Less Pain, Most of the Gain: Incrementally Deployable ICN

Seyed Kaveh Fayazbakhsh[*], Yin Lin[†], Amin Tootoonchian[○,±], Ali Ghodsi[‡,∓]
Teemu Koponen[¶], Bruce M. Maggs[†,◇], K. C. Ng[◇], Vyas Sekar[*], Scott Shenker[‡,±]

[*]Stony Brook University, [†]Duke University, [○]University of Toronto
[±]ICSI, [‡]UC Berkeley, [∓]KTH, [¶]VMware, [◇]Akamai

ABSTRACT

Information-Centric Networking (ICN) has seen a significant resurgence in recent years. ICN promises benefits to users and service providers along several dimensions (e.g., performance, security, and mobility). These benefits, however, come at a non-trivial cost as many ICN proposals envision adding significant complexity to the network by having routers serve as content caches and support nearest-replica routing. This paper is driven by the simple question of whether this additional complexity is justified and if we can achieve these benefits in an incrementally deployable fashion. To this end, we use trace-driven simulations to analyze the quantitative benefits attributed to ICN (e.g., lower latency and congestion). Somewhat surprisingly, we find that pervasive caching and nearest-replica routing are not fundamentally necessary—most of the performance benefits can be achieved with simpler caching architectures. We also discuss how the qualitative benefits of ICN (e.g., security, mobility) can be achieved without any changes to the network. Building on these insights, we present a proof-of-concept design of an incrementally deployable ICN architecture.

Categories and Subject Descriptors: C.2.6 [Computer-Communication Networks]: Internetworking

Keywords: Internet architecture, information-centric networking

1. INTRODUCTION

While the idea of information- or data-centric networking has been around for over a decade [5,21,41,46], there has been renewed interest in the last five years, inspired in large part by the CCN project [23]. This interest is evident on many fronts: recent papers on this topic, several workshops and conferences, and interest from telcos and equipment vendors (e.g., [9, 29]). Furthermore, there are several future Internet architecture proposals that incorporate or support ICN as a core building block (e.g., DONA [28], NDN [24], XIA [22], 4WARD [2], SAIL [37], and COMET [13]).

This departure, both in the earlier and more recent work, from the host-centric paradigm is motivated by the evolution of Internet traffic workloads. While the specific drivers have changed—e.g., Baccala's work was motivated by web traffic [5] while more recent work points to the growth of video traffic—the core tenet of various ICN proposals has not changed. The insight here is that a user's intent is to fetch some *data object* rather than connect to a specific host. By decoupling the data a user wants to access from how the data is delivered, ICN promises several natural benefits. These include: lower response time via pervasive caching and nearest-replica routing; intrinsic content integrity without external network-level indicators (e.g., HTTPS); simplified traffic engineering; and better support for mobility (e.g., [1, 18, 27, 49]).

Unfortunately, these benefits come at a non-trivial cost. Many ICN proposals envision significant upgrades to the entire network infrastructure requiring all end hosts and network routers to support ICN as a first-order primitive. This entails adding content stores to routers and supporting routing on content names as opposed to IP addresses. Given that some of the ICN proposals mandate wholesale changes to the network infrastructure, it is natural to ask if this complexity is worthwhile. Specifically, we ask:

- Does ICN provide significant benefits?
- If so, can we achieve the same benefits in a more incrementally deployable fashion within the scope of today's available mechanisms?

In order to address these questions, we begin by breaking down the potential benefits of ICN into two categories. The first class of *quantitative* benefits—lower response time and simplified traffic engineering—arise from a combination of a pervasive caching infrastructure coupled with intelligent nearest-replica routing. The second class of *qualitative* benefits stem from the ability to name content and verify content integrity through the naming scheme (e.g., self-certified names or digital signatures).

Having thus bisected the potential benefits, we first focus on the quantitative benefits. Rather than commit to any specific realization, we analyze a broad spectrum of ICN architectures along two key dimensions: *cache placement* (e.g., edge caches vs. pervasive caching) and *routing* (e.g., shortest path to origin servers vs. nearest replica routing). Using trace-driven simulations, we find that:

- On realistic request traces, the maximum performance gap between a simple *edge-based caching* architecture and a full-fledged ICN architecture (i.e., with pervasive caches and nearest-replica routing) is at most 9% with respect to response time, network congestion, and origin server load.
- Nearest-replica routing adds marginal (2%) value over simple shortest-path routing in ICN (on all metrics).
- Using sensitivity analysis on a range of configuration parameters, we find that the *optimistic best-case* improvement that ICN can provide is 17% over the simple edge-caching architecture (on all metrics).

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.

- Simple extensions to edge caching (e.g., doubling the cache size or enabling local cooperation) reduces even the optimistic best-case performance gap to less than 6% (on all metrics).

Note that we are not arguing that caching is not useful for typical workloads. Rather, our observation is that exploiting the benefits of cacheable workloads is far easier than we imagined. The quantitative benefits of caching largely arise from the fact that *some* cache exists; pervasive caching and nearest-replica routing add little value for the types of heavy-tailed workloads we expect in practice. In some sense, our work reconfirms (and extends) past results from the web caching literature to the ICN context (e.g., [7]).

Motivated by these findings, we analyze whether the remaining qualitative benefits can be achieved in an incrementally deployable fashion without router-level support. Somewhat surprisingly, we show that many of these benefits can be achieved using techniques that are already well known in the content distribution community. Building on these insights, we provide a reference design of an *incrementally deployable ICN* architecture or *idICN*. idICN is an application-layer ICN architecture that delivers most of the perceived benefits of ICN in a backwards-compatible fashion, without requiring any network layer support.

In some sense, this paper is an attempt to analyze the ICN literature from an end-to-end perspective—significant changes to the network can be justified only if they offer substantial performance improvements [38]. We find that most of the benefits, even those that seem to require changes to the core network infrastructure, can be achieved in an end-to-end fashion (i.e., implemented at the edge of the network).

2. BACKGROUND AND MOTIVATION

In this section, we begin with a brief overview of the common themes underlying different ICN proposals [2, 23, 24, 37]. Then, we use real request logs to motivate the need to revisit some of the assumptions about pervasive caching and nearest-replica routing.

2.1 ICN Principles and Benefits

While ICN proposals vary in terminology, implementation, and APIs to clients and network operators, we identify four main themes underlying all proposals:

1. *Decoupling names from locations:* Network applications and protocols are rearchitected so that communication is based on content lookup and transfer in contrast to today's host-centric abstractions.
2. *Pervasive caching:* In the limit, every network router also acts as a content cache. This means that in addition to traditional forwarding responsibilities, routers also serve requests for content in their caches.
3. *Nearest replica routing:* Routing is based on content names rather than hosts so that requests are routed to the nearest copy of the content. (In the worst case, this is the origin server hosting the content.)
4. *Binding names to intent:* The content name is intrinsically bound to the intent of the content publisher and the consumer. This binding helps users (and routers) to check the integrity and the provenance of the data without external indicators.

The proposals differ in specific mechanisms they use to achieve these properties and the specific API they expose [18]. For instance, some proposals prefer opaque identifiers [28] while others use human-readable hierarchical naming [23]. Our goal in this paper is not to focus on the specific ICN architectures; rather, we want to analyze the benefits arising from the principles underlying ICN.

Benefit	Feature			
	Decoupling names from locations	Pervasive Caching	Nearest-replica routing	Intrinsic Binding
Latency (§4, §5)		✓	✓	
Traffic Engg. (§4, §5)		✓	✓	
Mobility (§6)	✓		✓	
Ad hoc mode (§6)	✓		✓	
Security (§6)	✓			✓

Table 1: *Feature-Benefit Matrix for ICN: the ✓ shows the key features of ICN that contribute to each perceived benefit.*

Benefits: For completeness, we enumerate the perceived benefits of ICN that have been argued in prior work (e.g., [1, 18]):

