemma 1, we can say that $\eta_{\mathcal{R}}$ will serve as an upper bound for the expansion ratio $(\eta_{\mathcal{M}})$ of our mapping algorithm,

since our mapping is based on overlay adjacency relations. Evidently, for mapping $\mathcal{R}$, the mapped path length $|\mathcal{R}(P_{ov})|$ does not depend on the overlay path length $|P_{ov}|$. Hence, we can write:

$$\eta_{\mathcal{R}} = \frac{E\left[|\mathcal{R}(P_{ov})|\right]}{E\left[|P_{ov}|\right]} \tag{3}$$

Now, we compute $E[|\mathcal{R}(P_{ov})|]$, for the random mapping strategy $\mathcal{R}$. Since, $\mathcal{R}$ does not try to preserve the adjacency relations in overlay, we can say that two adjacent nodes in the overlay will be mapped to two arbitrary nodes in the underlay and their distance will be the expected underlay path length, $E[|P_{un}|]$. Thus, we can compute the expected underlay path length for any overlay path as follows:

$$E\left[|\mathcal{R}(P_{ov})|\right] = E\left[|P_{ov}|\right] \times E\left[|P_{un}|\right] \tag{4}$$

Hence, we can compute $\eta_{\mathcal{R}}$ for mapping $\mathcal{R}$ by combining (3) and (4) as: $\eta_{\mathcal{R}} = E[|P_{un}|]$.

Now, to compute $E[|P_{un}|]$, we use the fact that the underlay, i.e., the Internet AS-topology, is a power law graph. In a power law graph, the number of vertices of degree d is proportional to $1/d^{\beta}$ for some exponent β. It has been reported in [19] that the range of β is $2 < \beta < 3$ for the Internet AS-topology, and for this range of β the average distance between two nodes in a power law graph, and hence $\eta_{\mathcal{R}}$ (an upper bound for $\eta_{\mathcal{M}}$), can be derived as follows [19]:

$$\eta_{\mathcal{R}} = E\left[|P_{un}|\right] = (2+\epsilon)\frac{\log\log N_{un}}{\log\left(\frac{1}{(\beta-2)}\right)} \tag{5}$$

where $\epsilon \ll 1$.

We can also define a loose lower bound for the expansion ratio. Let us consider some mapping strategy $\mathcal{A}$ and a routing scheme $\mathcal{T}$ with the following property. If $\mathcal{A}$ maps overlay nodes x and y to some nodes x' and y', respectively in the underlay, then $\mathcal{T}$ can route messages between x' and y' in shortest path in the underlay network, regardless of the path length and the intermediate nodes between x and y in overlay. Thus we can write

$$\eta_{\mathcal{A}} = E\left[\frac{|\mathcal{A}(P_{ov})|}{|P_{ov}|}\right] = \frac{E\left[|\mathcal{A}(P_{ov})|\right]}{E\left[|P_{ov}|\right]} + Cov\left(|\mathcal{A}(P_{ov})|, \frac{1}{|P_{ov}|}\right) \tag{6}$$

If $\mathcal{A}$ is an adjacency preserving mapping then the co-variance between $|\mathcal{A}(P_{ov})|$ and $|P_{ov}|^{-1}$ will be positive, otherwise for a random mapping it will be zero. In addition, the expected underlay shortest path $(E[|\mathcal{A}(P_{ov})|])$ between the mapped endpoints of P_{ov} is basically the expected underlay distance $E[|P_{un}|]$, since the distances between any pair of nodes in the underlay are calculated along a shortest path. Thus, we get from (6) the following:

$$\eta_{\mathcal{A}} \leq \frac{E\left[|P_{un}|\right]}{E\left[|P_{ov}|\right]} \tag{7}$$

We can get $E[|P_{un}|]$ from (5). For computing $E[|P_{ov}|]$ we exploit the fact that the overlay network is a nearly regular graph. We can calculate the expected distance between any two overlay nodes as follows:

$$E\left[|P_{ov}|\right] = \sum_{|P_{ov}|=1}^{K} Pr\left(|P_{ov}|\right) \times |P_{ov}| \tag{8}$$

Here, $Pr(|P_{ov}|)$ is the probability that an overlay path is of length $|P_{ov}|$, and K is the overlay graph diameter. According to the Moore bound [20] (p. 180), the diameter K of a d-regular graph $(d > 2)$ with N_{ov} nodes is as follows:

$$K = \frac{\log \frac{N_{ov}(d-2)+2}{d}}{\log(d-1)} \tag{9}$$

Since, the overlay graph is a regular graph, every node has d paths of length 1 and there are N_{ov} such nodes. So, the number of paths of length 1 is $N_{ov}d/2$. Again, from every node, we can reach $(d-1)$ nodes, and from those $(d-1)$ nodes, we can reach another $(d-1)$ nodes without considering the overlaps. So, the number of paths of length 2 is $N_{ov}d(d-1)/2$. Similarly, the number of paths of length $|P_{ov}|$ is $N_{ov}d(d-1)^{(|P_{ov}|-1)}/2$, for $1 \leq |P_{ov}| \leq K$. So,

$$\begin{aligned} Pr\left(|P_{ov}|\right) &= \frac{\frac{N_{ov}d(d-1)^{(|P_{ov}|-1)}}{2}}{\sum_{|P_{ov}|=1}^{K} \frac{N_{ov}d(d-1)^{(|P_{ov}|-1)}}{2}} \\ &= \frac{(d-2)(d-1)^{(|P_{ov}|-1)}}{(d-1)^{K}-1} \end{aligned} \tag{10}$$

