LIRA: A Location Independent Routing Layer based on Source-Provided Ephemeral Names

Ioannis Psaras, Konstantinos V. Katsaros, Lorenzo Saino, George Pavlou
Dept. of Electrical & Electronic Engineering, University College London
WC1E 7JE, Torrington Place, London, UK
{i.psaras, k.katsaros, l.saino, g.pavlou}@ucl.ac.uk

ABSTRACT

We identify the obstacles hindering the deployment of Information Centric Networking (ICN) and the shift from the current IP architecture. In particular, we argue that scalability of name resolution and the lack of control of content access from content providers are two important barriers that keep ICN away from deployment. We design solutions to incentivise ICN deployment and present a new network architecture that incorporates an extra layer in the protocol stack (the *Location Independent Routing Layer*, LIRA) to integrate location-independent content delivery. According to our design, content names need not (and should not) be *permanent*, but rather should be *ephemeral*. Resolution of non-permanent names requires the involvement of content providers, enabling desirable features such as request logging and cache purging, while avoiding the need for the deployment of a new name resolution infrastructure. Our results show that with half of the network's nodes operating under the LIRA framework, we can get the full gain of the ICN mode of operation.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*

Keywords

Location independence, Name-based routing, ICN

1. INTRODUCTION

Network routing based on content identifiers has recently become a topic of extensive discussion, due to the benefits that could be provided by a location-independent data distribution network [43], more commonly referred to as an Information-Centric Network (ICN). For instance, the ICN *request-response* mode of operation alleviates client mobility issues [41] and natively supports interdomain multicast [36]. Furthermore, content security (as opposed to channel security) is inherently supported by transmitting signed copies of content [17]. This in turn allows for in-network caching, which can transform the Internet into a native content distribution network [44]. Finally, as shown recently [32], the ICN paradigm can bring benefits also at the transport layer, where caches can be exploited to alleviate congestion.

On the other hand, enormous effort has been spent to de-ossify the end-to-end Internet transmission model to enable new functionalities. Examples include IP Multicast and Anycast and supporting IP mobility at the network layer [15], [4], [18]. However, the difficulties of deploying those solutions at large scale led to the design of application-layer solutions such as overlay caching instead of native, in-network caching, overlay indirection techniques [39], [30], [20], DNSSEC and IPSEC to enhance security, just to name a few. Even though these solutions have the potential to enable new services (or applications), they appear inferior compared to an ICN mode of operation, as they cannot natively support security, mobility, in-network caching and multicast: in all cases the in-network forwarding entities are forced to operate on the five-tuple `<sourceIP, destinationIP, sourcePort, destinationPort, protocol>` being, therefore, completely content-agnostic.

Arguably, the ICN paradigm has the potential to deal with the Internet's most daunting problems in a native manner. To reach this point, however, a new architecture based on core ICN principles will have to be deployed over the current IP Internet architecture, clearly, a rather challenging task.

In this paper, we identify two main obstacles that hinder the deployment of ICN on top of the current Internet. These are: *i)* scalability of name resolution, a core networking problem [28] and *ii)* content provider-controlled access to content, a business model problem, which however, is deeply integrated into the core networking principles of today's Internet and therefore, affects the design of any new architecture. Content access control here is linked to content access logging and the transmission of content transparently to the content provider from in-network caches. We discuss each of these two challenges in more detail next. Based on these considerations, in this paper, we propose a fully backward-compatible and incrementally-deployable ICN-oriented architecture that meets scalability concerns, but at the same time takes into account the business requirements of the main Internet market players.

1.1 Name resolution scalability

Two main schools of thought have emerged in the ICN-related literature regarding name resolution and name-to-

location mapping. The first one, mainly adopted by the original CCN/NDN proposal [17], advocates the hop-by-hop resolution of *requests* or *Interests* at the data plane. Effectively, name resolution is coupled with name-based forwarding with each Interest packet being locally resolved to the next (router) hop. This approach has the advantage of locally making forwarding decisions, but on the downside, huge volumes of state need to be maintained in (manually-set) FIB tables [29]. CCN/NDN routers effectively have to keep state *per packet*, an issue traditionally considered as an implementation challenge [40].[1] To deal with the scalability problems [22] of the original proposal and the huge state that needs to be kept at all routers, recent developments in the NDN space have proposed an NDN-based DNS system, dubbed NDNS [1], as well as the involvement of content providers to help in the name-resolution process [2].

The second school of thought decouples name-resolution from name-based routing by using a separate name-resolution system, similar in nature to DNS (*e.g.*, [11], [5], [42], [21]). Although this approach avoids pushing excessive state to router forwarding tables, it requires the deployment of new infrastructure by operators. For instance, as shown in [22], the support of the DONA [23] architecture at tier-1 Autonomous Systems (ASes) requires the deployment of small-to-medium size data centres to support name resolution. Such, extra infrastructure built in from scratch has the obvious downsides of huge investment requirements, as well as the shift challenge to this new mode of operation.

Moreover, focusing on the practical deployment of ICN, the full cycle of the name resolution process still remains unclear. Name resolution and data delivery mechanisms often build on the implicit assumption that content names or identifiers are already available to the end users, prior the aforementioned coupled or decoupled resolution steps. Obviously, developing a mechanism for the retrieval and delivery of content names to the end users raises concerns regarding both scalability aspects related to the enormous size of the namespace, and compatibility issues with respect to both application and network layer interfaces.

We also note that the requirements of today's dynamic and interactive applications would not be served adequately by fully transparent in-network caching driven solely from search engine content name results. We discuss and evaluate these concerns later.

1.2 Content access control

Content Providers (CPs) and CDNs require, for commercial and regulatory reasons, full control over the content requested and transferred. This has been largely overlooked by research efforts in the ICN area, which have mainly focused on naming schemes and name resolution systems to address scalability issues. For instance, the consensus around opaque and permanent content names ignores the fact that content can be served from ISP-operated in-network caches, transparently to the CP or CDN. *"Pay per click"* business models, however, would face significant limitations from this design choice in an ICN setting, that would practically prevent CPs and CDNs from billing their customers. Alternative approaches based on ISP-CDN collaborations to log content requests cannot but be unrealistic: DNSs can keep track of requested content and could possibly report back to the relevant CPs/CDNs. This, however, would mean that SLAs should be in place between *all ISPs and all CPs/CDNs at a global scale*, a rather unrealistic assumption.

At the same time, transparent in-network caching mechanisms would typically allow only limited control over the content delivered to clients. That is, coarse grained TTL-based mechanisms would be the only means for CPs/CDNs to manipulate updated content, leading either to the delivery of stale content, or the unnecessary delivery from the CP. That said, active cache purging is another requirement that calls for control of content from CPs and CDNs.

Although content access control might sound as a trivial implementation or a business model issue, we argue that it might well hinder the engagement of CPs and CDNs from the adoption of ICN. Summarising, we argue that these concerns of: *i)* scalability and incremental deployment support of a name-oriented architecture, and *ii)* exclusive content access control at the CP side with simultaneous support for transparent in-network caching have been overlooked by the community so far. As a consequence, the full potential of an ICN mode of operation has not been exploited in full yet, making the adoption and deployment of the ICN paradigm an unrealistic target.