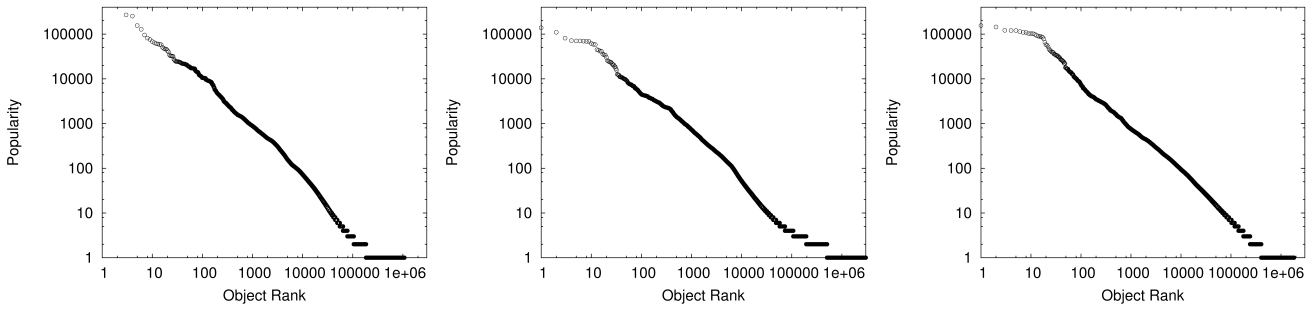
- *Lower response latency:* A pervasive caching infrastructure means that the requests do not necessarily need to traverse the entire network toward the origin server.
- *Simplified traffic engineering:* Caching also helps network operators by automatically eliminating content hotspots, which simplifies the traffic engineering logic necessary to balance network load.
- *Security:* By elevating content as a first-class citizen, ICN intrinsically binds the user's intent to the eventual data being delivered without having to rely on external confirmation of the provenance or authenticity of the data.
- *Mobility:* Shifting from host- to content-centric routing also makes it easier to support mobile clients, as traditional problems with handoffs, retransmissions, etc., simply go away.
- *Ad hoc mode:* Another benefit of ICN is the ability of nodes to communicate and share content without any infrastructure support. Imagine a user wanting to share a photo between a mobile phone and a laptop; today we have unwieldy workarounds via cloud-based services [47]. Further imagine that they are in an airplane without a wireless network; in this case, they cannot share the content because they do not have IP working.
- *Others:* There are other perceived benefits such as DDoS resilience [16] and disruption tolerance that are less well explored. These appear to be specific instances or combinations of the above benefits. For instance, disruption tolerance seems to be a combination of support for mobility and ad hoc mode. Similarly, DDoS resilience stems from avoiding content hotspots and universal caching.

Table 1 summarizes the benefits and the ICN principles contributing to each perceived benefit. We can see that the *quantitative* performance benefits—low latency and traffic engineering—essentially arise as a result of the pervasive caching and nearest-replica routing infrastructure envisioned by ICN solutions. Unsurprisingly, we find that this is also the topic that has received the greatest attention in the ICN community.[1] The second class of *qualitative* benefits such as mobility, security, and support for ad hoc mode are rooted in the naming-related aspects of ICN (and to a lesser degree from nearest-replica routing).

2.2 Motivation: Heavy-Tailed Workloads

Many measurement studies have observed heavy-tailed or Zipf distributions (i.e., the i^{th} popular object has a request probability proportional to $\frac{1}{i^\alpha}$ for some $\alpha > 0$) in request popularities (e.g., [7, 20]). In this section, we use request logs collected from three CDN

[1]For instance, in the most recent ICN workshop at SIGCOMM 2012, roughly half the papers were related to caching in ICN.



(a) US (b) Europe (c) Asia

Figure 1: *Request popularity distribution across different geographical locations. While the specific exponent parameters vary slightly across the different locations, we can see that the popularity distribution is Zipfian.*

Location	Requests	Zipf parameter
US	1.1M	0.99
Europe	3.1M	0.92
Asia	1.8M	1.04

Table 2: *Analysis of requests from three CDN cache clusters in different geographical regions.*

vantage points to reconfirm such heavy-tailed behavior in recent workloads.

Dataset: The CDN serves a diverse workload spanning diverse content types: regular text, images, multimedia, software binaries, and other miscellaneous content. We use daily request logs from three geographically diverse locations. Each log entry contains four relevant fields: an anonymized client IP, anonymized request URL, the size of the object, and whether the request was served locally or forwarded to a remote location.

Figure 1 visually confirms that request popularity is heavy-tailed and close to a Zipfian distribution; each curve is almost linear on a log-log plot. While the specific exponents and y-intercepts do vary slightly across locations and content types, the main takeaway is that object requests are reasonably approximated by heavy-tailed Zipfian distributions. Table 2 summarizes the Zipf-fit parameters for the three locations that we use to guide our simulation study.

Why does Zipf matter? Anecdotal evidence suggests that in the presence of Zipf workloads, having multiple caching layers or cooperative caching provide limited improvements [6, 52]. To understand this better, we begin with a simple analysis on a binary tree topology. We use an analytical optimization model to reason about the *optimal* cache management scheme—the best static placement of objects across the tree nodes given a Zipfian workload. The workload is a collection of requests, each arriving at a leaf of the tree chosen at random. Given a request, as long as the current node does not have the object, the request is forwarded to the parent node. The root is assumed to host all objects. As a simplifying assumption, we assume all caches are of the same size.[2]

A tree is small enough to be amenable to such analysis. At the same time a tree is instructive because from the view of a content origin server, the distribution topology is effectively a tree.

[2]We do not show the full formulation for brevity. The high-level idea is to solve the problem of deciding where to cache specific objects and how to assign requests to different caches to minimize the expected latency (i.e., number of hops traversed by requests) as an integer linear program.

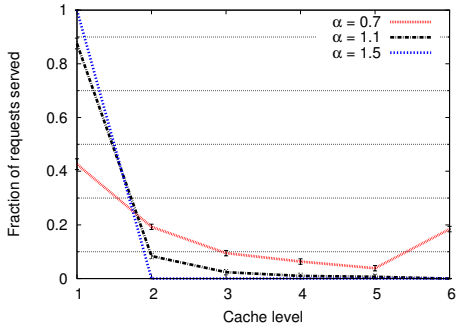


Figure 2: *Utility of different cache levels with a simplified optimization model on a binary tree with 6 levels. Level 6 here is the origin server to which requests are sent on cache misses.*

Figure 2 shows the fraction of requests served at each level of the tree for different request distributions. Here, level 6 denotes the origin server. We see that the intermediary levels of the tree (i.e., levels 2–6) add little value beyond caching at the edge or satisfying the request at the origin. Consider the setting with $\alpha = 0.7$. In this case, the expected number of hops that a request traverses is $0.4 \times 1 + \ldots + 0.18 \times 6 \approx 3$. Now, let us look at an extreme scenario where we have no caches at the intermediate levels; i.e., all of the requests currently assigned to levels 2–6 will be served at the origin. In this case, the expected number of hops will be $0.4 \times 1 + 0.6 \times 6 = 4$. In other words, the latency improvement attributed to universal caching is only 25%. Note that this is actually unfair to the edge caching approach, as it only has half the total cache capacity.

We also extended this optimization-driven analysis with another degree of freedom, where we also vary the sizes of the cache allocated to different locations. The results showed that the optimal solution under a Zipf workload involves assigning a majority of the total caching budget to the leaves of the tree. (We do not show the detailed results due to space limitations.)

The above reconfirmation that request workloads are Zipf and our simple tree-based intuition motivate us to evaluate to what extent pervasive caching and nearest-replica lookup are really necessary to achieve the quantitative benefits of ICN.

3. DESIGN SPACE FOR CACHING

The measurements and simplified analysis from the previous section raise the question of whether pervasive caching and nearest-replica routing are strictly necessary. We do not claim novelty for

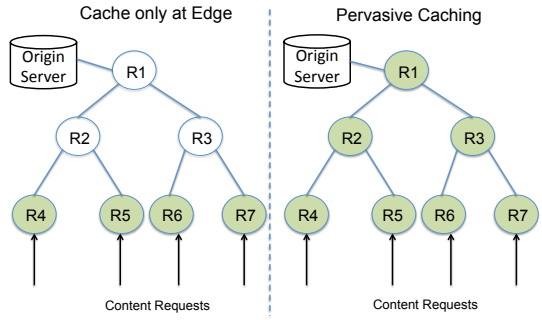


Figure 3: *Example of two cache placement strategies: caches placed at select network locations such as at the edge of the network or pervasively throughout the network. The shaded nodes are routers augmented with content caches while the others are traditional IP routers.*

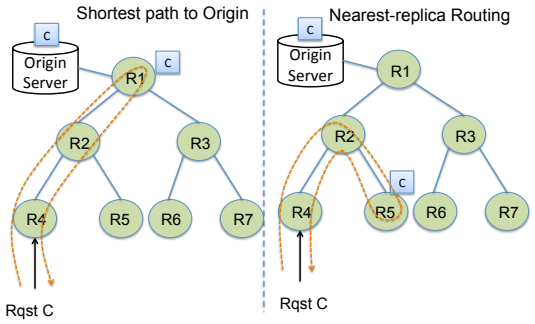


Figure 4: *Example of two request routing strategies: requests are routed along the shortest path to the origin server and served from some available content cache along that path or the requests are routed to the nearest cached copy (e.g., ICN).*

the general observation that ubiquitous caching may have limited impact with Zipfian distributions. Our specific contribution here lies in providing a detailed analysis of caching in an ICN-specific context, which involves an entire network of caches and name-based forwarding, and comparing it with more easily deployable alternatives.