From (8) and (10) we get,

$$E\left[|P_{ov}|\right] = \frac{(d-1)^{K}\left(K(d-2)-1\right)+1}{(d-2)\left((d-1)^{K}-1\right)} \tag{11}$$

In Section VI, we compare the expansion ratio for our mapping strategy against the above computed bounds.

VI. Experimental Results

In this section we report the results obtained by simulating αRoute, DONA [21] and Content Centric Networking (CCN) [5] using realistic Internet (AS level) topologies and a large name dataset. We have developed a discrete event simulator to evaluate these approaches. The simulator first advertises the names from our name data set to build the forwarding table for αRoute, DONA and CCN, respectively. Then we process a stream of name look up events drawn from some well studied content lookup distributions [6], [7]. We measure several metrics to evaluate, compare, and contrast the performance of αRoute with DONA and CCN.

DONA proposes a hierarchical name resolution infrastructure that defines a new entity called Resolution Handlers (RH). Each logical domain (e.g., Autonomous system) has its own RH that stores name-based routing information for that domain. The existing parent-child relationship between the domains (ASs) results in a logical tree between the RHs. The root RH of this tree maintains routing information for all content in the network. Name resolution is performed using the route-by-name paradigm that is deployed above the IP layer. Network operators can define global and local routing policies similar to BGP.

CCN on the other hand, proposes to completely redesign the Internet by replacing IP with content chunks as a universal component of transport. CCN utilizes two key messages to perform routing: Interest and Data. These messages are routed using a route-by-name paradigm. A client, looking for a content, broadcasts an Interest message over all available connections. Any CCN node having the original content or a replica or a cached

TABLE I
DATASET CHARACTERISTICS

Dataset	Mean degree	Median degree	95^{th} percentile degree	Powerlaw exponent
DIMES	7	2	16	2.06
S40K	8	2	12	1.94
S55K	9	2	14	1.92

copy of that content, can respond to the Interest message with a corresponding Data message. A CCN node maintains the following tables: Forwarding Information Base (FIB), Pending Interest Table (PIT), and Content Store (CS). The FIB is used to store probable source(s) for a content. The PIT stores the return path information for each unresolved Interest message. This information is used to forward a Data message back to the requester. CS is used for caching content.

We compare the number of underlay messages, lookup latency and message processing load for αRoute, DONA and CCN. We measure the index storage load for αRoute to show the uniformity of load distribution. In case of DONA, storage load is highly skewed as the root node stores all the names. In case of CCN, storage load is fixed as we limit the FIB size to $10K$ to set an optimistic operating point for CCN. Currently prefix aggregation enables $\sim 10^5$ BGP routes to serve $\sim 10^9$ IPv4 addresses, achieving a scaling factor of $\sim 10^4$ [1]. However, the *ad hoc* nature of content names allows for a far lesser degree of aggregation. One extreme would be to have the same order of FIB entries as the name data set size. But the hierarchical nature of names in CCN allows some level of aggregation. Therefore, we set an optimistic limit of $\sim$10 scaling factor between the name data set size and the FIB table size. We also evaluate the performance of overlay to underlay mapping algorithm used in αRoute by computing the expansion ratio (η) as described in Section V-B.

A. Dataset

a) Inter-domain topology: We use three different topologies to evaluate αRoute: AS level Internet topology collected by the DIMES project [22] and two synthetic AS topologies generated using Inet (v 3.0) Internet topology generator [23]. The DIMES dataset contains topology data for $26K$ ASs, while the synthetic dataset contains topology data for $40K$ $(\mathrm{S}40K)$ and $55K$ $(\mathrm{S}55K)$ ASs. In case of CCN, we perform the simulation after the Forward Information Base (FIB) at each router reaches steady state. However, this scheme prevents us from simulating CCN on $\mathrm{S}40K$ and $\mathrm{S}55K$ as it requires a significant amount of time to reach steady state due to large number of routers for these topologies. For DONA, we assume each AS owns a single RH. Each name in DONA is propagated from the originating RH to the root RH, and each RH on the path stores a link for that name. This feature makes it infeasible to simulate DONA for $\mathrm{S}40K$ and $\mathrm{S}55K$ topologies, as it requires significantly large amount of memory and time just to build the initial RH tree. Some characteristics of these datasets are given in Table I

b) Name: αRoute can work with any arbitrary alphabet $\mathcal{A}$. However, for the evaluation we restricted $\mathcal{A}$ to the 36 alpha-numeric characters in English, i.e., a... z and 0... 9. While generating the partitioning sets $(\{S_i\})$, we investigated all possibilities up to the length $\zeta = 4$. The published names for the evaluation purpose have been collected from three different sources: (a) synthetically generated $50K$ hierarchical names using the top domains obtained from Alexa,[1] (b) $\sim$ $130K$ hierarchical names from dmoz[2] open directory project, and (c) $\sim 810K$ names from URL Blacklist.[3]

We have generated a sufficiently large sequence of content lookup requests using a Zipf-Mandelbrot random variate generator with parameters α and q set to 1.04 and 100, respectively [6], [7].