1.3 Contributions

Although clean slate research has revealed many of the benefits that ICN can bring, we argue that deployability has to be put at the forefront of any ICN design, rather than being treated as an afterthought. We address the deployability concerns discussed above by introducing a novel information-focused network architecture, which overcomes scalability concerns and is fully backward compatible with the current IP architecture.

Our proposed architecture first introduces a name resolution process tailored to carefully manage information exposure *e.g.,* enabling content access logging (Section 2.1). This name resolution process is combined with a new naming scheme, which builds on the notion of *ephemeral names* (Section 2.2). *Name resolution is controlled by content providers* based on a fully backward-compatible mechanism that supports in-network caching and the direct control of ephemeral names' lifetime, thus facilitating content access logging and active purging of stale cached data. The proposed mechanism completes the full cycle of the name resolution process, delivering content names to clients, without imposing any requirement for additional mechanisms.

In-network caching, name-based routing and support for network-layer multicast are all integrated in the *Location-Independent Routing Layer* (LIRA), an extra layer in the protocol stack placed at "level 3.5" of the protocol stack, above the IP and below the transport-layer (Section 2.3). LIRA "absorbs" the location-independence nature of ICN, leaving the network layer to operate based on IP addresses. Resolution of content names does not rely on large volumes of FIB table entries, and routing takes place based on a hybrid of IP addresses (at the IP layer) and location-independent transient content names (at the LIRA layer) (Section 3). Our design does not require blanket adoption in order to realise the benefits of ICN. Instead, ISPs can incrementally deploy LIRA nodes with little investment. Furthermore, the fact that routing is (in the worst case) based on IP addresses guarantees full backward compatibility with the current Internet architecture. Our results show that

[1] See also [27] for an elaborate discussion on issues related to web transfers where the current NDN model is not sufficient.

even with a subset of nodes upgraded to support LIRA functionality, our design achieves considerable performance gains (Section 4).

2. CONCEPTS AND COMPONENTS

2.1 Content provider-assisted name resolution

In order to deal with the scalability concerns raised above, we design a name resolution scheme which involves the content provider and does not require extra name-based resolution machinery (*e.g.,* [5], [42], [11], [7]), or manually-set, bloated FIB tables (*e.g.,* [17], [13]). In particular, any user will have to consult the CP (or CDN) and "ask" for the name/`contentID` before any content transfer can start (see next section for details on the `contentID`). Users reach the content provider based on the standard procedure of the current Internet, that is, based on URLs, DNS resolution and IP addresses. This first part of the resolution (*i.e.,* reaching the CP to get the `contentID`) is based on IP addresses and is location-dependent. We note that users do not get the whole of the chunk from the CP (but only the `contentID`), which can be served from any other cache in the network. In this way, we realise *semi-transparent* in-network content caching, which we argue is in the best interests of both CP/CDNs and ISPs alike. As discussed later on in this section, the second part of the name resolution, which also leads to the content transfer itself is location-independent, according to the philosophy of ICN. Summarising, the *"content provider-controlled name resolution procedure"* introduced here is fully backward compatible and does not require extra investment from ISPs, or CPs/CDNs.

2.2 Ephemeral Names

To provide full content access control to CPs, we introduce the concept of *ephemeral names*, which are used for location-independent content delivery. Our primary motivation behind the introduction of *ephemeral names* is to avoid dissemination of the name/`cID` of a content to other users, as this could potentially lead to accessing the content from in-network caches, transparently to the CP/CDN. This section explains the structure, usage and design principles of these names.

2.2.1 Name Structure

The LIRA architecture uses flat names composed of two parts (see Fig. 1). The main part of the naming structure, the `contentID`, or `cID` reflects the name of the content itself and is based on the premise of *ephemeral or transient names*. According to this concept, content providers choose arbitrary strings and assign them to the content they host. The names are flat, in the sense that they bear no structure related to routing (*e.g.,* aggregation); however, CPs may impose structures related to the internal organisation of their content. Ephemeral names should be unique to guarantee collision-free name resolution, which can be easily achieved with the use of arbitrary hashes. The names are self-certifying and "expire" after some time interval.[2] This transitioning interval should be coarser than the time needed to support in-network caching and multicast (*e.g.,* names should not change on a per-request basis) - see Section 2.2.3 for details.

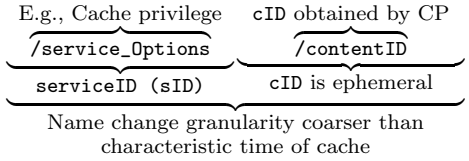


Figure 1: Ephemeral Names

The second part of the *ephemeral name*, the `serviceOptions`, can be used to realise preferential treatment of content. Although the use of this part of the name is not necessary in our architecture, and is not necessarily of ephemeral nature, we believe that it can help in the caching and scheduling process. For instance, the `serviceOptions` part can be used to flag content that should or should not be cached. We leave such investigations for future work.

2.2.2 Incentives and Disincentives for Adoption

In case of permanent names, search engines would operate based on names (similarly to today's operation based on URLs). This operation is clearly not in the best interests of CPs/CDNs given the *"pay per click"* models in use today and transparent in-network caches used in ICN.

Transient names dis-incentivise search engines from disseminating `cIDs`, but at the same time allow for both access logging at the CP/CDN and transparent in-network caching. One might claim that search engines would prefer to provide the `cID` directly to users, as this would lead to faster content access (*i.e.,* users would not need the extra RTT to travel to the CP/CDN to get the `cID`). However, given (i) the transient character of names, and (ii) the delivery of bundles of `cIDs` by CPs (see Section 2.2.3), this would require search engines to devise mechanisms for retrieving and disseminating `cIDs` each time they change, only to save a single RTT in each bundle. This limits the incentives of search engines to provide `cIDs` without the consent of CPs/CDNs.

Moreover, and most importantly, ephemeral names allow CPs/CDNs to actively control the cached content served to their clients *e.g.,* by changing the `cIDs` of content chunks existing cached copies get practically invalidated. This is an important feature of the proposed approach, which cannot be supported in alternative proposals (*e.g.,* [17, 9, 13]).

2.2.3 Transitioning Interval

The combination of the name resolution at the CP, together with the ephemeral nature of content names supports a number of desirable features. First and foremost, name resolution is under the control of the CP, enabling access logging. Secondly, versioning of updated content and purging of old content from in-network caches is also under the control of the CP.

Although TTL-like techniques, such as the CCN staleness option, can support content updating, it is not easy to set such values given today's interactive applications. Setting TTL values for individual content items (*e.g.,* [3]) would always face the tradeoff of short TTLs resulting in unnecessary delivery from the content provider, while longer TTLs would result in delivering outdated content. Using ephemeral names, cached content can instead be actively invalidated when needed.

