iDNS: Enabling Information Centric Networking Through The DNS

Spencer Sevilla*
spencer@soe.ucsc.edu
* UC Santa Cruz
Santa Cruz, CA

Priya Mahadevan†
priya.mahadevan@parc.com
† Palo Alto Research Center
Palo Alto, CA

J.J. Garcia-Luna-Aceves*†
jj@soe.ucsc.edu

***Abstract*—Information centric networking (ICN) architectures represent a conceptual shift from naming *end-hosts* in the Internet to naming *content* directly, and require either significant changes to the existing IP infrastructure or replacing it entirely. We present iDNS (information-centric DNS), an evolutionary path towards deploying ICN at Internet scale based on modifications to the DNS that leave the current routing infrastructure unmodified. We build and evaluate an iDNS prototype, and use it to show that iDNS achieves the benefits associated with ICN (i.e. location-independent naming, nearest-replica-routing) in a manner that both leverages current infrastructure, including content delivery protocols and caches, and supports future evolution towards other network-layer ICN architectures.**

I. Motivation and Introduction

Internet communication has shifted primarily to content distribution. Internet content continues to grow exponentially, and a large portion of this content is user-generated, to be shared with other users. This shift has prompted a substantial research effort directed towards Information-Centric Networking (ICN). This body of work [1], [2], [3], [4] proposes a "clean slate" redesign of the Internet architecture, shifting the focus from addressing *hosts* to denoting *content* in the form of named-data objects (NDOs). Although this shift to addressing content directly can provide many performance benefits in the future, the need to replace today's TCP/IP stack and routing infrastructure constitutes a significant obstacle to the adoption and deployment of ICN, given the ubiquity of IP routers and switches. Moreover, many open questions remain regarding naming, caching, routing, security, discovery, and scalability that must be solved before any one proposal is mature enough to be deployed at Internet scale.

Independently of the ICN clean-slate research thrust, significant changes and improvements have been made to Web technologies over the last decade in order to better support content distribution. These technologies (e.g., Content Delivery Networks (CDNs), caches, and proxies) were designed to alleviate some of the problems associated with content distribution and dissemination; however, they have evolved in a very ad-hoc, disjointed manner.

We believe that the Domain Name System (DNS) is the right place to develop and deploy Internet-scale ICN without requiring changes to the current routing infrastructure. On the one hand, the DNS is a primary barrier towards a more information-centric Internet because the current approach of mapping domain names to host addresses implicitly encodes location information. On the other hand, the DNS is one of the few core components of today's Internet suite that can be extended without any impact to the underlying routing infrastructure, and without significant difficulty, given that intermediate DNS servers must forward record queries and responses even if they do not recognize the record type.

This paper describes how to achieve information-centricity in the Internet by extending the DNS to store *content* names in addition to hostnames, and adapting existing content-delivery protocols or developing new applications to take advantage of these new records. Extending the DNS and accompanying protocols in this manner takes full advantage of the existing Internet routing infrastructure and core DNS servers, and can be easily deployed from the edge in an "opt-in" fashion.

This paper is organized as follows: Section II provides an overview of related work in ICN and Web technologies. Section III describes our proposed changes to the DNS, which we call information-centric DNS (iDNS). Section IV describes proposed changes to content-delivery protocols to take advantage of iDNS. Section V provides a qualitative analysis of iDNS compared to other ICN systems while Section VI analyzes the scalability of iDNS. Section VII provides experimental results obtained with a prototype implementation of iDNS, and Section VIII concludes the paper.

II. Related Work

A. Information Centric Networking

The various ICN proposals generally share a common set of *benefits*, namely: persistent and unique naming of data, efficient content-distribution, secure content provenance and authentication, and better support for network mobility, disruption, and multihoming. By the same token, ICN proposals also share a number of *characteristics*, including: routing based on content names, divorcing content names from location, ubiquitous content-caching at intermediate routers, and nearest-replica-routing [5], [6].

Distinguishing between the common *benefits* of ICN proposals and their common *characteristics* is crucial to our work, because the architecture we propose is a departure from prior ICN proposals. Some of our design decisions, such as preserving IP routing, conflict with the characteristics of prior work. However, we argue that this is acceptable because

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

the goals of information-centricity lie in the set of benefits attained, not in the methods by which they are attained. Thus, for the remainder of this document we focus on the aforementioned set of ICN benefits, rather than characteristics.