B. Results

1) Message Count: Message count represents the number of underlay (physical network) messages for each content lookup. This metric measures how efficiently a routing algorithm can utilize the underlying network. A lower message count implies higher bandwidth utilization and vice versa. Fig. 4(a) reports the Cumulative Distribution Function (CDF) of message counts for αRoute, DONA and CCN. αRoute is run on three network topologies: DIMES, $\mathrm{S}40K$ and $\mathrm{S}55K$. DONA and CCN are run on the DIMES topology (Section VI-A). The X-axis shows the number of messages and the Y-axis shows the percentage of content lookups requiring at-most that many messages, e.g., for the DIMES topology, αRoute can complete 90% content lookups with at most 8 messages. αRoute shows similar performance for all three topologies.

DONA and CCN perform significantly worse than αRoute, with CCN having the worst performance. DONA requires up to 50 messages for completing 80% of content lookups. αRoute requires at most 6 messages for the same. αRoute performs better than DONA as lookup messages in DONA can go up to the root RH if the source and destination RHs are in different geographic locations. However, in the case of αRoute this problem is mitigated by caching links to the destination ASs after the first lookup. This feature reduces communication overhead for subsequent lookups.

In case of CCN, even with 200 messages only 60% content lookups can be completed. CCN uses flooding to discover new content, which is the reason behind its degraded performance. Another point to be noted is that CCN cannot provide 100% content lookup guarantee because of the bounded FIB size. In our simulations, CCN was able to discover 99.91% contents at most. However, CCN will be able to discover all existing content if we could provide unlimited FIB size.

2) Lookup Latency: The second metric we measure is the content lookup latency, i.e., the time required to discover a content in the network. Fig. 4(b) reports the CDFs of lookup latency for αRoute, CCN, and DONA on the DIMES topology.

[1]Alexa, the web information company (www.alexa.com)

[2]ODP—Open Directory Project (www.dmoz.org)

[3]urlblacklist.com

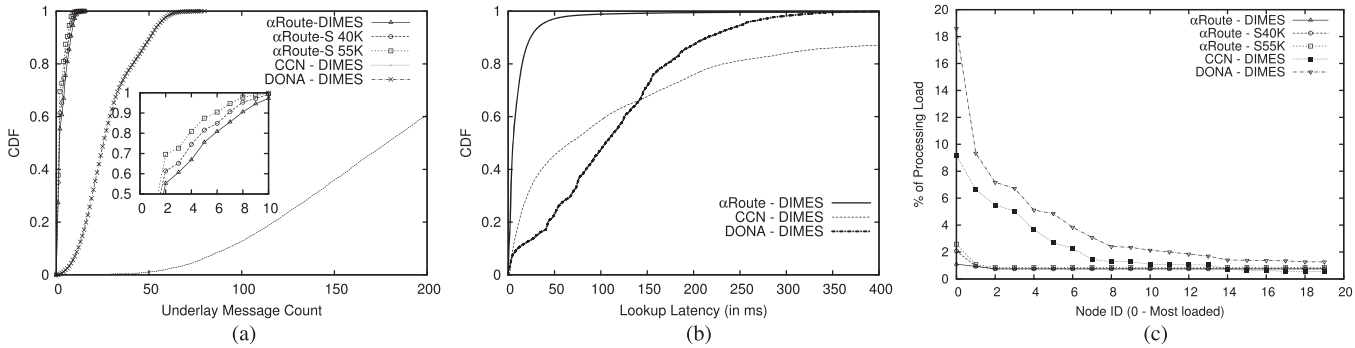


Fig. 4. Performance comparison between αRoute, CCN and DONA. (a) Message count distribution. (b) Latency distribution. (c) Message load distribution.

We cannot provide latency data for αRoute on the S40K and S55K datasets as the Inet Internet topology generator does not provide latency measurements. Here, we first note that for 99% of the cases lookup latency in αRoute is less than 100 ms. On the other hand, DONA and CCN can resolve only 40% and 58% lookups within 100 ms, respectively. The second observation from Fig. 4(b) is that the CDF for αRoute rises much faster compared to CCN and DONA. Both CCN and DONA require significantly longer lookup times to provide comparable performance to αRoute. For example, CCN can resolve 99% lookup requests in 7857 ms, which is approximately 80 times higher than that of αRoute.

