αRoute: A Name Based Routing Scheme for Information Centric Networks

Reaz Ahmed*, Md. Faizul Bari*, Shihabur Rahman Chowdhury*, Md. Golam Rabbani*,
Raouf Boutaba*, and Bertrand Mathieu†
*David R. Cheriton School of Computer Science, University of Waterloo
{mfbari | sr2chowdhury | m6rabban | r5ahmed | rboutaba}@uwaterloo.ca
†Orange Labs, Lannion, France
bertrand2.mathieu@orange-ftgroup.com

***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 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 novel name-based routing scheme (αRoute) for ICN that offers efficient bandwidth usage, guaranteed content lookup and scalable routing table size.**

I. Introduction

Information Centric Networking (ICN) has recently received significant attention in the research community. ICN philosophy prioritizes a content ("what") over its location ("where"). To realize this separation of a content from its location, a name based 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.

Today's Internet exists as an interconnection of thousands of Autonomous Systems (ASs) from around the globe. The biggest Internet routing table 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. Google has indexed approximately 10^{12} URLs [2], which would impose 7 orders of magnitude scalability requirement on a routing scheme similar to BGP.

The routing scalability issue in ICN is 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 like BGP 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, the crux of the problem lies in the fact that ICN requires Internet routers to maintain a Brobdingnagian amount of routing state, which does not seem to be possible with existing technology. However, in reality the scalability requirement will be much higher for the following reasons: (i) content names are not as aggregatable as IP addresses, (ii) 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 form of specialized format limits the usability of the system, and finally, (v) supporting content replication, caching, and mobility reduces the degree of route aggregation that can be applied, as multiple routes for the same content need to be maintained in the routing table.

In this paper we address the routing scalability issue for ICN. We propose a name-based overlay routing scheme named αRoute, which is scalable and offers content lookup guarantee (Section II). 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 III). We also provide qualitative comparison of our approach with existing approaches for ICN routing in Section IV. Finally, we conclude and outline future research directions in Section V.

II. αRoute: A Name-based DHT

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 assign 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 for αRoute.

A. Partitioning

Our partitioning policy has three desirable characteristics: first, it places similar names in same partition, second, it provides an upper-bound on the number of partitions and third, it creates non-overlapping partitions. We treat strings as unordered sets of characters; *e.g.*, the string "www.rocket.com" will be treated as { w, r, o, c, k, e, t, c, m }. Now we can classify all the strings in the key space in separate partitions, based on the presence or absence of characters in a string. At first, we create some partitioning sets (S_i) over the 36

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

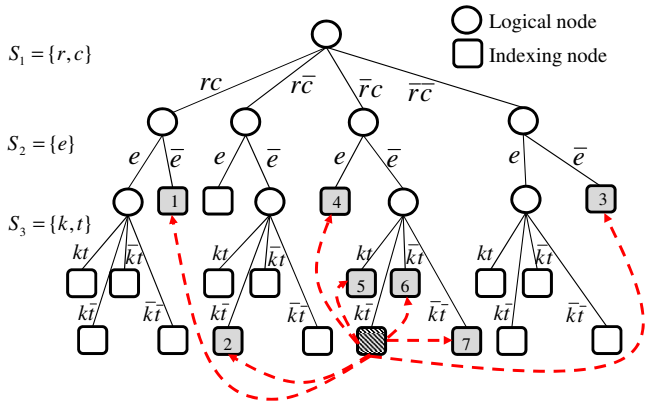

Fig. 1. An example partitioning tree

alphanumeric characters (lower case alphabet and digits) in English based on the expected frequency of each character. Then we create a logical partitioning tree by associating S_i with i-th level in the tree and by expanding each i-th level node using the permutations of character-presence in each S_i. Leaf nodes in this tree represent the partitions.