B. Efficient Content Distribution in the Web

Several Web technologies today (e.g., CDNs and caches) have evolved to help scale content distribution. Though these technologies are not information-centric, they implicitly support location-independent naming in that they serve the same data object from several locations. In this vein, HTTP itself can be considered to be information-centric in that URLs name a piece of content [8]. However, the host-name component of a URL is bound to a location in the network. Using the host name in a URL literally inhibits true location-independent naming, and is a significant obstacle in adapting these technologies to be more fully information-centric.

Directing clients to other hosts (such as CDNs or mirrors) requires DNS redirection accomplished by changing the host-name (and by extension the URL), thereby fragmenting the namespace and encoding location-dependence into a content name. Fragmenting the namespace in this manner reduces the effectiveness of content caches, because a cache has no way of identifying that a particular piece of content is duplicated across multiple URLs. Additionally, given that HTTP requests are sent directly to the IP address of a host, local caches must be placed directly along the network path and sniff all HTTP headers to provide any benefit. Qualitatively, this design forces a fundamental tradeoff between persistent content-naming and efficient content distribution. Moreover, Web technologies focus almost exclusively on these two points, and provide no mechanisms for content security (aside from securing the connection between two hosts) or support for client mobility.

III. iDNS and the Content Record

Our approach, which we call the *Information-Centric DNS* (iDNS), extends the DNS to denote *content* in addition to *hosts*, and adapts content delivery protocols to reflect this change. Such a shift clearly achieves location-independent content naming and resolves the core of the problem described in Section II-B. However, we show that this shift *also* achieves the other ICN benefits detailed in Section II-A.

At the core of iDNS is a *Content Record* (CR), which is a new type of DNS resource record that refers to a particular NDO or name prefix. Clients desiring an NDO or name prefix must first resolve the corresponding CR through the DNS, which contains the address of one or more servers hosting the content along with protocol-specific metadata necessary to fetch the content and verify its authenticity.

The CR format is illustrated in Figure 1 and contains, in addition to the standard DNS resource record fields, a field stating whether the content can be cached, fields for object and record security, a field identifying the content delivery protocol and any protocol-specific values, and a list of one or more addresses where the content can be found. The addresses are included in the response as individual DNS A{AAA} Records. These addresses are not necessarily those of the publisher or origin, but could potentially be a CDN node, alternate mirrors, or even a nearby cache.

A. Object and Record Security

The object security field in a CR can take several forms. One example is a hash value calculated over the content. Another is the public-key of the publisher, used by the client to verify a signature provided with the content object. The CR object security field enables the content *record* to secure the content *object*. However, for such a scheme to work, the CR itself must be secured. Given that the CR is just another type of DNS record, the CR may be secured through any one of several existing security approaches proposed to date, such as DNSSEC [9].

The process of *creating* a new CR must also be secured, and potentially on a much finer-grain basis than the DNS is today. For example, only Spencer should be allowed to publish CRs under the prefix /parc/videos/spencer, and only Priya under the prefix /parc/videos/priya. This fine-grained security can be accomplished through any of the current access-control techniques used by content servers supporting multiple publishers today, such as HTTP and FTP servers. These servers can provide each registered user with their own directory, typically protected by a username and password, which corresponds to a particular prefix or subtree. Given that a particular DNS zone manages the records under the zone, different domains may handle security, registration, and scalability differently without heavily impacting the DNS itself.

B. Address Record Selection

When a client successfully resolves an iDNS query for an NDO, it receives the CR and one or more address records. In the event that the client receives several address records, it must assume that the records have been ranked by the DNS for locality, availability, or some other metric, and thus should request the NDO from the first address first. Policies may arise and be standardized for address record ranking and ordering, similar to the rules specified in [10] for host IP address selection. However, the logic or complexity of such a ranking is beyond the scope of this paper, except to specify that the ranking should be assumed by the client.