αRoute performs much better than DONA and CCN due to the neighbor selection mechanism (outlined in Algorithm 2) and link caching. αRoute always chooses a neighbor that has the highest number of mapped neighbors ensuring that similar patterns get mapped to close-by ASs in the network. This feature eventually creates a clustering of the underlying ASs based on the hamming distance between their character presence patterns. Now, whenever an AS X caches a link to another AS (with a pattern of high edit distance) Y, every close-by AS to X can lookup Y much faster through X.

It is worth mentioning that, in terms of lookup latency, αRoute also outperforms DMap [6], which reports 86.1 ms as the 95th percentile of lookup latency on the DIMES topology. In case of αRoute, the 95th percentile lookup latency is $\sim$70 ms.

3) Message Load: The third metric is the distribution of message load for name lookup. Message load of an AS x is calculated by dividing the number of messages processed at x by the product of x's degree and the total number of messages propagated in the network. We calculate message load using following equation:

$$x\text{'s message load} = \frac{\text{messages processed by } x}{x\text{'s degree} \times \text{total message count}} \quad (12)$$

Here, we are assuming that an AS's message processing capacity scales linearly with its degree. Therefore, by dividing the total number of processed messages at x by its degree we want to observe whether the message processing load was distributed according to an AS's connectivity. Then we divide the result by the total number of messages in order to normalize the metric.

Fig. 4(c) shows the percentage of content lookup messages processed by the top-20 most loaded ASs. An ideal routing protocol will produce a straight line parallel to the X-axis in this plot yielding perfect load distribution. αRoute comes very close to the ideal scenario in all three topologies (DIMES, S40K and S55K) only with the exception of the first node being slightly more loaded than the other ASs.

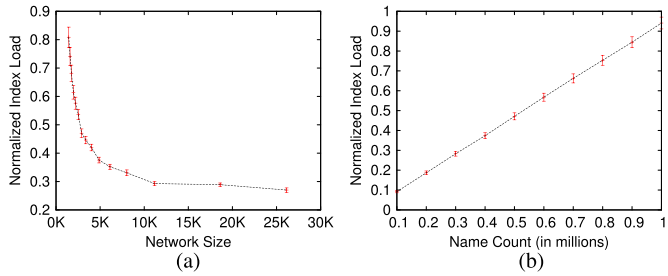


Fig. 5. Distribution of indexing load for αRoute. (a) Impact of network size. (b) Impact of content volume.

DONA and CCN exhibit largely varying message load distribution. DONA requires the ASs to store name records according to their position in the logical tree hierarchy. The root AS stores all name records in the network and the leaf ASs store names that originated only from themselves. This behavior is clearly exhibited in Fig. 4(c) through the highly skewed processing load distribution. CCN also shows a similar pattern as a significant number (almost 10%) of name lookups have to go through the AS with the highest node degree (similar to the root AS in DONA). CCN shows better performance than DONA as CCN routers aggregate lookup requests for the same content from different interfaces and forwards just one request. DONA does not have an equivalent query request aggregation mechanism.

αRoute shows almost perfect load balancing. αRoute caches links to unknown ASs whenever they are part of a name lookup path. This feature allows αRoute to bypass the root (and also other ASs with high node degree) AS for subsequent name lookups and helps to distribute the lookup load in the network.

4) Indexing Load: Fig. 5 shows the distribution of indexing load for αRoute. Fig. 5(a) shows the average (along with 95% confidence interval) number of indexes stored per node relative to its capacity (degree). Here network size is varied from around 1500 to 26K nodes, while 1 million names are published. It is evident from the figure that the average number of stored indexes drops exponentially and the small error bars confirm the uniform distribution of indexing load. In Fig. 5(b), the number of published names is varied while the network size is fixed at 26K nodes. As expected, the index load increases linearly with

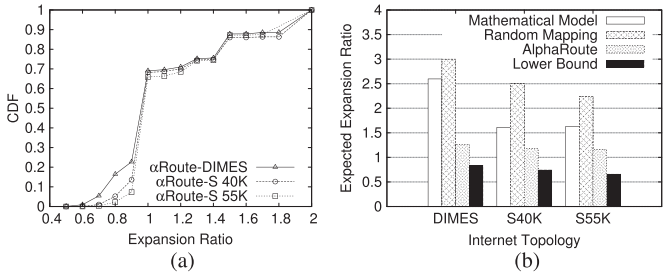


Fig. 6. Analysis of mapping performance. (a) Expansion ratio distribution. (b) Comparison of expansion ratio.

the number of published names and the error bars are very small. This further confirms the uniform index load distribution.

5) Mapping Performance: Fig. 6 analyses the performance of our mapping algorithm described in Section IV. Fig. 6(a) shows the CDF of expansion ratio per content lookup of αRoute on three topologies, namely: DIMES, S40K, and s55K. In 70% cases the expansion ratio is less than 1 corresponding to path compression, as explained in Section IV-B. In the rest of the 30% cases we have expansion ratio greater than 1 corresponding to the path expansions from overlay to underlay. Since path compression is much higher than path expansion, the resulting expected expansion ratio is very close to 1 as reported in Fig. 6(b). In particular, the expansion ratio for αRoute in DIMES, S40K, and S55K topologies are 1.26, 1.09, and 1.11, respectively.