This concept of partitioning is exemplified in Fig. 1, where the tree has been grown upto height three. The first, second and third level partitioning sets are $S_1 = \{r, c\}$, $S_2 = \{e\}$ and $S_3 = \{k, t\}$, respectively. 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. In general, $2^{|S_i|}$ branches will leave a node at level i. Here, $|S_i|$ denotes the number of characters in S_i. According to the tree in Fig. 1, "www.rocket.com" will be assigned to the left-most leaf node in the tree. In this partitioning strategy, tree height will grow with network size and the maximum tree height is restricted by the allowed character-set in the published names.

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 corpus of content names are parsed to compute character frequencies, (ii) all possible combinations of characters upto a fixed length (4 in our experiments) are generated, and (iii) the combination that produces the most balanced branching is selected. For example, let us assume that the tree is grown upto 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 (leaving out r, c, and e) of length at most 4 and select the combination that produces the most balanced branching at height 3, which is $\{k, t\}$ in the figure. The partitioning set can be precomputed given a name corpus and the character distribution computed from a large enough corpus is highly unlikely to change.

Non-English characters in a string may be treated in two alternative ways. First, if we want to limit ourselves to the 36 alpha-numeric characters in English, then non-English characters can be mapped to English characters using some predefined rules. Alternatively, we can incorporate non-English characters in the partitioning process, which may increase routing overhead. It is worth noting that we can accommodate around 64 billion nodes using the partitioning tree of 36 alpha-numeric characters.

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 architecture which achieves both of these goals. In the next section we will present a mapping algorithm to achieve these goals in an underlay network as closely as possible.

Routing table: Each partition in the aforementioned partitioning tree can be identified by a *pattern* $s_1 s_2 \ldots s_h$ where, s_i is a character presence combination over the characters of S_i and h is the height of the partitioning tree. Each leaf node of the aforementioned tree corresponds to an AS in the Internet. For example if $S_i = \{r, c\}$ then s_i can be any of rc, $r\bar{c}$, $\bar{r}c$ or $\bar{r}\bar{c}$. We define a *prefix* of a *pattern* as a leftmost sub-pattern of any length, e.g., $s_1 s_2 \ldots s_t$, where $t \leq h$.

Now we describe the routing table entries for the AS responsible for partition $s_1 s_2 \ldots s_i \ldots s_h$. For some level i, the AS's routing table will have $2^{|S_i|} - 1$ routing links corresponding to the partitions $s_1 s_2 \ldots t_i \ldots s_h$, where t_i is a character presence combination over the characters of S_i and $t_i \neq s_i$. In general, the routing table at each AS will have $\sum_{i=1}^{h}(2^{|S_i|} - 1)$ entries.

We can better describe the routing table entries with an example. Consider the shaded AS in Fig. 1 with prefix $\bar{r}c - \bar{e} - k\bar{t}$. 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, prefix characters corresponding to S_1 and S_3 will 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 prefix than other ASs, *e.g.*, the AS with prefix $rc - \bar{e}$; its first routing entry would be $r\bar{c} - \bar{e}$, which is an internal(logical) node. Here any AS with prefix $r\bar{c} - \bar{e}$ can be considered as the first 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. 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 2^{nd} 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 partition tree, as in the example of Fig. 1, does not exist in terms of network links because logical nodes in the tree are not assigned to any physical entity in the network. Rather, the tree exists logically at each indexing ASs 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. 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 minima, i.e., a node with shorter prefix than all of its neighbors. In case of a tie, X will choose the node storing higher number of index records. Once the local minima, say Y is found, X will request Y to increase its prefix by one step. If prefix of Y is $s_1 s_2 \dots s_t$ and Y has $2^{|S_t|}$ siblings in level t then Y will increase its prefix to $s_1 s_2 \dots s_{t+1}$ and X will become a new sibling of Y, otherwise X will become a sibling of Y at level t. Accordingly, X has to populate its routing table using the routing information at Y. It can be trivially proven that all the neighbors of X will be within 2 hops from Y.