Given the diversity of ICN proposals, we want to avoid tightly coupling our analysis to any specific architecture. To this end, we consider a broad design space of caching infrastructures characterized by two high-level dimensions:

1. **Cache placement:** The first dimension of interest is where caches are located in the network. From the perspective of the origin server serving content to users, the network looks like a tree of routers/caches. Figure 3 depicts two possible strategies in this distribution tree. At one extreme, every network router is also a content cache. Alternatively, we can envision caches deployed close to the network edge. We can also consider intermediate placement solutions; e.g., due to economic constraints operators may only install caches at locations that serve to sufficiently large populations [29]. A related question here is provisioning the compute and storage capacity of the various caches. For instance, we can consider a network where all caches have the same capacity or make the caches proportionally higher for nodes serving larger populations.
2. **Request routing:** An orthogonal dimension to placement is how content requests are routed through the network. As representative samples, we consider two design points in Figure 4. In this example, a request for the object C arrives at node $R4$. The origin server and possibly some other nodes have copies of C. In the first case, a request is routed along the tree toward the origin server until it finds a node with the desired content. In the second case, we assume that the network routes the request based on the name toward the closest replica. We can also consider intermediate strategies. For instance, we can consider cooperative caching within a small search scope to look up nearby nodes and reverting to shortest-path routing toward the origin if these lookups fail.

In this paper, we are less concerned with the discovery protocols used to populate content routing tables [23] or the feasibility of name-based lookup in high-speed routers [34]. Since our goal is to evaluate the *potential benefits* of pervasive caching and nearest-replica routing, we conservatively assume that routing and lookup have zero cost.

There is possibly a third aspect of *cache resource management*. Given that prior work (e.g., [39]) and our own experiments show that the LRU policy performs near-optimally in practical scenarios, we use LRU for the rest of this paper. We also tried LFU, which yielded qualitatively similar results.

4. BENEFITS OF CACHING

In this section, we use simulations to analyze the relative performance of different caching architectures with respect to three key metrics: (1) *response latency*; (2) *network congestion*; and (3) *server load*.

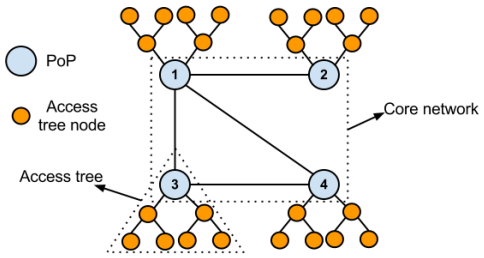


Figure 5: *An example network topology with four PoP nodes and their corresponding access trees.*

4.1 Setup

We use PoP-level network topologies from educational backbones and Rocketfuel [43]. From each PoP-level topology (*core network*), we create its corresponding router-level topology by considering each PoP as the root of a complete k-ary tree [43]. We refer to this as the *access tree*. The baseline results presented in this section use k=2 and set the depth of each access tree to 5. We study the sensitivity of the results to these parameters in Section 5. Figure 5 shows an example network topology with four PoPs. We annotate each PoP with the population of its associated metro region and assume that the requests at each PoP are proportional to its population. We assume a homogeneous request stream where requests at different network locations are drawn from the same object popularity distribution—we analyze the effect of popularity spatial skew in the next section.

Requests arrive at the leaves of each access tree. Within each PoP, the requests arrive uniformly at random at one of the leaf nodes of that access tree. Each PoP additionally serves as an *origin server* for a subset of the set of entire objects; the number of objects it hosts is also proportional to the population.[3] We assume that each

[3]We also experimented with other models such as uniform origin assignment and found consistent results.

cache has sufficient budget (i.e., storage capacity) to host a certain number of objects. We use different *budget configurations*; e.g., uniform or proportional to the population. Note that a PoP node serves two roles: (1) as the root node of an access tree, and (2) as the origin server for a set of objects. As a regular cache, we assume the PoP node has a fixed budget, but as an origin server, we assume it has a very large cache to host all the objects it "owns".

Representative designs: We choose four representative designs from the design space described in Section 3:

- **ICN-SP**: This assumes pervasive cache placement and shortest path routing toward the origin server. That is, any cache along the shortest path may respond to the request if it has the object.
- **ICN-NR**: This extends ICN-SP with nearest-replica-based routing. Our goal here is not to design new routing strategies or evaluate the overhead of these content-based routing protocols. We conservatively assume that we can find and route to the nearest replica with zero overhead.
- **EDGE**: This is the simplest strategy where we only place caches at the "edge" of the network. The notion of "edge" depends on other economic and management-related factors and whether it is viable to operate caches deep inside the network. We use edge to represent the leaves of our access topology since our goal is to do a relative comparison between the different schemes.
- **EDGE-Coop**: This uses the same placement as EDGE, but with a simple neighbor-based cooperative strategy. Each router does a scoped lookup to check if its sibling in the access tree has the object, and if so, reroutes the request to the sibling.

Cache provisioning: We consider two cache budgeting policies for setting the cache size B_r for each router r. If there are a total of O objects being requested across the network of R routers, we assume that the total cache budget of the network is $F \times R \times O$, for some value of $F \in [0, 1]$. As a baseline, we pick $F = 5\%$ based roughly on the CDN provisioning we observe relative to the universe of objects each CDN server sees in a day. We vary the budget parameter in the next section.

Given this total budget, we consider two possible splits:

1. *Uniform:* Each router r gets a fixed cache capacity to store 5% of the universe of all objects.
2. *Population-proportional:* We divide the total budget such that each PoP gets a total budget proportional to its population and then divide this budget equally within that access tree.

We have also tried other cache budgeting policies and observed results that are qualitatively consistent. Due to space constraints, we do not report the results from those settings.

Note that this method of dividing the budget can be viewed as unfair to the EDGE and EDGE-Coop settings as they have a total budget that (for binary trees) is half the capacity of the ICN-SP and ICN-NR cases. Thus, we also consider a new representative design **EDGE-Norm** where we ensure that the total budgets are the same. That is, we take the EDGE configuration and multiply the budget of the edge caches by an appropriate constant (for example, 2 in case of binary trees) to make sure that the total cache capacity is the same across different representative designs.

All representative designs use LRU for cache management. Each node on the response path, which starts at the node that the requested object is found (the origin server or a cache) and ends at the leaf at which the request has arrived, stores the object in addition to forwarding it towards the client.

For reasons of scalability, we use a request-level simulator and thus we do not model packet-level, TCP, or router queueing effects. Since our goal is to understand the relative performance of the different caching architectures at a request granularity, we believe this is a reasonable assumption. We optimistically assume that ICN-SP and ICN-NR solutions incur no lookup or discovery overhead when modeling the response latencies and network congestion.

Having described the simulation setup, we present the baseline results in the next sub-section.

4.2 Baseline Results

We use trace-driven simulations using the CDN request logs and corresponding synthetic request logs, which have similar numbers of requests, objects, and the best-fit Zipf popularity distribution.

For this section, we use the Asia trace from the CDN. We assume that this trace is the universe of all requests. We assign each request to a PoP with a probability proportional to the corresponding PoP's population. (We vary the popularity skew across PoPs in Section 5.) Within each PoP, requests are uniformly distributed among the leaves.

For the following results, we report normalized metrics w.r.t. to a system without any caching infrastructure. Thus, we focus on the *improvement* in response latency, reduction in network congestion, and reduction in server load. In each case, a higher value of the metric implies that caching is more beneficial.

Response latency: We report response latency in terms of the *number of hops* between the request and the location from which it was served. Figure 6(a) shows the percentage improvement in latency for the four caching architectures (plus EDGE-Norm) in comparison with a network with no caching (i.e., all requests are routed to the origin PoP). We make three main observations. First, the gap between the different caching architectures is quite small (at most 9%); this is consistent across the different topologies. Second, EDGE-Coop consistently achieves comparable latency improvement relative to ICN-NR with a maximum gap of 3%. Third, nearest replica routing (ICN-NR) does not offer significant benefits over ICN-SP.

Figure 7(a) shows the latency improvements for the case of uniform budget assignment across PoPs. We see no major change in the relative performances of the different architectures.

Network congestion: Other parallel work has focused on the interaction between ISP traffic engineering and "content engineering" and showed that there are natural synergies to be exploited here [25,35]. Here, we focus on a simpler question of network congestion under different caching architectures. The congestion on a link is measured simply as the number of object transfers traversing that link.

Figure 6(b) shows the effectiveness of caching in reducing the congestion level across the network. We focus on the maximum congested link in the network. Analogous to the query delay analysis, the percentage shown in each case indicates the improvement over the base case with zero budget. Once again, we see that EDGE-Coop delivers close to the best performance (with a maximum gap of 4%) and that the gap between the solutions is fairly small.[4] The success of edge-based approaches in this context is particularly promising. Unlike nearest-replica routing, caching at the edge strictly reduces traffic in the core of the network and thus eliminates any concerns that ISP traffic engineering and content engineering could be in conflict [25]. Figure 7(b) shows similar results with uniform budget assignment across PoPs.