A crucial part of ICN is directing clients to nearby copies of NDOs. For iDNS to support this functionality, the address of a local content server must be included in the address set, and the address set must be properly ranked to reflect this locality by the time the response reaches the client. Thus, in iDNS we allow any node along the DNS response path to add address records or reorder the records in the set, with the understanding that DNS servers closer to the client have a better understanding of the client's environment.

Though there could be several DNS servers along the return path, in practice there are typically only two: the authoritative DNS server for the CR, and the local DNS server for the client. Thus, in iDNS we expect that the local DNS server will be largely responsible for directing clients to nearby caches. This

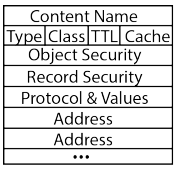
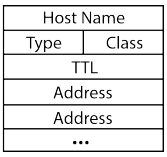


Fig. 1: Content Record vs Host Record



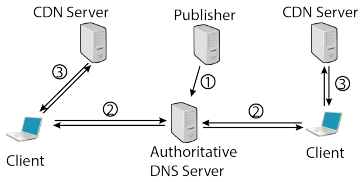


Fig. 2: Dynamic Record Generation

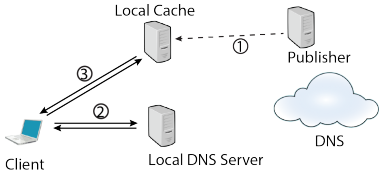


Fig. 3: Local Record Generation

approach has the added benefit that the local DNS server sees the address of the client itself, and is therefore able to perform finer-grained localization than the authoritative DNS servers that see only the address of the local DNS server. Multiple studies [11], [12] have shown that this address is only useful for coarse-grained localization, and this limits the effectiveness of CDNs powered by DNS redirection.

IV. Content Replication

Once a client receives a CR, it uses the information in the record to fetch the NDO from the most convenient site where the NDO is cached or replicated. We divide content replicas into two forms, long-lived and cached, with the difference being that long-lived replicas can generally be relied upon to provide the content, whereas caches make no such guarantees. Caches typically provide "best-effort" reliability, given that the requested content may be available, may have been evicted, or may never have been cached before.

A. Long-Lived Content Replication

A publisher may add servers, use mirroring sites, or deploy a CDN to replicate content. In contrast to the ad-hoc methods employed by content delivery protocols today, iDNS provides integrated support for such long-lived replication of content: the publisher simply contacts the authoritative DNS server for the CR and adds an address record referring to the new server hosting that content.

The authoritative DNS server for the CR may order the addresses in a certain way, or only return a subset of the addresses, based on the address of the local DNS server issuing the request. This process is illustrated in Figure 2, where the publisher registers the CR (step one), and then two clients using different local DNS servers query the same authoritative DNS server (step two) and receive different address-set orderings. Accordingly, they then request the same NDO from two different content servers (step three).

Distinguishing between *publishing* content (creating new CRs) and mirroring or serving content is important from a security standpoint. Only the owner of a prefix should be allowed to create a new CR under that prefix. However, this same restriction need not apply to parties wishing to mirror or re-host a piece of content. Content mirrors often arise out of immediate necessity [13], and sometimes the content publisher is either unaware, cannot be contacted, or does not have the necessary resources to scale up content delivery. Thus, other parties may be allowed to append their address to an existing CR without the explicit permission of the publisher. However, as long as these parties are not allowed to change the metadata in the CR, including the object security field, clients can easily identify malicious or illegitimate content. Such a restriction can be enacted through the access-control policies mentioned in Section III-A.

B. Content Caching

Caching is an equally important part of scalable content distribution. Any DNS server on the return path may be aware of a nearby content-cache, and can direct the client to this cache by simply adding the address of the cache to the address set. This process has the potential to be most effective when performed at the local DNS server, since the local DNS server knows the address of the client itself. Figure 3 illustrates this process, where after a content cache receives an NDO (step one), the local DNS server directs the client (step two) to request the NDO from this cache (step three).

V. Comparing iDNS with Prior ICN Proposals

Having provided a technical overview of iDNS, we return to the previously stated goals and benefits of ICN with the intent of showing that iDNS achieves all of the benefits of prior ICN proposals.