In Fig. 6(b), we plot the expansion ratios in the average case (Section V) $\eta_{\mathcal{R}}$ along with the lower bound $\eta_{\mathcal{A}}$. We also plot the expansion ratios we obtained by simulating αRoute and a random node mapping strategy. In the random mapping approach we map an overlay node to an AS uniformly at random and then add as many random links between AS pairs as there are cached links in αRoute. The second step ensures a fair comparison between random mapping and αRoute.

The expansion ratio for the random mapping is greater than the expected average from the mathematical model, which shows that this is not a trivial problem to solve. In case of αRoute, it is evident from the plot that the expansion ratio of our mapping scheme is much less than the average case and very close to the lower bound. This is mostly due to the fact that αRoute caches links to other ASs, but the mathematical model does not consider caching. We can also see that the expansion ratio for αRoute is close to 1, which means that the adjacency relationships in overlay (DHT) network are well preserved by the proposed mapping algorithm.

VII. Discussion

Although αRoute solves most of the desirable features (as presented in Section II) for inter-domain ICN routing, it has a few limitations. In this section we explain some of these limitations and possible ways to mitigate their effect. In addition, we outline some possibilities for deploying αRoute.

A. Limitations

Proper functionality of αRoute depends on the ASs' willingness to collaborate in indexing and query routing. It will be a problem if an AS refuses to index a content assigned by αRoute. An index replication scheme can be used to reduce the impact of such behavior. In addition, index caching can reduce the number of queries for a popular content from reaching the indexing AS. An AS can benefit from resolving queries and serving the assigned indexes. The more indexes an AS has, the higher number of queries it can serve for its own users and for its peer ASs. This way an AS can reduce outgoing traffic, and can have an upper hand in the bandwidth exchange ratio with its peer ASs. These factors can provide adequate incentive for an AS to serve and cache indexes from other ASs.

Another problem may arise if an AS has less than required performance for query resolution. We can use an explicit protocol for an AS to inform its parent AS in αRoute tree about any performance issue. Alternatively, a parent AS may monitor its child ASs' performance. If a performance problem is experienced, the parent AS can redistribute the indexing load within itself or to other child or peer ASs with lesser load.

The algorithm for constructing the partitions, $\{S_i\}$, is offline. We currently select S_is in such a way that the names in the corpus are uniformly distributed over the leafs of the logical indexing tree. For a fairly large corpus, this offline computation should give nearly uniform distribution of names over the resolution nodes. However, the balance in load distribution may become skewed over time as new content is added, and old content is removed from the network. The impact of such skewness can be diminished by locally redistributing the indexes between adjacent ASs in the logical partitioning tree.

B. Deployment Considerations

There are a number of alternative ways to deploy αRoute in the Internet. Here we briefly discuss some of these alternatives. Inline with the existing ICN proposals, αRoute can be implemented by leveraging Software Defined Networking (SDN) technologies (similar to [24]) or by overlaying on top of the existing IP routing infrastructure (similar to this IETF draft [25]). Both of these alternatives can be realized by implementing αRoute protocol in software switches, which can be deployed on the emerging White Box switching hardware [26]. αRoute routers will replace the border routers for inter-domain routing at the AS level, as explained in Section IV-A. However, an AS will still have the flexibility to choose αRoute or any existing protocols (e.g., RIP or OSPF) for intra-domain routing. In addition, we have to deploy some dedicated servers as the Name Assignment Authority (NAA) to compute the partitioning sets $\{S_i\}$, and for the initial assignment of partitions to Tier-1 ASs.

VIII. Related Work

The last few years have witnessed a significant number of research efforts in the field of ICN. Several of these research works identified content naming and name-based routing as the key research challenges in ICN, and proposed different solutions for these problems. A comprehensive survey on different naming and routing schemes proposed for ICN can be found in [27]. In this section, we briefly present the major research efforts on naming and routing in ICN.

A. Hierarchical Routing in ICN

DONA [21] provides a hierarchical name resolution infrastructure including new network entities named Resolution Handlers (RH). Each logical domain (e.g., Autonomous system) in the network has its own RH, which stores name based routing information of that domain. The parent child relationship among the domains results in a logical tree between the RHs. The root RH of this tree needs to maintain routing information for all content in the network, which severely confines the scalability of this mechanism. NetInf [28] adopts a hierarchical DHT [29] based approach for name based routing. It proposes to have DHTs for name resolution in the network's Points of Presences (PoPs). The PoP level DHTs are further aggregated into higher level DHT to provide name resolution service in a larger domain. The topmost level in the DHT hierarchy in NetInf, called REX, needs to store index for all the content in the network, which results in performance bottleneck and scalability issues. LANES [30] also proposes a multi-layered routing architecture for ICN. The topmost layer, named the *rendezvous layer* is responsible for inter-domain routing. This layer is maintained by a hierarchical DHT. In general, the hierarchical routing techniques suffer from scalability issue, because the upper layer nodes in the hierarchy need to store routing information for all the content in their children's subtrees. On the other hand, αRoute does not create any routing hot spot and thus can scale better than the hierarchical approaches.