III. From Overlay to Underlay

In order to route using αRoute, we have to map the nodes in the αRoute overlay graph to the AS topology. In this section we first explain the impact of Internet topology on the mapping process (Section III-A), then we present the mapping algorithm (Section III-B) followed by the lookup (Section III-C) and caching (Section III-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, following the current tradition, we assume that ASs will collaborate using αRoute for inter-domain routing, while for intra-domain routing an AS may extend its own αRoute prefix or it may use a separate intra-domain routing protocol.

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

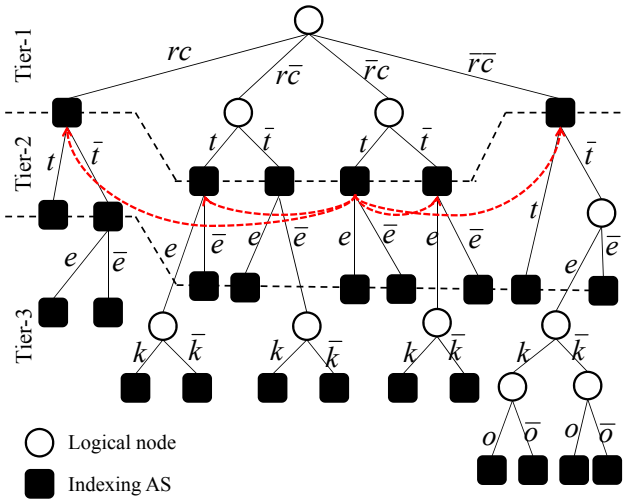

Fig. 2. Mapping a partitioning tree to AS-topology

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 [4] 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 [5] on Internet topology

Algorithm 1 PerformNeighbourMapping(ξ, ρ)

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

```
S_patterns ← set of all unmapped patterns starting with ρ
while There are unmapped neighbours do
    nbr ← neighbour with highest number of mapped neighbours from ξ
    nbr.pattern ← select a pattern from S_patterns that minimizes the hamming distance between nbr and its neighbours
    Remove nbr.pattern from S_patterns
    Mark nbr as mapped
end while
```

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 [6] 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 heterogeneous 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.

B. Mapping Algorithm

The mapping procedure is initiated by a centralized entity referred to as the Name Assignment Authority (NAA). The NAA chooses a set of prefixes and assigns them to the Tier-1 ASs. The prefixes are selected in such a way that the expected name resolution related processing load on each Tier-1 AS is distributed proportionally to its capacity. In the next step, each Tier-1 AS executes Algorithm 1 to assign prefixes to Tier-2 ASs. 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 neighbour (nbr) that has the highest number of mapped neighbours. nbr is then assigned a prefix in such a way that its distance (in the hamming space) from all its neighbours is minimized and the process goes on until all neighbours are mapped. After the Tier-1 ASs have executed this mapping process, the already mapped Tier-2 ASs map their neighbours using the same Algorithm. The process goes on in a nested recursive manner until each AS is mapped to a prefix.

In terms of mapping an edge in overlay graph (or 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 has more than one mismatch (hamming distance) between their prefixes, it 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 results 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 αRoute graph to a physical path in underlay AS-topology. In the experimental results section, we will provide quantitative measures for these three cases.

C. Content lookup

To lookup a content, we first create a pattern depending on the presence or absence of the letters in the given name (or keywords) matching the partitioning strings (S_is). Then we can use αRoute to route the lookup request to the AS indexing the names matching this pattern. At the indexing AS, we will find one or more index records of the form $< N_l, P_l >$, which indicates that the content with name N_l is stored at AS with pattern P_l. Now we use αRoute to reach the AS responsible for pattern P_l.