[4]The absolute improvement values for latency are typically lower than the numbers for the congestion improvement. The reason is that we are looking at the *average* in the latency metric and the *maximum* in the case of congestion.

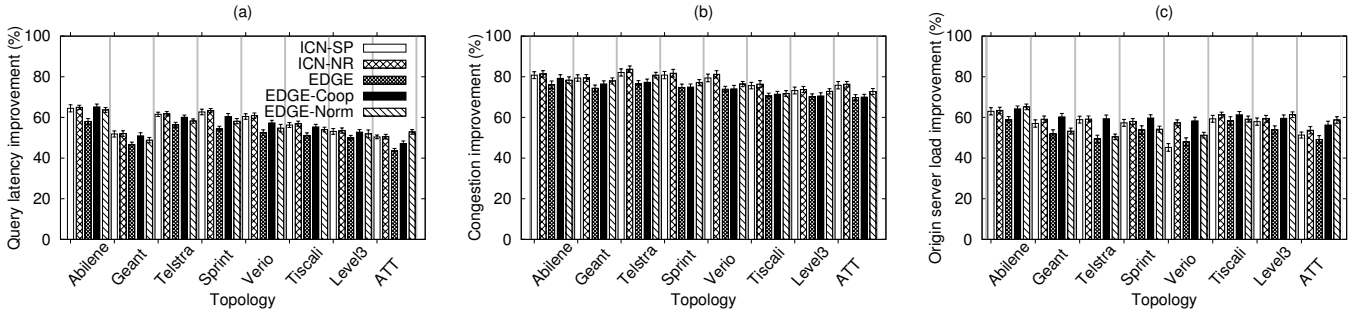


Figure 6: *Trace-based simulations results. Cache budget and origin server allocation are set to be proportional to population. Parts (a), (b), and (c) show improvements in query latency, congestion, and maximum origin server load, respectively.*

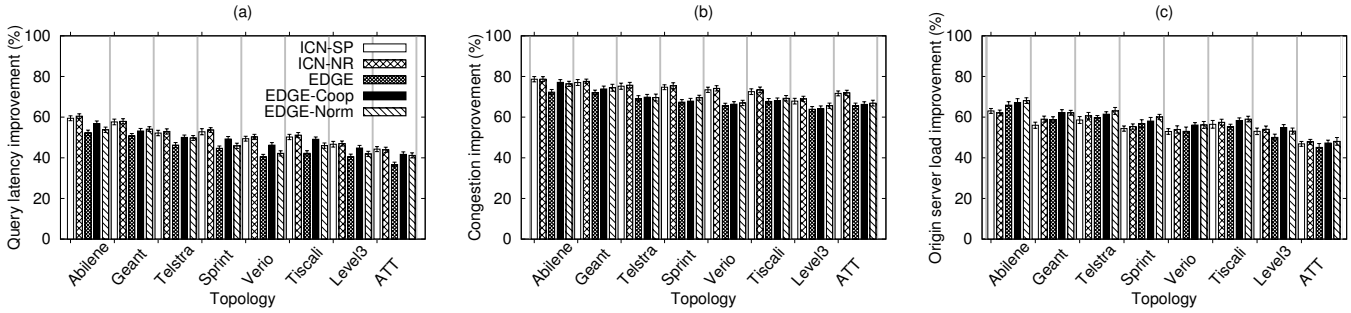


Figure 7: *Trace-based simulations results. Cache budget and origin server allocation are set to be uniform across the network. Parts (a), (b), and (c) show improvements in query latency, congestion, and maximum origin server load, respectively.*

Server load: Next, we consider the load on the origin servers (i.e., the PoP nodes hosting the objects) in Figure 6(c). The metric we use here is the percentage reduction in the requests served by the origin server with the highest observed load in the network (again, over the base case of no caching). Once again, we see that the various cache architectures show similar performance: a maximum performance gap of 9% between EDGE-Coop and ICN-SP and a 2% gap between ICN-NR and ICN-SP. Figure 7(c) shows similar results with uniform budget assignment as well.

Validating a synthetic request model: Ideally, we would like to vary the request popularity distribution. One concern is whether the performance gaps using synthetic request traces are comparable to real traces. That is, in addition to visually and statistically confirming the distribution fit in the previous section, we want to ensure that this translates into system-level performance metrics.

To address this issue, for each request trace, we also generate a synthetic request log with the best-fit Zipf distribution. In Table 3 we show the difference between trace-driven and synthetic request data-driven simulations w.r.t. the performance gap between ICN-NR and EDGE. The predicted gap of ICN-NR over EDGE in different topologies (see Table 3) has a maximum value of 1.67%. The gap w.r.t. congestion and origin server load improvements are similar and not shown for brevity. These results suggest that using a Zipf-based synthetic log is a reasonable approximation for a real trace.

	Performance gap between ICN-NR and EDGE		
Topology	**Trace**	**Synthetic**	**Difference**
Abilene	6.89	7.81	0.92
Geant	5.92	6.96	1.04
Telstra	7.44	8.63	1.19
Sprint	7.09	8.76	1.67
Verio	7.40	8.94	1.54
Tiscali	7.11	8.05	0.94
Level3	6.18	7.32	1.14
ATT	7.25	8.04	0.79

Table 3: *Comparison of simulation results for query latency on request traces and synthetic data (with best-fit Zipf).*

4.3 Key Observations and Implications

In summary, we make three key observations:

- The performance gap between different caching policies on all three metrics (i.e., query latency, congestion, and server load) is small (at most 9%).
- The performance gap between ICN-SP and ICN-NR is negligible (at most 2%); i.e., nearest-replica routing adds marginal value over pervasive caching.
- Cache provisioning (i.e., population-based and uniform) does not affect the relative performance of the representative designs.

Implications: These results suggest that an "edge" caching deployment provides almost the same benefits to both users and the network as a universal caching architecture with nearest-replica routing. This is important because edge deployment is naturally suited for an incremental deployment path for ICN on two counts. First, there is an immediate benefit (and incentive) to a group of users who have a cache server deployed near their access gateways. Second, and perhaps more crucially, this benefit is independent of deployments (or the lack thereof) in the rest of the network. This naturally motivates users to deploy a cache, or a CDN or ISP to deploy a cache on their behalf, without depending on adoption by other providers.

5. SENSITIVITY ANALYSIS

The results of the last section are based on a fixed configuration with a specific popularity distribution, cache size, access-tree arity, etc. In this section, we perform an extensive sensitivity analysis across different configuration parameters using synthetically generated request traces. For clarity, we only show results from the largest topology (AT&T) as the results are similar across topologies.

Rather than look at all cache architectures, here we focus on the two extreme points in this section, namely, ICN-NR and EDGE. In the following results, we report a *normalized improvement* metric:

$$RelImprov_{ICN-NR} - RelImprov_{EDGE}$$

where $RelImprov$ is the improvement over the no-caching scenario that we mentioned in the previous section. By construction, a positive value of this measure implies ICN-NR performs better than EDGE and a negative value implies that EDGE performs better.

For clarity of presentation, we take the following approach in running the sensitivity analysis. First, we begin by analyzing one dimension at a time, while retaining the baseline setup from the previous section for the remaining parameters. Then, we focus on the combination of parameter(s) that provides the best performance improvement for ICN-NR.

5.1 Single-Dimension Sensitivity

Zipf parameter α: Figure 8(a) shows that with increasing α, the gap between EDGE and ICN-NR becomes less positive. This is intuitively expected—as α increases, popular objects get a larger share. This reduces the value of pervasive caching and nearest-replica routing because most of the requests are served from the edge caches.

Cache budget: Next, we consider the effect of increasing the cache size in Figure 8(b). As in Section 4, we represent the per-router cache size as a fraction of the total number of objects being requested. We see that the maximum improvement that ICN-NR can provide is around 10% when each cache can store $\approx 2\%$ of the objects. We also observe an interesting non-monotonic effect in the performance gap as a function of cache size. The reason is that with very small caches, none of the caching architectures are effective. With a sufficiently large cache ($> 10\%$), however, the edge caches account for a significant fraction of the requests and thus the marginal utility of interior caches is very low.

Spatial skew: In the previous section, we considered a homogeneous request stream where requests at different network locations are drawn from the same object popularity distribution. There are likely to be regional differences across request streams at different locations. Thus, we explore the effect of *spatial skew* in Figure 8(c). A spatial skew of 0 means that the requests at all locations follow the same global popularity distribution (i.e., objects have a unique global ranking). A spatial skew of 1, at the other extreme, implies that the most popular object at one location may become the least popular object at some other location.[5] Figure 8(c) shows that as the spatial skew increases, ICN-NR outperforms EDGE. Intuitively, with a large spatial skew, a less popular object at one location may become popular at a nearby location. Thus, caching objects with different popularity distributions across edge locations inside the network magnifies the benefit of ICN-NR.