B. Gossip Based Routing in ICN

CCN [5], CURLING [31] and TRIAD [32] use gossip based routing protocols, which incur significant management and control overhead. CCN proposes to replace IP address prefixes in BGP routing table with content name prefixes and route content requests by performing longest prefix matching in routing table. The routing table size in CCN grows almost linearly with the number of content. This growth rate is much higher than that of BGP routing table since the number of content is several order of magnitude higher than the IP prefixes. Hence, CCN struggles to scale with an increase in the number of content. The popular OSPF protocol used in intra-domain routing has been extended to work on top of CCN [33] to route content requests based on content names. However, the extended protocol, OSPFN has some limitations. First, the routers and the links are still identified by IP addresses instead of CCN like hierarchical names. Second, OSPFN does not have multi-path routing support and it relies on GRE tunnels to traverse legacy networks. Motivated by these limitations the authors later proposed NLSR [34], a more improved link state routing protocol for name based routing. To overcome OSPFN's limitations, NLSR proposes a naming scheme for the routers and the links in the network, multi-path routing support and propagation of routing updates via Interest and Data packets (defined by CCN) only. However, these two routing protocols still suffer from the scalability issues inherited from CCN. CURLING [31] proposes to remove the DNS and assigns a new entity, “Content Resolution Server (CRS)”, within each ISP to maintain routing information. Content publish and request resolutions are performed using a gossip based hop by hop protocol. This protocol also maintains the business relationships between the ISPs. CURLING also provides a mechanism for publishing and routing content requests in a non-global scope. CONET [35] attempts to solve the routing table scalability issue with CCN by proposing to have a fixed number of rows in the name based routing table. When a router misses routing information for a name based routing request, it resolves the request using a DNS like name-system and updates its name based routing table using some cache replacement policy. Routing in CBCB [36] is based on controlled flooding of attribute-value pairs. Interest for a particular content is issued by creating logical disjunctions of conjunctions of elementary constraints over the values of possible attributes. CBCB includes a broadcast protocol that ensures loop-free paths, and a content based routing protocol that avoids sending content to uninterested receivers by pruning branches of the broadcast tree. The routing table size in a CBCB router is expected to be exponential in the number of attributes. CBCB requires network wide flooding to forward a request with any unknown attributes, which incurs significant network overhead. Distance-based Content Routing (DCR) [37] is a name based routing technique that builds a multi-instantiated destination spanning tree to compute the distance to multiple copies of a content and routes traffic depending only on distance information. DCR improves over the other broadcast based routing techniques by reducing the number of network messages, but has the same data plane scalability issue in terms of routing table size similar to CCN. In contrast to the gossip-based routing approaches, αRoute does not flood the network for content lookup, while requiring smaller routing tables proportional to the logarithm of network size. Thus avoiding the scalability issue as arising in gossip based routing protocols.

C. Hash Based Routing in ICN

A number of other works in the literature propose to hash content names to map to location identifiers and route based on the hashed values. Internet Indirection Infrastructure (i3) [38], LISP-DHT [39] and ROFL [40] are among the early DHT based proposals in this area. i3 [38] was one of the first architectures for routing on non-IP based labels. i3 uses a DHT and routes the labels using an overlay network. i3 provides a more abstract way to express packet delivery operations in applications and does not focus on mapping the overlay DHT to the underlying physical network. LISP-DHT [39], on the other hand, is not a routing protocol, rather it provides a scalable mechanism for locating content location from content names using DHT. The content location is then used by the routing layer for the actual routing. ROFL [40] provides network routing with the help of an inter-domain DHT. However, ROFL makes the assumption that content names are flat and do not carry any semantics. More recently, DMap [6] has proposed a global name resolution service for mobility and efficient content delivery in the Internet. In this architecture, each content is assigned a globally unique identifier (GUID). A content's GUID is hashed to obtain a list of IP addresses. The content's original location is indexed at these IP addresses. These indices are updated when that content is moved to a different location. Every network entity in DMap needs to have a global knowledge of the IP prefix advertisements from all the ASs to successfully locate

the index location of a content. Moreover, the routers in the Internet have to store extra index along with the BGP routing table that they are already maintaining, which imposes a significant storage overhead. Saino *et al.* explores the possibility of using hash-routing schemes to place and locate content in caches within an ISP's network, while optimizing the utilization of the in-network caching space [41]. The authors experimentally evaluate different hash-routing and cache placement strategies and show that some hash-routing technique's performance depend on the underlying network topology, while most of these techniques can achieve a high cache hit rate when used within an ISP's network. αRoute also works by hashing content names. Unlike the above mentioned approaches, αRoute is not an overlay based routing technique where the routing nodes need to store multiple tables for overlay and underlay routing. αRoute maps the hash based overlay network on top of a physical underlay and can be used to locate both the original content as well as a cached copy. Moreover, we do not make any assumption about content name in αRoute unlike some of the previous works such as [40].