A. Location-Independent Persistent Naming

iDNS ensures that content names are persistent and unique through the hierarchical nature of the DNS. It also decouples names from locations by separating the CR from its set of addresses, and maintains a namespace that is not fragmented, even when content is moved or mirrored across different content servers.

There is an ongoing debate [14], [15] on whether content names should be drawn from hierarchical or flat name spaces. Works advocating flat names propose using a peripheral name resolution service (NRS) to translate between user-readable names and routable ones, and various proposals argue whether this NRS should be flat or hierarchical.

In this debate, iDNS does not advocate a particular approach over another: the CR provides a natural point of convergence for either approach. Though the DNS itself is hierarchical, flat name-resolution protocols exist [7], [16], [17] and other protocols can be designed as necessary: they must simply map a name to a CR. Moreover, the DNS itself can be adapted to a flat naming scheme, as is proposed by NetInf [4].

B. Efficient Content Distribution

A major motivation for ICN is to relieve congestion and improve content distribution through a combination of caching and nearest-replica-routing. iDNS accomplishes this goal by enabling servers along the DNS response path to change the cached address-set via the guidelines in Section III-B.

Another ongoing debate in the ICN community [5], [18], [19], [20] is the effectiveness of different caching policies, such as ubiquitous caching compared to edge caching. iDNS can enable any caching policy based on the topology of intermediate DNS servers appending cache addresses.

C. Object Level Security Model

An ICN design primitive is the concept that content can come from any location in the network. Thus, the traditional security model, which focuses on securing and authenticating *hosts*, must be changed to authenticate and secure *content* instead. iDNS preserves the concept that content may come from anywhere, and accomplishes object-level security through the security field in the CR.

Compared to other ICN proposals, an advantage of iDNS is that its trust model depends *only* on the DNS. As long as the base CR is secured, via DNSSEC or some other protocol, then the client can easily verify that the content received is legitimate. Furthermore, iDNS does not require any intermediate routers to verify the authenticity of the content. This technique avoids an open problem in the ICN community, where many questions exist regarding the trust and feasibility of a universal PKI (or other such security protocol) deployed at intermediate routers, as well as the feasibility and scalability of performing content verification at each router.

D. Mobility And Disruption

iDNS provides natural support for client mobility by focusing the content-localization logic into the local DNS resolver. When nodes leave and rejoin the network, DHCP already provides them with the address of a local DNS server, and this information is all that is necessary to localize the client. The client then sends subsequent CR requests to the new local DNS server, which directs the client to nearby caches if they exist.

E. Differences With Prior ICN Work

Architecturally, iDNS differs from other ICN proposals in one key fashion: it uses two request-response pairs, one to locate the content and the other to fetch it. This approach contrasts with other ICN proposals, which typically employ a single request-response pairing. Conceptually, this separates the act of *locating* content from the act of *distributing* it, and this split enables two separate topologies to coexist: one for content-location and the other for content-distribution. This design is a key strength of iDNS, because it effectively supports "near-replica routing" without relying on large content tables or a content-routing protocol. DNS names are simply routed swiftly without any localization or fragmentation, and then the content-request itself is routed over IP.

This split has another important ramification, in that it enables both steps in the system (content location and distribution) to evolve independently of each other, bridged only by the format of the CR. Hence, iDNS can support multiple different approaches to naming and caching, as well as a large suite of alternative content delivery protocols, including HTTP, DNS, BitTorrent, and Gnutella.

VI. Analysis of Scalability

Supporting NDO resolution through the DNS increases the number of records in the DNS by several orders of magnitude, roughly from 10^6 to at least 10^9 [21], [22]. Though the DNS is known to be a highly scalable distributed system, such a significant increase in scale merits further examination. In particular, we examine the scalability of two key parts of the system: the authoritative DNS servers in charge of storing and serving the records; and the local DNS servers in charge of forwarding queries, caching entries, and returning records to clients.

A. Scalability of the Authoritative DNS

By increasing the number of records served by the DNS, we implicitly increase the amount of (a) storage and (b) processing power necessary to serve these records. Additionally, if we increase the average name length (a likely assumption), we potentially incur additional DNS referrals.