[5]While the specific spatial skew metric we use is not crucial, we define it for completeness: Suppose there are O objects and P PoPs, and r_{op} denotes the rank of object o at PoP p. Let $S_o = stdev(r_{op})$ be the standard deviation of ranks of object o across all PoPs. Then, *spatial skew* $= \frac{avg(S_o)}{O}$.

arity	Latency gain (%)	Congestion gain (%)	Origin load (%)
2	10.29	9.14	6.27
4	9.12	8.28	5.35
8	7.95	7.01	4.66
64	1.76	0.90	0.34

Table 4: *Effect of access tree arity on performance gain of ICN-NR over EDGE.*

Access-tree arity: Our baseline uses a fixed binary tree. Here, we evaluate how the structure of the access tree impacts the performance difference by changing the arity while adjusting the height of the access trees to keep the total number of leaves per tree fixed. Table 4 shows that as the access-tree arity increases, the performance gap between ICN-NR and EDGE decreases. This is not surprising: with our cache budgeting mechanism, the ratio of total cache budget between EDGE and ICN-NR in a tree of arity k is $\frac{k-1}{k}$; with a higher k this ratio comes closer to 1. In some sense, increasing arity in this case has a similar effect to normalizing the cache budgets in EDGE-Norm.

Other parameters: For completeness, we mention three other parameters that might be relevant. First, rather than assume unit latency cost per hop, we vary the *latency model* in two ways: (1) arithmetic progression of latency toward the core and (2) a scenario where the latency of each hop at the core network is d times higher. (We pick this latency model to magnify the benefit of ICN-NR.) Under both models, the maximum performance gap between ICN-NR and EDGE is less than 2%. This can be explained in part by the intuition from Section 2.2; the intermediate levels see far fewer requests.

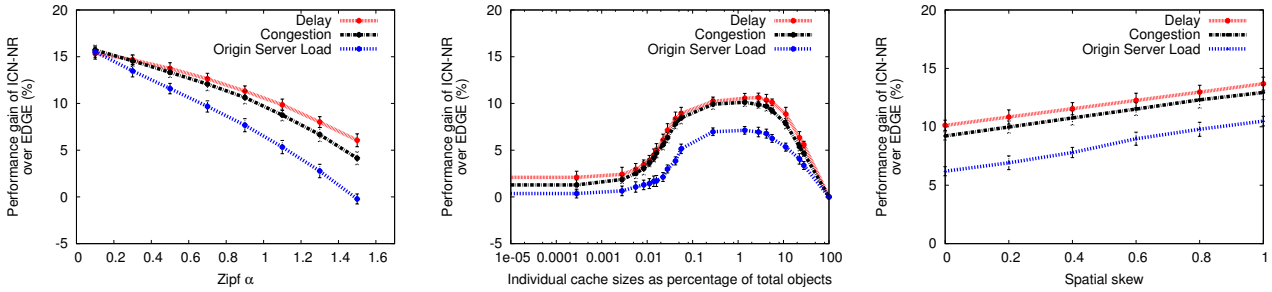
Second, we vary the request serving capacity. In this case, the number of queries each node can serve in a certain period of time is limited. If a request arrives at a cache that is overloaded, this request is redirected to the next cache on the query path (or the origin). Again, we see that the maximum performance improvement of ICN-NR over EDGE in this case is less than 2%.

Finally, we investigated request streams with heterogeneous object sizes (as observed in the real traces). This has minimal impact on our performance results (less than 1%), as we do not see a strong correlation between an object's size and its popularity.

We do not present the results for a range of other parameters as their effects are small compared with the above parameters.

5.2 Best Scenario for ICN-NR

We want to understand under what scenario(s) ICN-NR has the best performance benefits over EDGE and by how much. To this end, we begin by ordering the configuration parameters in decreasing order of the magnitude of the relative improvement they yield. Then, we progressively change one dimension at a time to maximize the gap between ICN-NR and EDGE in Figure 9. In the figure, Baseline is the configuration from Section 4. In each subsequent configuration, we change one of the configuration parameters (while all other parameters maintain their current values) as follows: (1) Alpha* uses $\alpha = 0.1$; (2) Skew* sets the spatial skew to 1; (3) Budget-Dist* uses uniform budgeting; and (4) Node-Budget* sets the cache sizes to be $F = 2\%$ of the number of objects requested. (For completeness, we also tried a brute force exhaustive enumeration of parameters and found that the best case is identical to combining the best single-dimensional results.) We see that with the best combination of parameters, ICN-NR can improve the performance at most 17% relative to EDGE.



(a) Zipf α (b) Cache budget (c) Spatial skew

Figure 8: *Effect of varying different simulation parameters on the performance gap between ICN-NR and EDGE. Here, we consider a fixed total cache budget across the nodes.*

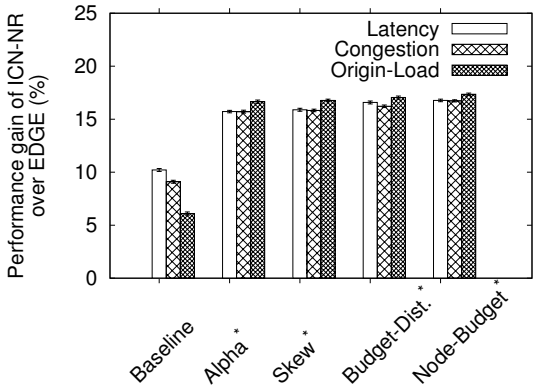


Figure 9: *Exploring the best scenario for ICN-NR by progressively setting configuration parameters to yield the maximum performance gap w.r.t. EDGE.*

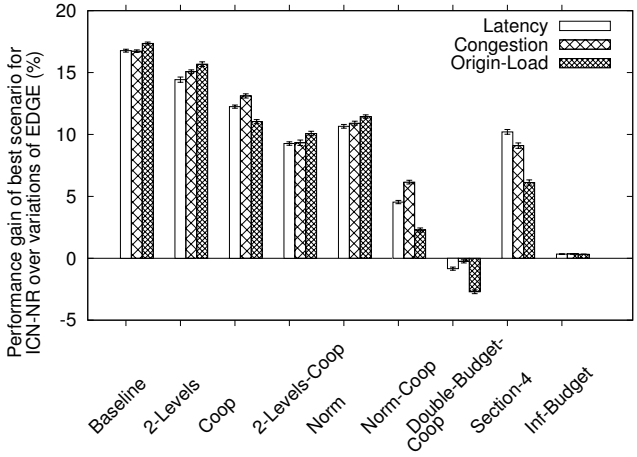


Figure 10: *Bridging the performance gap between the best scenario for ICN-NR and EDGE via simple extensions to EDGE. We also show the baseline from Section 4 and a hypothetical infinite cache setting.*

The next question we ask is whether this performance gap is fundamental or whether it can be bridged using simple extensions to EDGE. As we saw in Section 4, cooperation (EDGE-Coop) and doubling the budget (EDGE-Norm) reduces the gap in the baseline simulations. Figure 10 shows how several natural extensions to EDGE bridge the performance gap. In this figure, Baseline refers to EDGE without any changes; 2-Levels is EDGE augmented with one more layer of caching (at the level above the edge); Coop refers to EDGE-Coop; 2-Levels-Coops combines the features of 2-Levels and Coop; Norm refers to EDGE-Norm; Norm-Coop is a combination of EDGE-Norm and Coop; Double-Budget-Coop is the same as Norm-Coop with the budget doubled. There are also two points of reference in the figure: Section-4 is the set of performance measures from Section 4 and Inf-Budget is a scenario in which both EDGE and ICN-NR have infinite caches (i.e., each cache has enough space to store O objects). We see that the combination of EDGE-Norm and local cooperation can bring down the gap to around 6%.

5.3 Key Observations and Implications

The main observations from our sensitivity analysis are:

- The key parameters that effect the relative performance of ICN-NR over EDGE are Zipf α and spatial skew.
- The best possible performance benefit of ICN-NR over EDGE (across all metrics) by setting the above parameters to be favorable to ICN-NR is only 17%.
- Simple extensions to EDGE such as putting all the cache at the edge and enabling local scoped cooperation can reduce even this best case performance gap to 6%.
- Doubling the edge cache sizes can in fact make EDGE better than ICN-NR.

Implications: In summary, these observations imply that we can match the best-case quantitative performance of ICN, and avoid most of the deployment headaches, merely by increasing the size of the edge caches or enabling simple cooperative strategies.