IX. Conclusion and Future Work

In this paper we proposed αRoute, a name based routing scheme for ICN. αRoute guarantees content lookup while ensuring efficient bandwidth usage and small routing table size. Routing in αRoute relies on a flexible overlay network topology that can be efficiently mapped to the Internet AS-topology. We also presented mathematical bounds on the routing scalability of αRoute and the operating range of our mapping scheme. Simulation results show that the routing efficiency of αRoute is very close to the lower bound and much better than random mapping. Compared to the existing routing techniques, our approach has a number of advantages. First, routing can be done on names without sacrificing efficiency or completeness. Second, after finding the node responsible for a query name, it is easy to find other names within 1 or 2 edit distance; since the nodes responsible for storing those names will be 1 or 2 overlay hops away from the query target. Third, in contrast to hierarchical routing mechanisms, there is no bottleneck node in the proposed system. We achieve a capacity proportional load distribution by placing the ASs at different levels in the partitioning tree based on capacity. Fourth, compared to other tree-based routing approaches, we conveniently select the size of the partitioning sets ($|S_i|$), to tune the depth of the tree. This allows us to easily decrease routing hops by increasing the number of routing-links, and vice versa. The performance of αRoute can be greatly improved by adopting the caching strategies proposed in Section IV-D. We intend to investigate αRoute's performance in presence of indexing and content caching and experiment in a large scale testbed.

References

[1] "BGP routing table analysis reports," [Online]. Available: http://bgp.potaroo.net/

[2] "We knew the Web was big," 2008 [Online]. Available: http://googleblog.blogspot.com/2008/07/we-knew-webwas-big.html

[3] "Domain counts & Internet statistics," [Online]. Available: http://www.domaintools.com/internet-statistics

[4] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. ACM ICN*, 2011, pp. 44–49.

[5] V. Jacobson *et al.*, "Networking named content," in *Proc. CoNEXT*, 2009, pp. 1–12.

[6] T. Vu *et al.*, "DMap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *Proc. IEEE ICDCS*, 2012, pp. 698–707.

[7] O. Saleh and M. Hefeeda, "Modeling and caching of peer-to-peer traffic," in *Proc. IEEE ICNP*, 2006, pp. 249–258.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1990.

[9] S. Bokhari, "On the mapping problem," *IEEE Trans. Comput.*, vol. C-30, no. 3, pp. 207–214, Mar. 1981.

[10] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," *Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, Aug. 1999.

[11] M. Boguñá, F. Papadopoulos, and D. Krioukov, "Sustaining the internet with hyperbolic mapping," *Nature Commun.*, vol. 1, p. 62, 2010.

[12] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM*, 2002, vol. 2, pp. 618–627.

[13] S. A. Krashakov, A. B. Teslyuk, and L. N. Shchur, "On the universality of rank distributions of website popularity," *Comput. Netw.*, vol. 50, no. 11, pp. 1769–1780, Aug. 2006.

[14] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. ACM IMC*, 2007, pp. 15–28.

[15] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an ip network," in *Proc. ACM IMC*, 2008, pp. 71–84.

[16] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Trans. Netw. Service Manage.*, vol. 10, no. 3, pp. 286–299, Sep. 2013.

[17] Y. Li, H. Xie, Y. Wen, and Z.-L. Zhang, "Coordinating in-network caching in content-centric networks: Model and analysis," in *Proc. IEEE ICDCS*, 2013, pp. 62–72.

[18] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for ICN," in *Proc. ACM ICN*, 2012, pp. 55–60.

[19] F. Chung and L. Lu, "The average distance in a random graph with given expected degrees," *Internet Math.*, vol. 1, no. 1, pp. 91–113, 2004.

[20] C. Godsil, G. Royle, and C. Godsil, *Algebraic Graph Theory*. New York, NY, USA: Springer, 2001, vol. 8.

[21] T. Koponen *et al.*, "A data-oriented (and beyond) network architecture," *Comput. Commun. Rev.*, vol. 37, pp. 181–192, Aug. 2007.

[22] Y. Shavitt and E. Shir, "DIMES: Let the internet measure itself," *Comput. Commun. Rev.*, vol. 35, no. 5, pp. 71–74, 2005.

[23] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," University of Michigan, Tech. Rep. CSE-TR-456-02, 2002.

[24] M. Vahlenkamp, F. Schneider, D. Kutscher, and J. Seedorf, "Enabling information centric networking in IP networks using SDN," in *Proc. IEEE SDN4FNS*, 2013, pp. 1–6.

[25] L. C. Li, X. Xu, J. Wang, and Z. W. Hao, "Information-centric network in an ISP," ICN Research Group, Work in Progress, 2013 [Online]. Available: https://tools.ietf.org/html/draft-li-icnrg-icn-isp-00

[26] "White box switch," [Online]. Available: http://whiteboxswitch.com/

[27] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 44–53, Dec. 2012.