1) Storage and Processing Power: Increasing the number of records in the DNS correspondingly increases the work necessary to store and serve these records. However, it can be qualitatively argued that a comparable amount of work is already performed today by HTTP servers. Given that today's DNS resolves nothing more than a hostname, an HTTP server must manage an entire directory tree, parse the HTTP path accordingly, and return the necessary piece of content. In contrast, DNS servers must only return a corresponding CR, not the content object itself. Even today, the performance of TLD servers (e.g., `com` or `org`) shows that a particular DNS zone *can* support thousands of entries.

Fortunately, the DNS is a well-designed, hierarchically distributed system. This design ensures that if an organization struggles to serve or update their CRs, this inadequacy is contained to the CRs of this organization and does not slow down or create problems elsewhere in the DNS. Thus, there exists a powerful and natural motive for an organization to successfully manage the publication of their content and provision adequate resources to do so.

2) Latency and Referrals: DNS requests start at the root and descend the hierarchy as necessary. For example, with no cached information, DNS resolution for www.parc.com consists of three requests: the first to the root name-server, the second to the authoritative server for com, and the last to the authoritative server for parc. Thus, as names contain more components, they necessarily result in more requests and referrals.

This design means that the behavior of DNS, in particular referrals, depends on the structure of the content name: the

same set of records may result in different behavior, depending on how their names are structured. Accordingly, to provide a meaningful analysis, we had to make assumptions about the distribution and structure of content names. In particular, we assume that the structure of content names in a DNS-based system mirrors the structure of HTTP names used today: for example, the URL www.parc.com/index.html would correspond to four iDNS zones, with the zone `www` being in charge of the CR for `index.html`. Such an assumption is safe and useful, because HTTP does not mandate the format of the path component, and the assumption enables us to draw conclusions from existing HTTP traffic.

Building on this assumption, we analyzed a large set of HTTP GETs[1]. In our analysis, we stripped out the hostname and then examined the rest of the HTTP path for the number of components. For example, a GET for parc.com/index.html would have a value of 1, whereas parc.com/videos/index.html a value of 2. Our results are shown as a histogram in Figure 4, with a mean value of 3.9 and standard deviation of 2.89.

Notably, prior analysis of DNS traffic has shown that DNS requests and referrals are largely mitigated through local DNS caching. Jung et al [23] observe that the average DNS query results in 1.2 *referrals* and a latency of approximately 60ms, despite the fact that the average DNS name has 3.3 *components*. These results are encouraging, because they illustrate the effectiveness of caching in improving DNS performance.

Based on the above results, we believe that caching and other optimizations used for host-name resolution with the DNS will be equally successful when extended to content objects. When deployed at a large scale, we expect the average name to consist of ~ 6.8 components, and to result in ~ 2.4 referrals and an average latency of ~ 100ms, all of which are acceptable values.

B. Scalability at the Local DNS Server

iDNS increases the work required by the local DNS server, because it must direct clients to nearby content-caches. In its simplest form, redirection is accomplished by appending an address to the address set of each DNS response. This operation, which must happen for each request/response pair, constitutes less work than a transparent cache carries out today when it inspects HTTP headers. More complex schemes for cache load balancing may evolve, but such schemes represent a fundamental tradeoff between additional complexity at the local DNS server or decreased efficiency at the content cache. This tradeoff is important to highlight, because such a tradeoff can only be examined and optimized for a particular local topology and set of hardware, yet we show that iDNS provides support for such optimizations.

Given that local DNS servers often cache records to improve DNS performance, increasing the number of DNS records can have an adverse affect on the local cache. However, multiple studies [23], [24] indicate that the DNS cache size is not a limiting factor on performance, because the distribution of DNS objects is Zipf, and the individual record objects are quite small. In fact, DNS objects are so small that the common DNS caching utility `dnscache` provides a default cache size of 1MB and a *maximum* cache size of 16MB! Thus, there is ample room for DNS caching to expand by several orders of magnitude before an impact on performance is felt.