6. INCREMENTALLY DEPLOYABLE ICN

Our quantitative results showed that most of the benefits of caching can be achieved through edge caching. In this section, we show that the qualitative advantages of ICN (i.e., security, mobility, and ad hoc mode) are also achievable in an end-to-end fashion (see Table 1). Here, we outline one possible design of an incrementally deployable ICN or idICN, where caching as well as the qualitative aspects of ICN are implemented at the edge of the network. Note that our goal here is not to reinvent CDNs. Rather, we want a design that is architecturally simpler than today's CDNs and yet more incrementally deployable than clean-slate ICN designs. To this end, we deliberately choose a path that requires only moderate re-engineering and uses tools that are already available.

Our goal in designing idICN is to place the most crucial aspects of ICN functionality within reach of practical deployments. Towards this end, we build upon HTTP, as it already provides a fetch-by-name primitive (as opposed to IPs fetch-by-hostname). HTTP, however, comes with a host-centric naming (DNS) and security model (HTTPS), and requires explicit configuration of proxies. We remedy the former by using a self-certifying approach to naming, and the latter by using an existing protocol (WPAD) as a mechanism to automatically configure end-hosts to use a nearby edge proxy. Furthermore, we also show how ad hoc data sharing and mobility can be practically achieved in idICN.

We show that the qualitative properties from Table 1 can be achieved via purely end-to-end mechanisms and building on commodity technologies that already exist in the content distribution/HTTP world. We do not claim that idICN is the only feasible design, or an optimal one, and we intentionally leave open choices regarding specific algorithms or implementations to the end applications and administrative domains (ADs).

Figure 11 shows a high-level view of the idICN operation. First, clients automatically discover the location of the HTTP proxy configuration file as we will see in Section 6.2 (step 1). With the client configured to use the proxy, the client's HTTP requests are explicitly directed through the proxy cache (without even requiring the client to perform a name lookup or a per-request connection setup) (step 2). The cache responds immediately if it has a fresh copy of the requested object (step 7); otherwise, it queries the name resolution system (step 3). Using the information from the resolvers, the proxy sends a request towards the origin server (or replicas) (step 4). If the reverse proxy (deployed by the content provider) does not already have a fresh copy of the object, it routes the request to the origin server and receives the content (step 5). The reverse proxy adds relevant metadata (e.g., to provide content-oriented security) to the HTTP response and sends it to the proxy (step 6). The proxy authenticates the content using enclosed digital signatures (see Section 6.1) and serves the content to the client (step 7).

To advertise new content, origin servers publish the names of newly generated contents through the reverse proxy (step P1), which, in turn, registers the names with the idICN name resolution system (step P2) as well as DNS (for backward compatibility). Reverse proxies also generate signatures and a list of policies and mirrors, cache them, and include them in the HTTP header of their responses. Finally, thanks to the properties of the idICN names (as we will discuss shortly), presenting content along with its signature is sufficient to update the idICN name resolution system. We envision a system similar to SFR [51] to implement name resolution and registration in idICN—the specific realization is orthogonal to our goal of achieving the benefits of ICN.

Even though the full benefits of idICN are only available to content providers and to clients located in ADs that fully adopt idICN, there are three key advantages regarding its deployment: (1) idICN leaves the current infrastructure intact, operating in parallel during the transition to idICN, (2) clients in ADs that support idICN can still retrieve content from providers that do not support idICN, and (3) clients in ADs that do not support idICN can retrieve content from providers that do support idICN.

6.1 Content-Oriented Security

Most ICN designs adopt some form of self-certifying names [17], where names embed the relevant cryptographic information such as public keys or their cryptographic hashes. By signing the content, the principal responsible for the content (i.e., the owner of the relevant public key) can prove that the content is associated with the name. Thus, this model of security is completely data-oriented; the client can verify its provenance (e.g., [17, 30, 42, 51]) irrespective of who provided the data (e.g., CDN, local cache, or a stranger on the bus). This is in contrast to the current security model that tries to secure the data path or end hosts, and to retrieve the content from a trusted party.

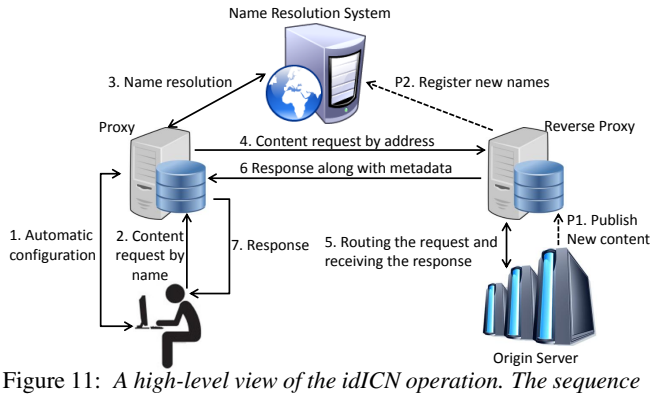


Figure 11: *A high-level view of the idICN operation. The sequence of steps per request depends on whether the proxy has the requested content. If it does, only steps 1, 2, and 7 are performed; otherwise, steps 1 through 7 are taken. In the latter case, if the reverse proxy has the content, step 5 is skipped. In parallel, the origin server publishes newly generated contents (steps P1 and P2).*

The ICN approach decouples delivery from security and separates trust management and verification. It is solely up to clients to decide which data sources they are going to trust (and how they such make trust determinations). We believe that this change in security models is important and long overdue, having been first proposed almost a decade ago [30, 51].

Content-oriented security can be provided via: (1) extending HTTP to negotiate and serve required metadata (keys, signatures, etc.), and (2) standardizing a self-certifying naming scheme. The Metalink HTTP extension [8], S-HTTP [36] (competing with HTTPS at the time), and HTTPi [40] (an improvement over S-HTTP to support today's common use cases) show how to possibly extend HTTP for this purpose. Moreover, the widespread use of self-certifying names in peer-to-peer networks (e.g., the use of content hashes as names in the Magnet URI scheme [31]) shows that self-certifying names, albeit for static content, are commonplace. idICN adopts a DONA-style self-certifying flat naming scheme and the Metalink description format (see [28]). We discuss both next.

Self-certifying names: Our naming scheme must be backwards compatible with DNS, yet offer the ICN security model. Following the lead of numerous existing proposals [44, 50, 51], we achieve this using an idICN proxy name resolver under `idicn.org` with names of the form `name.idicn.org` where name is a self-certifying name. We use names of the form `name=L.P` where `P` is a cryptographic hash of the publisher's public key and `L` is a content label assigned to the content by `P`.[6]

Client browsers issue content requests as they do today: by requesting an "address", which in this case is a URI encoded as `L.P.idicn.org`. For content to be reachable via DNS, it must be registered in the new `.idicn.org` domain. We rely upon a consortium of entities to host idICN resolvers (e.g., Google, Yahoo!, Microsoft, Akamai, and Verisign). Because the total traffic these resolvers would see is far below what these companies handle for their normal web services today, we think this is a rea-

[6]Backward compatibility with DNS does come at a price. Labels in a domain name are restricted to 63 characters; we cannot support hashing algorithms with digests $>$ 63 characters (e.g., SHA-512).

sonable assumption. These resolvers need only check for cryptographic correctness (rather than rely on any other form of trust); the `.idicn.org` namespace will provide a public API allowing anyone who can sign with `P`'s private key (or produce an authorizing signature from `P`) to register names of the form `L.P.idicn.org`. To resolve a request for a particular name, the name resolution system first looks for exact matches on `L.P` and, failing that, looks for a match for `P`. Moreover, the entries can point to other resolvers that can provide more fine-grained resolution (e.g., the basic resolver might only have an entry for `P`, which then points to a resolver that has entries for individual `L.P` names).

Metadata using metalink/HTTP: In practice, we also need to provide relevant metadata along with the content; e.g., locations of replicas or working copies, cryptographic hashes, and digital signatures. To this end, we leverage the Metalink standard [8], which is an XML-based download description format that provides the metadata of the named content.[7] Metalink-enabled HTTP clients and proxies understand the relevant HTTP headers (e.g., to verify the authenticity and integrity of the data, discover faster mirrors, etc.), while legacy clients simply ignore them.

Together, our naming scheme and metadata embedding enable the new data-oriented security model. We note that the client or the proxy should authenticate the content; the latter would put trust on proxies, while the former would require software changes, requiring incremental deployment.

6.2 Automatic Proxy Configuration

Since idICN is based on HTTP and uses its support for proxies, now we describe how hosts can automatically discover and connect to a nearby HTTP proxy without requiring any manual setup. Because content delivery primitives are baked into the basic ICN architecture, there is no need for transparent caching or other "hacks" that make the network brittle. Moreover, when used in the "broadcast" mode, ICN designs can be realized in ad hoc environments without any explicit management. Fortunately, there are widely available techniques to address both concerns. idICN provides automatic proxy configuration via built-in support in browsers and the OS [14, 33] and relies on Zero Configuration Networking (Zeroconf) [53], which enables content sharing in a network with no infrastructure for address assignment and name resolution.