[2]A few randomly set padding bits can be used in each named chunk to preserve both the self-certifying and the ephemeral character of names, without inflating the chunk size.

Along the same lines, the transitioning interval of ephemeral content names is an issue that requires further attention and is related, among others, to the popularity of the content as well as the size of content chunks. Frequent change of the name can result in suboptimal performance, since each change purges the content in caches. We deal with this tradeoff by setting the transitioning interval of content names to a value inversely proportional to the popularity of the content itself. Popularity is measured by the CP and can be based on the number of requests for the content in question, per some time interval. Although more sophisticated settings can be found, with this simple setting for the transitioning interval we avoid changing the `cID` of rarely accessed content too frequently, and we also avoid leaving the `cID` of popular content the same for too long.

Finally, to alleviate the need to travel to the CP for every chunk request, we assume that upon each request for a content item, the CP sends back to the client the "up-to-date" ephemeral names of the next few subsequent chunks, that is, not only the name of the immediately following one. The number of subsequent `cID`s sent by the CP to the client is left for future investigation.

2.3 LIRA: Location-Independent Routing Layer

Adding extra functionality, or altering completely the operation of *existing* core network protocols can prove difficult to be done incrementally (*e.g.,* IPv6) and "flag-days" are not an option for incorporating new components at a global scale. For these reasons, we propose *addition* instead of *replacement* of an extra layer to the protocol stack, which we call *Location-Independent Routing Layer* (LIRA). LIRA sits on top of the network (IP) layer and below the transport layer. It operates based on *ephemeral names* and integrates all the required functionality to realise *location independence*, taking advantage of *information centricity* and its well-known gains [43].

Although recent studies have proposed HTTP as the layer that can integrate information or content centricity [30], here we argue that in order for in-network caching and multicast to be smoothly incorporated in the new ecosystem, any information-centric operation needs to be *below the transport layer*. Otherwise, the transport protocol can merely connect two specific endpoints cancelling any notion of location-independent content transfer. Instead, breaking the end-to-end transmission model below the transport layer allows to leverage (ICN enabled) in-network caching, both in terms of native multi-source routing and localised congestion control [32], going far beyond traditional IP Multicast or Anycast mechanisms.

LIRA is implemented in just a small subset of nodes (see Section 2.5), which can be transparently planted in the network, and it manages incoming and outgoing content based on their names. The main name management functionality is implemented in a routing table, which we call *Content Forwarding Information Base* (C-FIB) (Section 2.4).

A similar notion to the LIRA layer has been proposed in the past in [8], but in a totally different context, addressing the exhaustion of IPv4 addresses. The evolution of NAT boxes (together with the painfully slow incremental deployment of IPv6) has dealt with this problem and hence, the related efforts became obsolete.

2.4 Content Forwarding Information Base

The Content Forwarding Information Base (C-FIB) table keeps track of recently requested and served content (in terms of `cID`s) and maintains forwarding information used for the delivery of those content items, providing also support for in-network caching and multicast. Upon subsequent request(s) for a content already in the C-FIB table, LIRA is redirecting requests towards the direction where the content has been sent, or served from, similarly in principle to breadcrumbs routing [33].

We note that *the C-FIB table essentially acts as a cache* for `cID`s served recently through this router (somewhat similarly to [26] and [14]). However, C-FIB table entries are not permanent, as in CCN's FIB, but rather are assisting in location-independent content delivery from neighbouring nodes (see Section 3 for details on the C-FIB structure).

The typical structure of the C-FIB table is illustrated in Table 1. The table maintains one entry per content chunk. The following information is maintained for each entry: *i)* `cID`, the content identifier of the chunk, *ii)* if_I, the *incoming interface i.e.*, the index of the interface from which the content is received, that is, the content source indicated by the DNS, *iii)* if_O, the *outgoing interface i.e.,* the index(es) of the interface(s) towards which the content is currently being forwarded, *iv)* if_{TI}, the *temporary interface*, *i.e.,* the index(es) of the interface(s) where the content has been forwarded, *v)* mIP, the *multicast IP* field that holds IP addresses of clients participating in a multicast session. Note that interface entries in the C-FIB table denote real interfaces (*i.e.,* directions towards which requests/content should be forwarded) and not IP addresses of sources/destinations (apart from the multicast IP field). By doing so we realise the *location independence* property of ICN in LIRA.

2.5 LIRA Nodes

The LIRA node structure is the main component of the proposed architecture, which integrates information centricity. LIRA nodes implement the LIRA layer with its C-FIB table discussed above in order to realise named content management and subsequently location independence. LIRA nodes also include caches that temporarily store named content chunks (*i.e.,* in-network caching). Although by default all LIRA nodes include both the C-FIB table and content caches, we also evaluate (in Section 4) the case of "lighter" LIRA nodes, where, based on node centrality metrics and to facilitate incremental deployment, some nodes implement the C-FIB table and some others implement caches.

Our design does not require all nodes of a domain to become LIRA nodes and it is operational regardless of this. Being always based on IP, nodes fall back to normal IP operation and route towards the direction indicated by location-based addresses. Note that all routers maintain the default IP-based FIB table. Therefore, incompatibility issues or requirements for simultaneous shift to ICN operation do not exist. As we show later on in the evaluation section, an average of 50% of nodes within a domain can provide considerable performance gain. Careful network planning (*e.g.,* depending on topological issues) and incremental upgrade of normal routers to LIRA nodes gives a major advantage to the proposed architecture in terms of deployability compared to other ICN architectures.

3. OVERVIEW OF MAIN OPERATIONS

We proceed with the description of the name resolution

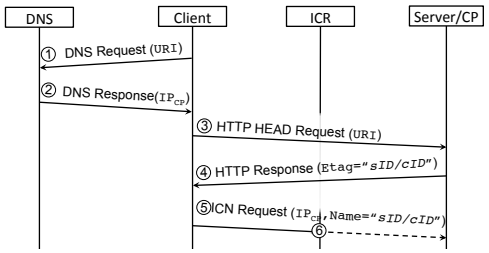


Figure 2: Name resolution and content delivery

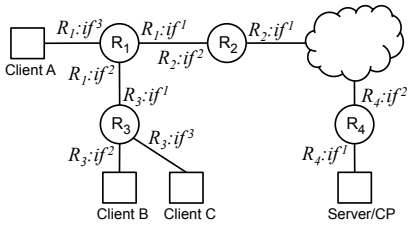


Figure 3: Example topology: labels $R_i : if^j$ denote the index j of each router's (R_i) interface.

R_1	cID	if_I	if_O	if_{TI}	mIP
t_1	x_1	1	3	-	-
t_2	x_1	1	-	3	-
t_4	x_1	1	2	3	-
t_5	x_1	1	-	2, 3	-
t_7	x_2	1	3	-	-
t_9	x_2	1	2, 3	-	B's IP
t_{10}	x_2	1	-	2, 3	-

Table 1: Routing Table at R_1 - Fig. 3

R_3	cID	if_I	if_O	if_{TI}	mIP
t_3	x_1	1	2	-	-
t_6	x_1	1	-	2	-
t_8	x_2	1	2	-	-
t_{10}	x_2	1	-	2	-

Table 2: Routing Table at R_3 - Fig. 3

and content delivery process, illustrated in Fig. 2. We then give details of the entries of the *Content Forwarding Information Base* (C-FIB) table during the content delivery process. For this purpose we use the network topology presented in Fig. 3. Tables 1 and 2 are also used to present the entries of the C-FIB table(s) for a sequence of important events taking place in our example scenarios (denoted with timestamp t_i).