[28] C. Dannewitz *et al.*, "Network of Information (NetInf)—An information-centric networking architecture," *Comput. Commun.*, vol. 36, no. 7, pp. 721–735, Apr. 2013.

[29] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT: A hierarchical name resolution service for information-centric networks," in *Proc. SIGCOMM ICN*, 2011, pp. 7–12.

[30] K. Visala, D. Lagutin, and S. Tarkoma, "LANES: An inter-domain data-oriented routing architecture," in *Proc ReArch*, 2009, pp. 55–60.

[31] W. K. Chai *et al.*, "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *IEEE Commun. Mag.*, vol. 49, no. 3, pp. 112–120, Mar. 2011.

[32] D. Cheriton and M. Gritter, "TRIAD: A scalable deployable NAT-based Internet architecture," Technical Report, Jan. 2000.

[33] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF based routing protocol for named data networking," University of Memphis and University of Arizona, Tech. Rep., 2012.

[34] A. Hoque *et al.*, "NLSR: Named-data link state routing protocol," in *Proc. ACM ICN*, 2013, pp. 15–20.

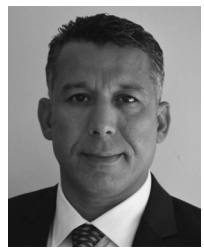
[35] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, "CONET: A content centric inter-networking architecture," in *Proc SIGCOMM ICN*, 2011, pp. 50–55.
[36] A. Carzaniga, M. J. Rutherford, and A. L. Wolf, "A routing scheme for content-based networking," in *Proc. IEEE INFOCOM*, 2004, pp. 918–928.
[37] J. Garcia-Luna-Aceves, "Name-based content routing in information centric networks using distance information," in *Proc. ACM ICN*, 2014, pp. 7–16.
[38] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," *Comput. Commun. Rev.*, vol. 32, no. 4, pp. 73–86, 2002, ACM.
[39] L. Mathy and L. Iannone, "LISP-DHT: Towards a DHT to map identifiers onto locators," in *Proc. ACM CoNEXT*, 2008, Art. no. 61.
[40] M. Caesar *et al.*, "ROFL: Routing on flat labels," *Comput. Commun. Rev.*, vol. 36, no. 4, pp. 363–374, 2006.
[41] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Proc. ACM ICN*, 2013, pp. 27–32.

Reaz Ahmed received his Ph.D. degree in computer science from the University of Waterloo in 2007. He received M.Sc. and BSc. degrees in computer science from Bangladesh University of Engineering and Technology (BUET) in 2002 and 2000, respectively. He received the IEEE Fred W. Ellersick Award in 2008. His research interests include future Internet architectures, information-centric networks, network virtualization, and content sharing peer-to-peer networks with focus on search flexibility, efficiency, and robustness.

Md. Faizul Bari is a Ph.D. candidate at the School of Computer Science, University of Waterloo. He received M.Sc. and B.Sc. degrees in computer science and engineering from Bangladesh University of Engineering and Technology (BUET). His research interest includes future Internet architecture, network virtualization, and cloud computing. He has received the Ontario Graduate Scholarship, President's Graduate Scholarship, and David R. Cheriton Graduate Scholarship at the University of Waterloo.

Shihabur Rahman Chowdhury is a Ph.D. student at the School of Computer Science, University of Waterloo. He received his B.Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET). His research interests include virtualization and softwarization of computer networks. He is a recipient of the Ontario Graduate Scholarship, Presidents Graduate Scholarship, and GoBell Scholarship at the University of Waterloo.

Md. Golam Rabbani is currently working as a Software Development Engineer at Amazon Web Services. He received the Masters of Mathematics and Master of Information Technology degree from the University of Waterloo, Canada, and Monash University, Australia, in 2014 and 2011, respectively. He worked as a System Engineer in a leading telecommunications company in Bangladesh from 2007 to 2009. His research interest include data centers, cloud computing, future Internet architecture, and wireless communications.



Raouf Boutaba received the M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, France, in 1990 and 1994, respectively. He is currently a Professor of computer science at the University of Waterloo, Canada. His research interests include resource and service management in networks and distributed systems. He is the founding Editor-in-Chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (2007–2010) and on the editorial boards of other journals. He received several best paper awards and recognitions including the Premiers Research Excellence Award, the IEEE ComSoc Hal Sobol, Fred W. Ellersick, Joe LociCero, Dan Stokesbury, Salah Aidarous awards, and the IEEE Canada McNaughton Gold Medal. He is a Fellow of the IEEE, the Engineering Institute of Canada, and the Canadian Academy of Engineering.

Bertrand Mathieu is a Senior Researcher at France Telecom, Orange Labs, since 1994. He received a Diploma of Engineering in Toulon, an M.Sc. degree from the University of Marseille, and a Ph.D. degree from the University Pierre and Marie Curie, Paris, France. His research activities are related to dynamic overlay networks, P2P networks, and information centric networking. He has contributed to several national and European projects, and published more than 50 papers in international conferences, journals, and books. He is a member of several conferences Technical Program Committees, and an SEE Senior Member.