Client proxy configuration: Hosts in idICN use the Web Proxy Autodiscovery Protocol (WPAD) [14] to locate a URL of a Proxy Auto-Config (PAC) file [33]. To support WPAD, networks need to configure their DHCP or DNS servers to announce the PAC file location. Once the PAC file is located and fetched, the browser invokes the JavaScript function `FindProxyForURL(url, host)` contained in the file to determine the proxy to use for a given URL. WPAD and PAC are widely supported by all major operating systems and browsers [4] and are extensively used in enterprise networks.

Content sharing in ad hoc mode: For completeness, we also discuss content sharing without any infrastructure for network configuration and name resolution. We do note that the techniques required to enable ad hoc operation of idICN are optional and orthogonal to the rest of our design.

To support the ad hoc mode, idICN relies on two aspects of Zeroconf: (1) IP address assignment without obtaining outside information (e.g., from a DHCP server) [10, 45], and (2) distributed name publishing and resolution over multicast using the familiar DNS interface in the absence of a centralized DNS server (mDNS) [11]. Support for IP link-local configuration and mDNS is readily built in Linux distributions through Avahi[8], in OS X and in iOS through Bonjour[9]; and several open-source cross-platform implementations are also available. We note that support for Zeroconf does not require any changes to the networking devices (e.g., wireless routers) as long as they are not filtering local multicast traffic.

To show the feasibility of sharing cached content in a network with limited local connectivity, we prototyped a simple HTTP proxy (350 lines of Python code) to expose Chrome browser's cache over the network when the IP address is link-local. Consumers do not need to do anything to access available content as long as they have a Zeroconf stack and use mDNS as a fallback name resolution mechanism. Only users who wish to share their browser cache need to deploy our prototype. The proxy publishes an alias for the machine for each domain name with content in the cache, and serves content out of the Chrome cache if requested.

As an example, consider a case where Alice and Bob are connected to the same network, and Alice has a cached copy of CNN headlines while Bob is looking for it. Upon acquiring a link-local IP address, Alice's ad hoc proxy publishes domain name `cnn.com` over mDNS. Bob enters `cnn.com` to fetch the CNN headlines and his browser initiates a DNS lookup for `cnn.com`. Without a configured DNS server to contact, Bob's name switching service sends an mDNS query for `cnn.com`, which resolves to Alice's machine address. Bob's browser now initiates an HTTP connection to Alice's ad hoc proxy to request `cnn.com` (via an HTTP GET) that the proxy serves out of Alice's browser cache. A limitation of this scheme, due to its reliance on DNS, is that if different machines have content for the same domain, only one of them will be able to publish it. Deployment of the flat names (`L.P.idicn.org`), however, addresses this issue.[10]

6.3 Mobility Support

To support mobility over HTTP, idICN requires applications to, first, incorporate session management (e.g., via HTTP cookies for stateful, or byte ranges for stateless, communications) and, second, update their location using dynamic DNS. With session management, applications can seamlessly work upon reconnection. This form of session management is quite common over HTTP (e.g., sessions spanning several days) and may even be a good substrate for DTN applications. With dynamic DNS updates, mobile servers must announce their locations. Upon loss of connectivity (e.g., because of moving the client, the server, or both), the application attempts to re-establish the communication. If the server has moved, the client's name lookup resolves to the server's new IP address.

6.4 Summary

We have outlined a dirty-slate, incrementally deployable design called idICN, which uses edge caching to gain most of the caching benefits of ICN, and end-to-end mechanisms to get the key qualitative properties of ICN. The design mainly utilizes previously standardized and widely used techniques from the past decade, requiring small changes to hosts or their protocols. We believe this is a key strength of idICN, as it significantly enhances its deployability.

idICN does, however, involve three changes to the Internet: (1) infrastructure deployment by ADs, which we expect to be a small barrier given the eagerness of ISPs to enter the CDN arena; (2) caching behavior, which can be realized on an AD-by-AD basis and need not be subject to global standards; and (3) actions by content providers to publish content within idICN, but allowing providers

[7] E.g., see `http://releases.ubuntu.com/releases/12.10/ubuntu-12.10-desktop-amd64.metalink`.

[8] `http://www.avahi.org/`

[9] `http://www.apple.com/support/bonjour/`

[10] Here, documents would be published over mDNS.

to adopt idICN independently. At first glance, these may appear as potential stumbling blocks, but in comparison with clean-slate ICN designs that require changing every router, every application, and every networking stack, we believe the changes mandated by idICN are minimal and incrementally attainable.

To demonstrate the feasibility and ease of use of idICN, we have developed a prototype for the reverse proxy (generating Metalink metadata and signatures) based on the Metalink plugin of the Apache Traffic Server. We emulate the support for a few websites to show idICN's operations using legacy clients. Please visit `http://www.idicn.org/` for more information.

7. DISCUSSION

Workload evolution: Internet workloads are in a constant state of flux. For instance, a combination of technology trends—social networks, user-generated content, and smartphones—is creating more "long-tailed" content [48]. Even if in the worst case, we approximate these trends using a combination of low α and high spatial skew as in Section 5, the marginal benefit of pure ICN architectures seems to be low. While we cannot further speculate how this evolution will play out, this only serves to reiterate the spirit of our work and parallel efforts to avoid "overfitting" the network infrastructure to specific workloads [19, 22].

Economic and policy aspects of idICN: As noted elsewhere [18], there are valid economic (e.g., analytics for providers), legal (e.g., serving content with access restrictions), and privacy concerns (e.g., caches know what you are requesting) surrounding ICN. It is likely that idICN inherits some of these difficulties as well and we do not have good answers yet. We do believe, however, that by scoping the degree of caching and making it easier to attribute where a request was served from, idICN might simplify solutions to address some of these concerns. For instance, we know exactly which caching proxy the resolver redirected a client to; this proxy can provide the necessary accounting and reporting. These are open and valid concerns for both ICN and idICN that need to be addressed in future work.

When is it viable to deploy a cache: Providers need incentives to deploy caches in idICN (and ICN for that matter) and thus a natural question here is where in the network should they choose to do so. Operating a cache involves both fixed upfront costs and several operational costs (e.g., rack space, bandwidth, power, and cooling). Based on informal and anecdotal evidence with CDN operators, a rough rule of thumb is that the lifetime of caching hardware is roughly 3–5 years and that this cache should serve enough traffic to be profitable. We speculate that idICN deployment will be driven by such economic considerations.

What idICN does not provide: We do acknowledge that a clean-slate ICN architecture may provide other benefits (e.g., broadcast support or unifying caching and error recovery) that fall outside the scope if idICN. Our focus in this paper is on the most prominently perceived benefits of ICN.

For instance, idICN does not attempt to address two less well-understood benefits of ICN: protection against denial of service and congestion control. ICN eliminates some simple DoS attacks due to IP spoofing [16]; the biggest benefit, however, comes from the caching to defend against request floods. We do not believe that there is anything fundamental here; the benefit here is simply amplifying the effective number of servers similar to commercial services that offer DoS protection today [12]. Note that an architecture based on edge caching, such as idICN, provides approximately the same hit-ratios as a pervasively deployed ICN, indicating that such an edge cache deployment can provide much of the same request flood protection as pervasively deployed ICNs. There is also some perception that the hop-by-hop flow control of some ICN proposals (maintaining flow balance) substantially reduces the dependence on end-to-end congestion control [23]. idICN does not attempt to provide a clean-slate solution and simply retains standard congestion control, but with separate congestion management on individual segments: proxy-to-proxy and proxy-to-host.

8. RELATED WORK

The ICN related work is vast, including at least three ACM SIGCOMM ICN workshops, two Dagstuhl gatherings, an ICNRG research group, and numerous journal special editions. Covering this here is impossible due to space constraints. We therefore focus on the biggest research projects as well as the research papers that are most relevant to our focus.

DONA consistently uses nearest-replica routing while CCN/NDN uses a hybrid of nearest replica (in LANs) and shortest path to origin (in WANs). Qualitative features, such as intrinsic binding and naming, can be made to only use end-to-end support as we suggested. For example, NDN supports both human-readable names and self-certifying names. The latter could be used in a similar fashion to our idICN design.

The PSIRP project and its successor PURSUIT [15] take a different approach based on the publish-subscribe paradigm. Many of the core contributions of these projects, such as zFilters [26], will be useful in network architecture designs. The rendezvous back-end of the project would, however, require major pervasive changes to the architecture. It can, nevertheless, be deployed incrementally; the name resolution service can be implemented in a way that does not require pervasive caching, nor replacement of the existing switching infrastructure on the Internet.