[1]One day (2012-11-01 00:00∼23:59) of HTTP traffic initiated by hosts at POSTECH University in South Korea, approximately 25 million requests

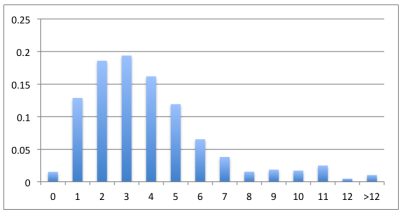


Fig. 4: Histogram of HTTP Path Components

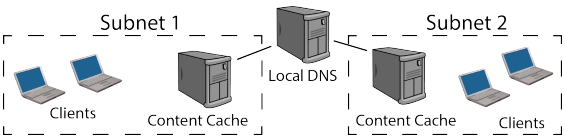


Fig. 5: Prototype Deployment Topology

VII. Experimental Deployment

To explore a common deployment scenario, we built a prototype iDNS system that employs hierarchical CCN-style naming, edge-caching through local DNS servers, and HTTP for content delivery. We wrote a simple iDNS client, local DNS resolver, and content cache in Java, and deployed the code (approximately 2000 lines, of which only 200 are unique to iDNS) across three servers and four clients at PARC configured in the topology shown in Figure 5. Both subnets use the same local DNS server, which directs clients to their closest cache based on their address; the primary difference between the two subnets is that the cache in Subnet 1 is directly along the network-path from the clients to the Internet, whereas this is not the case for Subnet 2. To avoid changing the authoritative DNS server, we elected to encode CRs as a TXT record starting with `"CR:"`.

A. Name Format Translation

We start with a hierarchical content name (e.g., /parc/csl/papers/idns.pdf), which is translated to a DNS-resolvable name through a simple algorithm: First, reverse the order of all names broken by the / character to create the string idns.pdf/papers/csl/parc/. Next, swap each / character for a `.`, and each `.` for a /, to create the valid[2] DNS name idns/pdf.papers.csl.parc. This simple translation is one-to-one and reversible, which allows the DNS name to later be reconstructed into the original hierarchical content name.

To support HTTP, we define a *hostname length number* (HLN) to be included with the CR. The HLN is used to translate the content name from DNS to HTTP; this is necessary, given that HTTP URLs contain two hierarchical components,

[2]DNS explicitly prohibits use of the "/" character in *hostnames*, but allows it in other record types, such as TXT or our CR.

Cache Policy	None	Transparent	iDNS
Client Latency	$\mu = 45ms$ $\sigma = 5.03ms$	$\mu = 26ms$ $\sigma = 2.71ms$	$\mu = 29ms$ $\sigma = 3.45ms$
Cache Latency	N/A	$\mu = 52ms$ $\sigma = 4.47ms$	$\mu = 61ms/32ms$ $\sigma = 4.67ms/2.94ms$

TABLE I: Results

the hostname and the path. Thus, the HLN is needed to denote the number of components in the hostname, with the assumption that the remainder of the name is the content path. For example, when $HLN = 2$, the DNS name idns/pdf. papers.csl.parc.com translates to parc.com/csl/papers/idns.pdf, whereas $HLN = 3$ would create csl.parc.com/papers/idns.pdf. Once translated, the client then issues an HTTP GET request to the top address returned by the local DNS server.

B. Latency Results

We hosted a 456KB copy of this paper on a server at UCSC, and created a CR naming it as edu/ucsc/soe/ccrg/idns.pdf. We then had each client in our test topology request the file 10 times, using three different caching schemes: first without caching, second only using transparent caching along the network path, and third using iDNS cache location to explicitly address the same cache.

The first row of Table I shows the average latency (μ) and variance (σ) of the HTTP transfer, as perceived by the end client. As expected, these results show that a cache-hit reduces latency as compared to fetching the object from the origin; however, they *also* show that there is minimal performance difference between transparent in-line caching and our method, which directly addresses the cache itself and includes the origin addresses as an HTTP header option. This result is important to our design because it enables clients to take advantage of caches existing outside of the direct network-layer path to the server.

The second row contains the average time needed to populate the cache itself the first time the file is requested. When requesting a file from the origin, iDNS exhibits slightly more overhead compared to transparent caching (61ms to 52ms); this overhead is the natural result of coordinating two separate HTTP requests as opposed to simply sniffing and copying data. However, the second entry under iDNS (32ms) shows an interesting observed behavior: the first time an iDNS cache requests the file, it must be served from the *origin* server at UCSC, yet when the second cache requests the same file, it can locate and request it directly from the first iDNS cache. This behavior results in lower latency as well as distributing the load off the origin server.