3.1 Name Resolution and Content Delivery

The name resolution process is initiated through existing protocols (*i.e.,* DNS and HTTP) to guarantee backwards compatibility and facilitate adoption of ICN.

(1) As a first step (Fig. 2) and identically to what is happening today, users resolve URLs through a request to the DNS. (2) The DNS responds with the IP address of the content provider. (3) The user generates an HTTP HEAD request [16] at the application layer. At this stage, routing is location-dependent and is based on the IP address indicated by the DNS. At the content layer, the request is asking for the `cID`. (4) The CP sends back an HTTP response packet containing the up-to-date name, *i.e.,* `cID`, of the requested content in the ETag field of the HTTP response header [16].[3] The destination IP address of that packet is that of the requesting client. This packet can be piggybacked with data to avoid an extra RTT between the client and the CP. In this case, however, given that requests are sent per chunk, we cannot take advantage of in-network caching. This option can be considered in special cases (*e.g.,* when a client is close to the CP and chances of finding the content cached are slim). (5) The client issues a request for the first chunk of the content object (*e.g.,* client A in the example of Fig. 3). The request includes the IP address of the CP at the IP layer and the `cID` of the chunk at the LIRA layer.

[3]It is noted that the use of the ETag field for name resolution does not change the semantics of the field as it is intended to describe the content to be delivered and/or cached. Moreover, no restrictions apply to the format of this field allowing the realisation of ephemeral names.

(6) LIRA nodes along the path check the `cID` included in the request[4] against the entries of their C-FIB table. If an entry for the `cID` exists, then they forward according to this entry. If not, they forward according to the IP address. The IP address points to the CP, hence, content can always be resolved according to that in the worst case, *e.g.,* in case of LIRA-incompatible nodes or domains.

At this point, assuming the content is not locally cached (see Section 3.2 for details on in-network caching), the request is forwarded towards the CP. The index of the network interface used to forward the request is marked as the if_I for this content chunk (*i.e.,* interface 1 - see time t_1 in Table 1). At the same time, the index of network interface from which the request was received is marked as an output interface (interface 3 in our example). The content chunk is then sent back from the CP (or any other cache further down the path). During the data transfer no change is made in the C-FIB table entries of intermediate LIRA nodes (time t_1). When the chunk transfer completes, which is denoted by an End of Chunk (EoC) field, the intermediate LIRA nodes change their C-FIB entries for this `cID` by marking the interfaces through which they forwarded the data (*i.e.,* if_O) as if_{TI} (*temporary interface*) - interface 3 is moved to if_{TI} at t_2 in Table 1. This is done since the content can possibly be delivered from there too (*i.e.,* the content has possibly been cached towards this direction). When the client sees the EoC field/bit set, it forwards the next request towards the original CP (similarly to the initial request - step (3) above) in order to obtain the `cID` of the next chunk.

3.2 In-Network Caching

LIRA nodes by default support in-network caching. In the simplest case, on-path in-network caching is supported by simply performing a lookup of the `cID` of a request message, at the local cache index. In case the requested content chunk is cached locally, the corresponding data is returned through the network interface the request was received from (if_O). In our example scenario, client B issues a request for content

[4]We rely on the Protocol field of the IPv4 header (or the "Next header" in the IPv6 header) to enable LIRA nodes to identify those IP packets that can be handled by the LIRA layer *i.e.,* containing an ICN content name.

x_1. Once the request for x_1 reaches R_3, the C-FIB table of R_3 is updated to include $if_I = 1$ and $if_O = 2$ (t_3 in Table 2). Then, at time t_4, the request for x_1 reaches R_1. Content chunk x_1 is found cached at R_1 whose interface 2 is marked as if_O and the content is sent towards client B.

By introducing the if_{TI} field in the C-FIB table we further realise off-path in-network caching [26], as well as user-assisted in-network caching [25], [37]. When a content chuck is not found in the local cache, the LIRA node sends the received request towards both the (permanent) incoming interface if_I (as indicated by the name resolution process) and the temporary interface(s) if_{TI}. In our example, R_1 sends two requests for x_1 towards both the (permanent) if_I 1 and the if_{TI} 3 (t_4 in Table 1). Whichever of the two interfaces (1 or 3) starts receiving the requested data first is marked as the incoming interface for this content and the remaining (temporary) interfaces are pruned down. Pruning here can be realised through a negative ACK (NACK) packet which travels towards the source of the content. If if_I answers first, the if_{TI} is removed from the corresponding C-FIB table entry. Alternative strategies can be applied here, by selectively forwarding a request to one or more of the available interfaces *e.g.,* always forwarding only towards an off-path cache, since requests are always routable to the CP at the IP layer.

Finally, at time t_5 when x_1 transfer completes (from either the local, or a remote cache), interface 2 is added to the list of temporary incoming interfaces (if_{TI}) at R_1, since x_1 can now be found this way too (similarly to t_2). The C-FIB table of R_3 is also updated to include interface 2 as if_{TI} (step t_6).

We note that in order to avoid routing loops in case no other device towards if_{TI} (client A in our case) has the content cached, we discard requests (for items in the C-FIB table) that come in through its marked if_{TI}. This is done because any LIRA node towards the if_{TI} (client A in this case) will forward the request based on its IP address (carried at the IP layer and always pointing towards the permanent content source, hence through $R1$ in our example) if it finds no entry in its C-FIB table for the requested content. In turn, upon receipt of the request, $R1$ will send the request back towards the same direction (towards client A here), since it still has got the related entry in its C-FIB table. This will result in the request travelling back and forth creating an endless routing loop.

3.3 Multicast

Multicast support is enabled through the use of the if_O and mIP fields of the C-FIB table. As described above, during the chunk transfer, the network interface of the LIRA node where the incoming data is forwarded towards is marked in the if_O field. This if_O entry enables the LIRA node to suppress any subsequent request for the same content chunk by adding an extra outgoing interface to its C-FIB. This is similar to the PIT functionality in CCN [17].[5] Note that in all above steps the IP address (at the IP layer) of request packets has been pointing to the CP and of content chunks to the corresponding clients. However, in order to realise multicast transmission in this case (*i.e.,* avoid sending a second request for the same chunk towards the same direction), the LIRA node that suppresses subsequent requests needs to keep the extra IP address of the clients that generated the requests. We deal with this situation through the "multicast IP" (mIP) field in the C-FIB table. When data arrives at the branching LIRA node, it gets forwarded to all if_O interfaces. The mIP entries are used at the IP layer to allow for the delivery of the duplicated data to the requesting recipients.