Each AS has to maintain a routing table (as explained in Section III-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 1 and Fig 2), the routing table will contain an entry like $< L_k, I_k, h_k >$. 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 keep separate routing entries like $< P_k, I_k, 1 >$ 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

Strict index placement restriction is a major disadvantage for any DHT approach. To enable efficient content lookup we have to place a content's index at a specific network location. In addition, it introduces 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. Moreover, 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 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 outgoing lookup requests for resolving future lookup requests. This index caching strategy will effectively reduce the popular content lookup traffic at the rendezvous indexing AS.

Content caching: As previously reported [7], [8], content popularity in the Internet and hence lookup rate follows 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, we may face two barriers while deploying this strategy. First, a content owner may not allow the indexing AS to cache and serve its content due to financial and legal reasons. Second, an AS may be overloaded if the distribution of popular contents over the ASs is not uniform. The second obstacle can be reduced if ASs cache a popular,

permitted content and update content's index by adding a link to the cached copy. In the later strategy, 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. This approach can be effective while accessing large contents.

IV. Related Work

The last few years have witnessed significant number of research efforts in the field of ICN. Several of these research works address content naming and routing as key research challenges in ICN and have proposed different solutions for these problems. A comprehensive survey on different naming and routing schemes proposed for ICN can be found in [9]. DONA [10] provides a hierarchical name resolution infrastructure including new network entities named Resolution Handlers (RH), which stores routing information of the domain it is attached with. The root RH needs to maintain routing information for all content in the network, which severely confines the scalability of this mechanism. NetInf [11] and LANES [12] both proposes a hierarchical DHT based approach for ICN routing. However, the topmost level in the DHT hierarchy in NetInf, called REX, needs to store index for all the contents in the network, which results in a performance bottleneck and a scalability issue. CCN [13], CBCB [14], and TRIAD [15] 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 the routing table. On the other hand, routing in CBCB [14] is based on controlled flooding of attribute-value pairs.

V. Conclusion and Future Work

In this paper we proposed a novel name based overlay routing scheme αRoute and an effective strategy for mapping the overlay network to physical AS-topology. αRoute guarantees content lookup while ensuring efficient bandwidth usage and small routing table size. The proposed mapping strategy, on the other hand, produces small expansion in routing path. 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. A capacity proportional load distribution can be achieved by placing the ASs at different levels in the partitioning tree based on capacity. Fourth, compared to other tree-based routing approaches, we can conveniently select the size of partitioning sets ($|S_i|$), to tune the depth of the tree. This will allow us to easily decrease routing hops by increasing the number of routing-links, and vice versa. However, the proposed partitioning algorithm for constructing S_is has a shortcoming. We currently select S_is off-line in such a way that the sample names are uniformly distributed over the leafs of the tree. For a fairly large sample size, offline computation should give nearly uniform distribution of names over the resolution nodes. We intend to investigate other techniques for online computation of S_is. In addition, the performance of αRoute can be greatly improved by adopting the caching strategies proposed in Section III-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," http://bgp.potaroo.net/.

[2] We Knew The Web Was Big. [Online]. Available: http://googleblog.blogspot.com/2008/07/we-knew-webwas-big.html

[3] "Domain Counts & Internet Statistics," http://www.domaintools.com/internet-statistics.

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

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

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

[7] 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.

[8] O. Saleh and M. Hefeeda, "Modeling and Caching of Peer-to-Peer Traffic," in *ICNP 2006*, pp. 249 –258.

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

[10] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, August 2007.

[11] C. Dannewitz, "NetInf: An Information-Centric Design for the Future Internet," in *Proc. GI/ITG KuVS Workshop on The Future Internet 2009*.

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

[13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. Braynard, "Networking named content," in *CoNEXT*, 2009.

[14] A. Carzaniga, M. J. Rutherford, and A. L. Wolf, "A Routing Scheme for Content-Based Networking," in *INFOCOM 2004*.

[15] D. Cheriton and M. Gritter, "TRIAD: a scalable deployable NAT-based Internet architecture," Technical Report, January 2000.