VIII. Conclusion

iDNS is an evolutionary step towards development and deployment of ICN architectures. We show that iDNS maintains compatibility with existing approaches to routing and content-delivery, and requires only minor changes to end clients. This compatibility means that iDNS can be deployed *today*, yet it can be extended to support future developments (e.g., content routing and content security) in other ICN architectures as they mature. Our analysis of DNS and HTTP shows that iDNS can be deployed at Internet scale, and our prototype deployment shows that iDNS achieves the benefits of ICN without incurring any significant processing or control overhead.

References

[1] T Koponen, M Chawla, B G Chun, A Ermolinskiy, K H Kim, S. Shenker, and I Stoica. A data-oriented (and beyond) network architecture. 37(4):181–192, 2007.

[2] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12, 2009.

[3] N Fotiou, P. Nikander, D Trossen, and G C Polyzos. Developing information networking further: From PSIRP to PURSUIT. *Broadband Communications, Networks, and Systems*, pages 1–13, 2012.

[4] C Dannewitz. NetInf: An information-centric design for the future internet. 2009.

[5] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce M Maggs, K C Ng, Vyas Sekar, and Scott Shenker. Less Pain, Most of the Gain: Incrementally Deployable ICN. In *Proceedings of SIGCOMM 2013*, page 1. ACM, 2013.

[6] B Ahlgren, C Dannewitz, C Imbrenda, D Kutscher, and B Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.

[7] K Katsaros, N Fotiou, X Vasilakos, C Ververidis, C Tsilopoulos, G Xylomenos, and G Polyzos. On inter-domain name resolution for information-centric networks. *NETWORKING 2012*, pages 13–26, 2012.

[8] L Popa, A Ghodsi, and I Stoica. HTTP as the Narrow Waist of the Future Internet. page 6, 2010.

[9] S. Weiler and D. Blacka. RFC 6840: Clarifications and Implementation Notes for DNS Security (DNSSEC). *IETF Standard*, 2013.

[10] Richard Draves. Default Address Selection for Internet Protocol version 6 (IPv6). 2003.

[11] Z.M. Mao, C.D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang. A precise and efficient evaluation of the proximity between web clients and their local DNS servers. *USENIX Annual Technical Conference*, pages 229–242, 2002.

[12] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the effectiveness of dns-based server selection. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1801–1810. IEEE, 2001.

[13] The Slashdot Effect. http://en.wikipedia.org/wiki/Slashdot_effect.

[14] A Ghodsi, T Koponen, J Rajahalme, P Sarolahti, and S. Shenker. Naming in content-oriented architectures. pages 1–6, 2011.

[15] M F Bari and S Rahman Chowdhury. A survey of naming and routing in information-centric networks. *Communications . . .*, 2012.

[16] V. Ramasubramanian and E.G. Sirer. The design and implementation of a next generation name service for the Internet. *ACM SIGCOMM Computer Communication Review*, 34(4):331–342, 2004.

[17] Tam Vu, Akash Baid, Yanyong Zhang, Thu D Nguyen, Junichiro Fukuyama, Richard P Martin, and Dipankar Raychaudhuri. Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet. pages 698–707, 2012.

[18] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[19] G Xylomenos, X Vasilakos, C Tsilopoulos, V A Siris, and G C Polyzos. Caching and mobility support in a publish-subscribe internet architecture. *Communications Magazine, IEEE*, 50(7):52–58, 2012.

[20] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks. pages 27–40, 2012.

[21] Diego Perino and Matteo Varvello. A reality check for content centric networking. pages 44–49, 2011.

[22] We Knew The Web Was Big... http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html.

[23] J. Jung, E. Sit, H Balakrishnan, and R Morris. DNS Performance and the Effectiveness of Caching. *Networking, IEEE/ACM Transactions on*, 10(5):589–603, 2002.

[24] Edith Cohen and Haim Kaplan. Proactive caching of DNS records: addressing a performance bottleneck. *Computer Networks*, 41(6):707–726, April 2003.