Note however that multicast forks further down the path are handled locally. In our example, if an additional client C attaches to R_3 and requests for x_2 during the multicast session, its request will be suppressed by R_3 which will also store client C's IP address in the corresponding mIP field. R_1 will not be aware of client C's existence and R_3 is responsible for duplicating data for this client. Thus, the mIP state load is distributed to the participating LIRA nodes avoiding the overloading of nodes closer to the root of the multicast tree.

In our example network, client A issues a request for content x_2. The C-FIB table at R_1 marks $if_I = 1$ and $if_O = 3$ for `cID` x_2 (step t_7). Before the transfer of x_2 towards A completes through R_1 client B issues a request for x_2, which goes through R_3 and reaches R_1. R_3 updates its C-FIB table by putting $if_I = 1$ and $if_O = 2$ (step t_8). R_1 does not forward this request further; instead it adds interface 2 to the if_O field of x_2 and also stores the IP address of client B (taken from the corresponding IP layer field) in the mIP field (step t_9).

When x_2 arrives at R_1 (step t_9) it is forwarded towards client A through $if_O = 3$, but it is also replicated and forwarded towards client B, through $if_O = 2$, using mIP as the destination IP address. When the chunk x_2 transfer completes, router R_1 moves interfaces 2 and 3 and R_3 moves interface 2 to the if_{TI} field (step t_{10} - Table 1 and 2).

Note that the C-FIB table introduced here, incorporates the functionality of both the PIT and the FIB tables of CCN. For as long as the chunk transfer goes on and hence, the if_O field is filled (and the if_{TI} field is empty - t_1, t_7 and t_9 in R_1's C-FIB, see Table 1), the C-FIB table represents the PIT table of CCN/NDN. That is, based on this state, LIRA nodes are able to collapse/suppress subsequent requests for content already requested (or under transmission) and realise multicast. When the chunk transfer completes and the entry in if_O is moved to the if_{TI} field (t_2, t_4, t_5 and t_{10} in Table 1), then the C-FIB table reflects the FIB table of CCN/NDN. As mentioned above, however, the C-FIB table acts as a cache for recently served content and hence, it does not need to keep huge amounts of state information in the FIB part of the C-FIB. We discuss and evaluate both parts of the C-FIB table later in Section 4.

4. PERFORMANCE EVALUATION

It is generally not common to evaluate a network architecture merely in quantitative terms, given that the contribution of such studies comes mainly at a conceptual level. In our case, the contribution of the LIRA architecture comes mainly in terms of incremental deployment with backward compatibility guarantees. At the same time, however, LIRA can achieve all the quantifiable benefits of an ICN mode of operation.

To provide a thorough performance evaluation, we analyse conceptual and qualitative gains in Sec. 4.1 as well as quantitative gains in Sec. 4.2. The quantitative evaluation focuses

[5]Effectively, the C-FIB collapses both the CCN FIB and PIT in one table.

on the deployment of the LIRA concept from the operators perspective. In particular, given a fixed monetary budget that the operator is prepared to spend in order to deploy LIRA, we assess the best strategies of investing the capital in terms of extra equipment, which in our case translates to cache memory and C-FIB tables. We also demonstrate and quantify the benefits brought by LIRA to CPs, with a particular focus on cache purging and the control over the freshness of the cached content.

4.1 Qualitative Evaluation

Name resolution: by handing control of the name resolution process to CPs, LIRA avoids the need for either the deployment of a costly name resolution infrastructure, or the investment on in-network resources for the support of line-speed name resolution. The operation of the C-FIB as a cache for names/`cIDs` is similar to [14]. However, LIRA does not necessitate the use of an explicit off-path name resolution mechanism, as it rather falls back to IP, in a backward compatible manner. At the same time, by following a backwards compatible HTTP-supported name resolution mechanism, LIRA presents a complete interface for the interaction of end-hosts with an information-centric network. To the best of our knowledge, no exact mechanism has been proposed for the discovery (*i.e.,* not only resolution) of content names in alternative ICN architectural proposals.

Control of content access: LIRA enables CPs to directly monitor and control the access of end users to their content. Route-by-name approaches such as [17] fail to provide such support. CPs would be reluctant to accept transparent access to their content, thus dis-incentivising the adoption of such an approach to ICN by ISPs. Lookup-by-name approaches, on the other hand, such as [10] and [21], enable this type of control, by decoupling name resolution from forwarding. However, this comes at the cost of additional name resolution infrastructure and directly places the content access information in the hands of ISPs; in turn, this introduces the burden of new (business and technical) interfaces between all CPs and all ISPs at a global scale.

Mobility: although the issue of mobility in case of LIRA requires further investigation and at first sight it might seem that LIRA cannot deal with mobility efficiently, due to its dependence on IP, we note the following: upon a content request, the CP or CDN is sending back to the client the `cIDs` of the next few chunks, *i.e.,* not just the next one. That said, the clients operate based on IP-agnostic `cIDs`. Therefore, client mobility can be natively supported, as clients request for content based on identifiers (in combination to the IP address at the IP layer). Source mobility, on the other hand, is an issue that requires further investigation as is the case with all ICN architectural proposals.

Security: by supporting self-certifying `cIDs`, LIRA secures the content itself rather than the communication channel, similarly to other ICN architectures *e.g.,* [17].

Implementation: the proposed LIRA functionalities can be deployed on nodes with only firmware updates without the need for hardware replacement or upgrade. In fact, by relying on IP forwarding as a fallback in case of C-FIB misses, LIRA will never result in un-routable requests/content even if deployed on just a few nodes and with minimal memory. This is in stark contrast with previous ICN proposals like CCN and NDN which require well-dimensioned FIB and PIT structures to operate correctly and at line speed. C-FIB can be loaded in DRAM, which has been shown to be able to support line-speed per-packet lookups [45], [28], is inexpensive and abundant on modern routers based on either network processors or general purpose processors. It is in fact common to have at least a few GBs of spare DRAM on modern routers. Since the binary code implementing LIRA functionalities is likely to require negligible space, all available DRAM can be used for C-FIB and caching space. C-FIB entries in particular have very low memory requirements. In fact, even assuming that (i) the C-FIB is implemented using a hash-table with a load factor of 0.5 and with a circular queue for replacement and (ii) the unfavourable case that LIRA chunks are named using SHA-512 hashes and next hops information are coded on 2B, it is still possible to store over 15 million C-FIB entries per GB of DRAM. This makes C-FIBs and more generally the LIRA node architecture easy to incrementally deploy on today's routers.

4.2 Quantitative Evaluation

As mentioned earlier, the objective of this section is to evaluate the best possible way to invest in deploying the LIRA concept, from the operator's perspective. That said, we initially evaluate the main concepts of our proposal with regard to their projected gains in terms of cache hits. Although LIRA is far from a caching-specific architecture, caching is: *i)* the only straightforward quantitatively measurable aspect of an ICN architecture, and most importantly, *ii)* the main feature that requires investment from network operators. For these reasons and without by any means underestimating the gains from the above-mentioned qualitative benefits of the LIRA architecture, in this section, we focus on the evaluation of the main concepts included in LIRA as seen from an in-network caching perspective.