The NetInf design from the 4WARD project is also based on a name lookup resolution mechanism, using a DHT implementation [2]. We believe that the suggested clean slate design could be adapted to be deployed in a way that does not require pervasive caching. Furthermore, their support for an information abstraction model, which allows multiple different representations of the same object to exist, is highly useful and can be used in an end-to-end fashion with architectures like our idICN.

The Serval project is not an ICN, but rather focuses on supporting a service-centric network [32]. As such, Serval does not require a pervasive caching infrastructure. Serval shares many things with our idICN design, including self-certifying names and much functionality placed on end hosts. In particular, Serval provides details, such as API design, about how ICNs or service-centric networks could be integrated into the stack of modern computers.

We are not the first to raise questions about the value and viability of ICN architectures. These include concerns regarding the scalability of ICN-capable routers [34], the privacy implications of ICN [3], legal (e.g., access restrictions and copy right conerns [1]) and economic considerations (at an Internet scale deployment [34]) underlying such an infrastructure [1], and the performance benefits that ICN can provide [18]. While our work follows in this spirit, our contribution here is two-fold. First, we provide a *quantitative* basis to analyze the performance benefits of universal caching and nearest-replica routing. Second, we provide a reference design for an incrementally deployable ICN architecture that retains most of the advantages of hitherto proposed ICNs.

9. CONCLUSIONS

Our work can be viewed as an application of the end-to-end argument—we should impose significant changes to the net-

work only if doing so will offer substantial performance improvements [38]. We apply this principle to many of the perceived benefits of ICN architectures. We find that the components of ICN that might need drastic changes to the network as envisioned by some ICN proposals (pervasive caches and nearest-replica routing) do not appear to be fundamentally necessary. Furthermore, the other components of ICN can be implemented in a backwards-compatible fashion using techniques that already exist today. Building on these insights, we presented a roadmap for an incrementally deployable architecture that can achieve the benefits of ICN without a forklift upgrade to existing networks.

10. ACKNOWLEDGMENTS

We would like to thank our shepherd David Oran, the SIGCOMM reviewers, and Zafar Ayyub Qazi for their feedback. Barath Raghavan contributed significantly to early discussions that informed the idICN design. This work was supported in part by NSF grants CNS 1117161 and 1040838, and AFRL grant FA8750-11-1-0262. Seyed Kaveh Fayazbakhsh was supported in part by a Renaissance Technologies Fellowship.

11. REFERENCES

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

[2] P. A. Aranda, M. Zitterbart, Z. Boudjemil, M. Ghader, G. H. Garcia, M. Johnsson, A. Karouia, G. Lazar, M. Majanen, P. Mannersalo, D. Martin, M. T. Nguyen, S. P. Sanchez, P. Phelan, M. Ponce de Leon, G. Schultz, M. Sollner, Y. Zaki, and L. Zhao. 4WARD. `http://www.4ward-project.eu/`, 2010.

[3] S. Arianfar, T. Koponen, B. Raghavan, and S. Shenker. On preserving privacy in content-oriented networks. In *Proc. SIGCOMM Workshop on ICN*, 2011.

[4] Browser Support for PAC and WPAD. `http://findproxyforurl.com/browser-support/`.

[5] B. Baccala. Data-oriented networking. `http://tools.ietf.org/html/draft-baccala-data-networking-00`, 2002.

[6] D. Beaver, S. Kumar, H. C. Li, J. Sobel, and P. Vajgel. Finding a needle in haystack: Facebook's photo storage. In *Proc. OSDI*, 2010.

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *Proc. INFOCOM*, 1999.

[8] A. Bryan, N. McNab, T. Tsujikawa, P. Poeml, and H. Nordstrom. Metalink/HTTP: Mirrors and Hashes. RFC 6249 (Proposed Standard), June 2011.

[9] Emerging Network Consortium Brings Industries Together to Innovate with Content-Centric Networking (CCN). `http://www.mach.com/en/News-Events/Press-Room/Press-Releases/Emerging-Network-Consortium-Brings-Industries-Together-to-Innovate-with-Content-Centric-Networking-CCN`.

[10] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[11] S. Cheshire and M. Krochmal. Multicast DNS. Technical report, IETF, December 2011.

[12] CloudFlare security. `http://www.cloudflare.com/features-security`.

[13] COntent Mediator architecture for content-aware nETworks (COMET). `http://www.comet-project.org/`.

[14] I. Cooper, P. Gauthier, J. Cohen, M. Dunsmuir, and C. Perkins. Web proxy auto-discovery protocol. Technical report, IETF, May 2001.

[15] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos. Developing information networking further: From PSIRP to PURSUIT. In *Proc. BROADNETS*, 2010.

[16] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS and DDoS in named-data networking. *CoRR*, abs/1208.0952, 2012.

[17] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in Content-Oriented Architectures. In *Proc. SIGCOMM Workshop on ICN*, 2011.

[18] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proc. HotNets*, 2011.

[19] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Intelligent design enables architectural evolution. In *Proc. HotNets*, 2011.

[20] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube traffic characterization: A view from the edge, imc. In *Proc. IMC*, 2007.

[21] M. Gritter and D. R. Cheriton. TRIAD: A New Next-Generation Internet Architecture. `http://www-dsg.stanford.edu/triad/`, 2000.

[22] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: efficient support for evolvable internetworking. In *Proc. NSDI*, 2012.

[23] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. CoNEXT*, 2009.

[24] V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, k. claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh. Named Data Networking (NDN) project. `http://named-data.net/techreport/TR001ndn-proj.pdf`, 2010.

[25] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang. Cooperative content distribution and traffic engineering in an ISP network. In *Proc. SIGMETRICS*, 2009.

[26] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: line speed publish/subscribe inter-networking. In *Proc. SIGCOMM*, 2009.

[27] D. Kim, J. Kim, Y. Kim, H. Yoon, and I. Yeom. Mobility support in content centric networks. In *Proc. SIGCOMM Workshop on ICN*, 2012.

[28] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proc. SIGCOMM*, 2007.

[29] L. Li, X. Xu, J. Wang, and Z. Hao. Information-centric network in an ISP. `http://tools.ietf.org/html/draft-li-icnrg-icn-isp-01`, 2013.

[30] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *Proc. SOSP*, 1999.

[31] G. Mohr. Magnet uri scheme draft, 2002. `http://magnet-uri.sourceforge.net/magnet-draft-overview.txt`.

[32] E. Nordstrom, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, , and M. J. Freedman. Serval: An end-host stack for service-centric networking. In *Proc. NSDI*, 2012.

[33] Navigator proxy auto-config file format. Netscape Navigator Documentation, March 1996.

[34] D. Perino and M. Varvello. A reality check for content centric networking. In *Proc. SIGCOMM Workshop on ICN*, 2011.

[35] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, and B. Maggs. Enabling content-aware traffic engineering. *ACM SIGCOMM CCR*, 42(5):21–28, October 2012.

[36] E. Rescorla and A. Schiffman. The Secure HyperText Transfer Protocol. RFC 2660 (Experimental), August 1999.

[37] Scalable and Adaptive Internet Solutions (SAIL). `http://www.sail-project.eu/`.

[38] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4), Nov. 1984.

[39] A. Sharma, A. Venkataramani, and R. Sitaraman. Distributing content simplifies isp traffic engineering. In *Proc. SIGMETRICS*, 2013.

[40] K. Singh, H. J. Wang, A. Moshchuk, C. Jackson, and W. Lee. Practical end-to-end web content integrity. In *Proc. WWW*, 2012.

[41] D. Skeen. Vitria's publish-subscribe architecture: Publish-subscribe overview. `http://www.vitria.com/`, 1998.

[42] D. Smetters and V. Jacobson. Securing Network Content. Technical report, PARC, October 2009.

[43] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1), Feb. 2004.

[44] S. Sun, L. Lannom, and B. Boesch. Handle System Overview. RFC 3650 (Informational), November 2003.

[45] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007.

[46] Tibco enterprise message service. `http://www.tibco.com/`.

[47] Your gadgets are slowly breaking the internet. `http://www.technologyreview.com/news/509721/your-gadgets-are-slowly-breaking-the-internet/`.

[48] S. Traverso, K. Huguenin, I. Trestian, V. Erramilli, N. Laoutaris, and K. Papagiannaki. Tailgate: handling long-tail content with a little help from friends. In *Proc. WWW*, 2012.

[49] C. Tsilopoulos and G. Xylomenos. Supporting diverse traffic types in information centric networks. In *Proc. SIGCOMM Workshop on ICN*, 2011.

[50] G. Wachob, D. Reed, L. Chasen, W. Tan, and S. Churchill. Extensible resource identifier (XRI) resolution version 2.0. *Committee Draft*, 3, 2008.

[51] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the Web from DNS. In *Proc. NSDI*, 2004.

[52] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. In *Proc. SOSP*, 1999.

[53] The IETF Zeroconf Working Group, 2004. `http://datatracker.ietf.org/wg/zeroconf/charter/`.