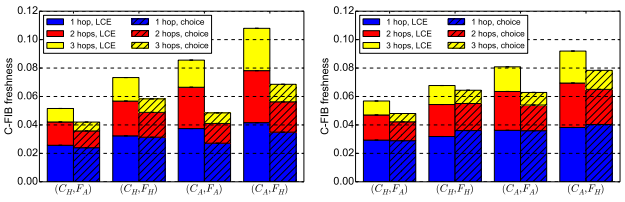
We use Icarus [35] to evaluate the performance of various aspects of our proposed framework based on real ISP topologies from the RocketFuel dataset [38] and synthetic workloads [19]. Due to space limitations, we omit evaluation of the multicast functionality offered by LIRA, since the related performance benefit is straightforward. Moreover, we only show results for the Telstra and Abovenet topologies, though we report that we obtain similar results with other topologies as well. We make the code, documentation and data required to reproduce our results publicly available.[6]

4.2.1 Efficiency of C-FIB and Deployment Strategies

LIRA nodes can have a content cache or a C-FIB table, or both. Given a fixed total cache and C-FIB capacity budget, in this section, we identify the best possible combination of cache and C-FIB deployment along two dimensions: i) deployment strategies and ii) caching strategies. We attempt to capture the interaction dynamics between nodes that cache content and nodes that can route to this content, in a location-independent manner, *i.e.,* through C-FIB table entries. *Our first objective is to investigate the effectiveness of C-FIB table entries in mapping the content cached in neighbour nodes. Our second objective is to see how C-FIB table entries eventually translate to cache hits.*

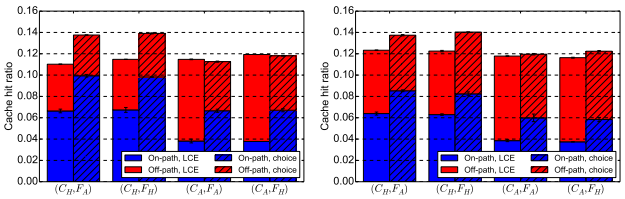
Modelling the performance of a network of caches is known to be complex [34], [12]. As a result, it is extremely difficult to formulate optimal cache placement algorithms which are also robust to realistic traffic variations. Arguably, the complexity of the optimal cache placement problem is another

[6] `http://www.ee.ucl.ac.uk/~lsaino/software/lira`



(a) Telstra (b) Abovenet

Figure 4: C-FIB Freshness vs. Deployment Strategies and Caching Strategies, Zipf $\alpha = 0.8$



(a) Telstra (b) Abovenet

Figure 5: Cache Hit Ratio vs. Deployment Strategies and Caching Strategies, Zipf $\alpha = 0.8$

obstacle hindering ICN deployments. Therefore, motivated by practical reasons, we propose four simple content cache and C-FIB placement algorithms and show that they are sufficient to provide tangible performance gains even with partial deployments. To deploy caches and C-FIBs, we rank nodes according to their betweenness centrality (*i.e.,* the amount of traffic traversing them following shortest path routing [6]) and deploy LIRA functionality using the following strategies:
(i) Cache in top 50% high centrality nodes, C-FIB table in all nodes: (C_H, F_A).
(ii) Cache in top 50% high centrality nodes, C-FIB table in top 50% high centrality nodes: (C_H, F_H).
(iii) Cache in all nodes, C-FIB table in all nodes: (C_A, F_A).
(iv) Cache in all nodes, C-FIB table in top 50% high centrality nodes: (C_A, F_H).

We run simulations and measure the mean *C-FIB freshness*, which we define as the ratio of entries stored in C-FIB tables which can correctly route to a copy of a content stored in a nearby cache. This metric captures how well the entries of the C-FIB tables deployed in the network reflect the current state of nearby caches. We further characterise the correct C-FIB entries by the hop distance to the LIRA node that caches the corresponding content. Note that in all cases, and regardless of the deployment strategy, the ratio of C-FIB table to cache entries is fixed (see next subsection for the evaluation of this ratio). As a result, C-FIB tables in fewer nodes (than those that deploy caches) keep more entries to match the number of cache slots (and vice versa).

We also analyse the results under different caching strategies: *Leave Copy Everywhere* (LCE), according to which a copy of a content is stored in every cache traversed and *random choice*, according to which a content is stored only in one randomly selected caching node along the delivery path. The rationale behind our choice is to evaluate deployment and caching performance under varying *caching redundancy* [31]. Our results are shown in Figs. 4 and 5.

C-FIB Efficiency. First of all, it is important to highlight the fact that the C-FIB table entries depict precisely the state of neighbour caches. This is proved by the fact that the *freshness ratio* in Fig. 4 directly translates to off-path cache gain in Fig. 5: for instance, the freshness result in case of (C_H, F_A) in Fig. 4a indicates that 5% of entries in the C-FIB table can correctly route to the content in neighbour caches. In turn, in Fig. 5a, the gain from off-path caching (red, top part of bar) is 4.5%. This is an important result that highlights the effectiveness of the C-FIB table in keeping an accurate record of the state of nearby caches (*i.e.,* up to 3 hops away in our evaluation).

Deployment Strategy. In terms of C-FIB freshness, deploying smaller caches over more/all nodes, *i.e.,* (C_A, F_*), seems to be more effective in capturing the state of caches from the C-FIB tables (*i.e.,* higher freshness in Fig. 4). This is explained by the fact that the "monitoring and mapping" mechanism provided by the C-FIB table has got a wider view of the neighbourhood and can therefore, find more content items locally. This also translates to more off-path cache hits in Fig. 5 for (C_A, F_*).

Out of the four deployment strategies under consideration here, (C_H, F_H) and (C_H, F_A) consistently perform best in terms of cache hits (in Fig. 5). This is irrespective of the freshness result, which shows that freshness improves when caches are deployed over all nodes (*i.e.,* (C_A, F_*)). In other words, it is better to have fewer but bigger caches placed in high centrality nodes (as also shown in [6]), rather than having smaller caches deployed in all nodes of the network.

Caching Strategy. As expected, in terms of cache hits, *choice* always performs best, for all topologies and for all deployment strategies, as a result of its reduced caching redundancy. Similar results have been reported before in [31]. LCE on the other hand, performs roughly the same across all deployment strategies. Note that the LCE result in Fig. 5 effectively reveals the performance of the CCN/NDN architecture. Due to space limitations, we do not present a full-fledged comparison between the architectures, but Fig. 5 reveals very well the cache-related performance of CCN/NDN.

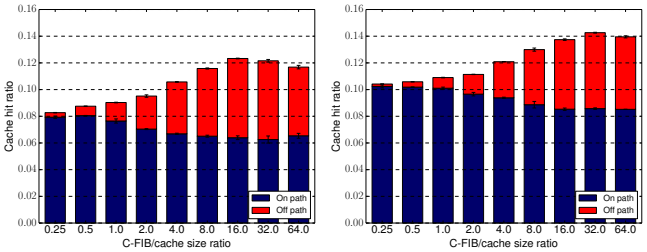
4.2.2 Memory Requirements and Scalability

We next quantify the performance benefit of off-path, C-FIB-routed, caching for various values of the C-FIB-to-cache size ratio (expressed in number of entries) and for the (C_H, F_A) strategy (Fig. 6).

Considering the overall cache hit ratio (both on- and off-path), we see a considerable increase when moving from a ratio value of 0.25 to a ratio value of 16, due to C-FIB routing redirections. The results are similar for a ratio equal to 32, but the gain in this case is marginal. Therefore, given that larger memory is required in order to deploy C-FIB tables 32 times bigger than the entries in the respective caches, we conclude that a value of 16 is optimal. Although in absolute values, off-path, C-FIB-based, caching contributes less than on-path caching, the gain is still far from negligible (*i.e.,* it can reach up to 50% in Fig. 6). We report that in our simulations, the gain from off-path caching can reach 100%, effectively doubling the gain from on-path caching.

Finally, it is interesting to note the slight decrease of on-path cache hit ratio as the C-FIB-to-cache size ratio in-

creases. This is attributed to cases where a content request encounters a C-FIB table entry and gets diverted to an off-path cache, before it actually hits an existing on-path cache. In this case, and given that the C-FIB table entry is found earlier in the path, we report that the delay to deliver the content back to the user is even shorter than finding the requested item in an on-path cache. This is especially so in case of LCE caching, where due to increased caching redundancy, a copy of a content has good chances of being found along the shortest path.



(a) LCE (b) Random Choice

Figure 6: Cache hit ratio vs C-FIB size, Abovenet

4.2.3 *Control of content to CPs*

One of the departing points in the design of the LIRA architecture is the direct control of content by the CPs/CDNs, as discussed earlier. We identify two main features that give direct control of the content to the CP or CDN. The first one is the control of access logging. In LIRA this is accomplished by the content provider-controlled name resolution, where clients need to get the up-to-date `cID` from the content provider. This requires an extra RTT to get to the CP or CDN. We remind that according to our discussion in Section 2.2.3, CPs/CDNs send more than one `cID` to the client, therefore, the journey to the CP/CDN happens rarely during the data transfer, or even only once in case of small files (*e.g.,* web). We assume this extra RTT to incur only a tiny performance penalty compared to alternative proposals that do not necessarily require this extra roundtrip.

The second feature that provides control of published content to CPs is the ability to actively perform cache purging. As described in Section 2.2, when CPs change the `cID` of a content item, previously cached items no longer get hits from new requests and eventually get evicted (denoted as *LIRA w/o replacement*). Taking a step further, we consider an extended version of this mechanism, where data packets explicitly indicate the `cID` values of the content items that should be immediately evicted from encountered caches (denoted as *LIRA w/ replacement*).

Figure 7 shows the cache hit ratio of the above mechanisms along with that of a simple TTL-based mechanism, where any cache hit returns the content to the client, even if this content is stale (*TTL-based (all hits)*) for various TTL values. In Fig. 7, we see that the cache hit ratio in the (*TTL-based (all hits)*) case increases with the value of TTL, since content remains longer in the cache. However, this also means that the corresponding cache hits result in the reception of stale content. Fig. 7 also shows the cache hit ratio for fresh only content (*TTL-based (fresh only)*), which initially increases, but then steadily drops as a result of high TTL values that increase stale cached content.

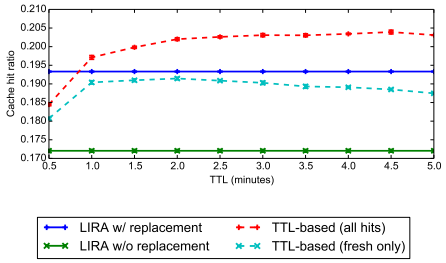


Figure 7: Delivery of Stale Content

The *LIRA w/ replacement* mechanism performs considerably better than *LIRA w/o replacement*, as it immediately frees the caching space from unnecessary stale content, and better than its *TTL-based (fresh only)* counterpart.

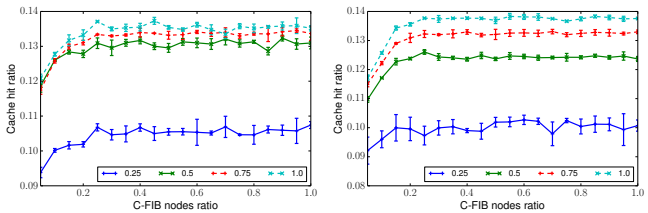
It must be noted that the *TTL-based (fresh only)* ratio is only provided as a benchmark, as TTL-based mechanisms cannot avoid serving stale content. On the other hand, LIRA provides a precise mechanism to avoid serving stale cached content altogether.

4.2.4 *Incremental Deployment*

We proceed to evaluate the last of the design targets behind LIRA, that of incremental deployability. To assess the performance gain of incrementally deploying the LIRA architecture, we begin by progressively adding C-FIB tables starting from the highest centrality nodes. We evaluate the performance in terms of cache hits in case of caches deployed in 25%, 50%, 75% and 100% of the nodes, starting from the highest centrality ones.

We observe in Fig. 8 that performance stabilises with the C-FIB table present in 20-30% of the nodes. C-FIB in less than 20% results in suboptimal performance, but performance does not increase considerably if we continue adding C-FIB to more nodes. In terms of caches, a 25% deployment rate results in poor performance, while the performance does not improve considerably when caches are deployed in more than 50% of nodes. The difference in performance between the 50% and 100% of nodes is in the area of 1% improvement in terms of cache hit ratio for the two topologies shown here (Telstra and Abovenet).

We conclude that adding C-FIB to the top 20-30% highest centrality nodes and caches to 50%-75% of highest centrality nodes achieves the full performance gain of the LIRA architecture. Although here we present results for Telstra and Abovenet topologies, our results are consistent along all six evaluated topologies of the RocketFuel dataset.



(a) Telstra (108 nodes) (b) Abovenet (141 nodes)

Figure 8: Incemental Deployment

5. CONCLUSIONS

There is a constant trend towards extra "flexibility" in communication networks, which started with the shift from (rigid) circuit-switching to (queuing-based) packet-switching [24]. We see location-independent, information-centric networking as the natural next step towards *"content switching"*. To move towards this direction, however, the research community needs to take into account the interests of the main Internet market players, as well as those of users.

We argue that ICN research so far has focused on designing conceptually sound and scalable name-based routing architectures, but largely ignored any incentives (provided through those architectures) to adopt the ICN technology. The interests of content providers and CDNs are largely different to those of ISPs and the shift to an ICN environment environment makes this difference even more pronounced. That said, unless a shift to an ICN environment takes into account the interests of both CPs/CDNs and ISPs, the incentives to adopt this technology will be limited.

In this paper we have taken these concerns into consideration and have designed an incrementally-deployable ICN architecture. The proposed architecture is based on the Location-Independent Routing Layer (LIRA) and directly involves the content provider in the name resolution process. Furthermore, ephemeral names give more power to the CPs/CDNs over the content they publish. Our evaluation shows that even with a limited number of nodes implementing the LIRA architecture, ISPs achieve a clear performance gain, while at the same time CPs/CDNs have full control of their content.

6. REFERENCES

[1] A. Afanasyev. ?addressing operational challenges in named data networking through ndns distributed database, ph.d. dissertation, ucla. Sept. 2013.

[2] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang. SNAMP: Secure Namespace Mapping to Scale NDN Forwarding. In *IEEE GI Symposium 2015*.

[3] B. Ahlgren and B. Ohlman. NetInf Protocol Extensions for Cache Control. IETF Internet-Draft draft-ahlgren-icnrg-netinf-cache-control-00.txt, Feb. 2014.

[4] H. Ballani and P. Francis. Towards a Global IP Anycast Service. In *ACM SIGCOMM âĂŹ05*, 2005.

[5] W. K. Chai and *et al.* . Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services. *IEEE Communications Magazine*, 49(3):112 –120, march 2011.

[6] W. K. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in information-centric networks (extended version). *Computer Communications*, 36(7):758–770, 2013.

[7] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan. COPSS: An Efficient Content Oriented Publish/Subscribe System. In *ACM/IEEE ANCS âĂŹ11*.

[8] D. R. Cheriton and M. Gritter. TRIAD: A New Next-Generation Internet Architecture, July 2000.

[9] C. Dannewitz. NetInf: An Information-Centric Design for the Future Internet. In *Proc. 3rd GI/ITG KuVS*, 2009.

[10] C. Dannewitz, M. D'Ambrosio, and V. Vercellone. Hierarchical DHT-based name resolution for information-centric networks. *Computer Communications*, 36(7):736–749, Apr. 2013.

[11] C. Dannewitz and *et al.* . Network of Information (NetInf) - An Information-centric Networking Architecture. *Comput. Commun.*, 36(7):721–735, Apr.

[12] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman. On the complexity of optimal routing and content caching in heterogeneous networks. In *INFOCOM, 2015 Proceedings IEEE*, April 2015.

[13] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini. CONET: A Content Centric Inter-networking Architecture. In *ACM SIGCOMM ICN Workshop*, ICN '11, pages 50–55, 2011.

[14] A. Detti, M. Pomposini, N. Blefari-Melazzi, and S. Salsano. Supporting the Web with an Information Centric Network That Routes by Name. *Comput. Netw.*, 56(17):3705–3722, Nov. 2012.

[15] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *Network, IEEE*, 14(1):78–88, 2000.

[16] R. Fielding and *et al.* . Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999.

[17] V. Jacobson and *et al.* . Networking named content. In *ACM CoNEXT 2009*, pages 1–12, New York, NY, USA.

[18] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775, June 2004.

[19] K. Katsaros, G. Xylomenos, and G. Polyzos. GlobeTraff: A Traffic Workload Generator for the Performance Evaluation of Future Internet Architectures. In *NTMS 2012*, pages 1–5, May.

[20] K. Katsaros, G. Xylomenos, and G. C. Polyzos. Multicache: An overlay architecture for information-centric networking. *Computer Networks, Elsevier*, 55:936–947, March 2011.

[21] K. V. Katsaros and *et al.* . On inter-domain name resolution for information-centric networks. In *IFIP Networking 2012*.

[22] K. V. Katsaros and *et al.* . On the Inter-domain Scalability of Route-by-Name Information-Centric Network Architectures. In *IFIP Networking '15*.

[23] T. Koponen and *et al.* . A data-oriented (and beyond) network architecture. In *ACM SIGCOMM 2007*, pages 181–192.

[24] J. Kurose. Information-centric networking: The evolution from circuits to packets to content. *Computer Networks*, 2014.

[25] H. Lee and A. Nakao. User-assisted In-network Caching in Information-centric Networking. *Comput. Netw.*, 57(16):3142–3153, Nov. 2013.

[26] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi. SCAN: Scalable Content Routing for Content-Aware Networking. In *IEEE ICC 2011*, pages 1–5, June.

[27] I. Moiseenko, M. Stapp, and D. Oran. Communication Patterns for Web Interaction in Named Data Networking. In *ACM ICN '14*.

[28] D. Perino and *et al.* . Caesar: A content router for high-speed forwarding on content names. In *ANCS '14*, pages 137–148.

[29] D. Perino and M. Varvello. A reality check for content centric networking. In *ACM SIGCOMM ICN Workshop*.

[30] L. Popa, A. Ghodsi, and I. Stoica. HTTP As the Narrow Waist of the Future Internet. In *ACM Hotnets-IX*, 2010.

[31] I. Psaras, W. K. Chai, and G. Pavlou. In-Network Cache Management and Resource Allocation for Information-Centric Networks. *IEEE TPDS*, 25(11):2920–2931, 2014.

[32] I. Psaras, L. Saino, and G. Pavlou. Revisiting Resource Pooling: The Case for In-Network Resource Sharing. In *ACM HotNets-XIII*, 2014.

[33] E. Rosensweig and J. Kurose. Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks. In *INFOCOM 2009, IEEE*, pages 2631–2635, April 2009.

[34] E. Rosensweig and J. Kurose. A network calculus for cache networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 85–89, April 2013.

[35] L. Saino, I. Psaras, and G. Pavlou. Icarus: a Caching Simulator for Information Centric Networking. In *SIMUTOOLS 2014*.

[36] M. Sarela and *et al.* . Forwarding anomalies in Bloom filter-based multicast. In *INFOCOM, 2011 IEEE*.

[37] V. Sourlas, L. Tassiulas, I. Psaras, and G. Pavlou. Information resilience through user-assisted caching in disruptive content-centric networks. In *IFIP Networking'15*.

[38] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *SIGCOMM '02*, pages 133–145.

[39] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *ACM SIGCOMM 'ĂŹ02*.

[40] C. Tsilopoulos, G. Xylomenos, and Y. Thomas. Reducing Forwarding State in Content-Centric Networks with Semi-Stateless Forwarding. In *IEEE INFOCOM*, April 2014.

[41] G. Tyson and *et al.* . A survey of mobility in information-centric networks: Challenges and research directions. In *NoM '12*.

[42] G. Tyson and *et al.* . Juno: A Middleware Platform for Supporting Delivery-Centric Applications. *ACM Trans. Internet Technol.*, 12(2):4:1–4:28, Dec. 2012.

[43] G. Xylomenos and *et al.* . A Survey of Information-Centric Networking Research. *Communications Surveys Tutorials, IEEE*, 16(2):1024–1049, Second 2014.

[44] G. Zhang, Y. Li, and T. Lin. Caching in information centric networking: A survey. *Computer Networks*, 57(16), 2013.

[45] D. Zhou and *et al.* . Scalable, high performance ethernet forwarding with cuckooswitch. In *CoNEXT '13*, pages 97–108.