On Caching and Routing in Information-centric Networks

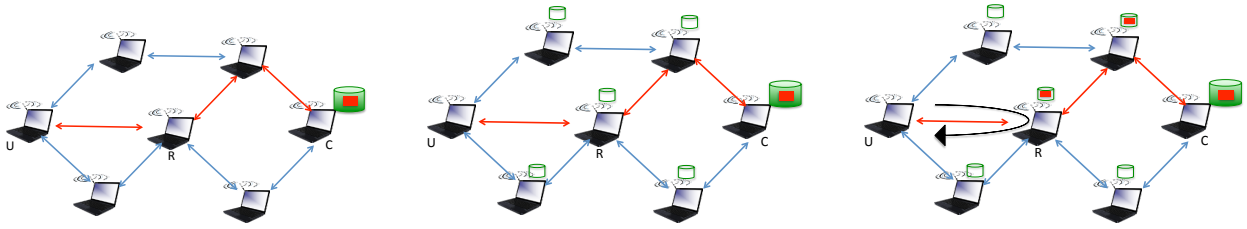
Anand Seetharam
Department of Computer Science, SUNY Binghamton
aseethar@binghamton.edu

Abstract

With the exponential growth of content in recent years, users are primarily interested in obtaining a particular content and are not concerned with the host housing the content. By treating content as a first class citizen, information-centric networks (ICN) seek to transform the Internet from a host-to-host communication model to a content-centric one. A key component of ICN is to cache content at storage-enabled routers. By caching content at in-network routers, network performance can be improved by delivering content from routers closer to the user and not from the origin servers (content custodians). In this article, we provide an overview of the state-of-the-art cache management and routing policies in ICN. We present a small comparison study of some state-of-the-art algorithms on a real Internet topology to help the reader appreciate how the different strategies compare against one another. Our simulation results demonstrate that it is hard to pick a winner among existing policies. We conclude the paper with a discussion of open research questions.

I. Introduction

In recent years, a new networking paradigm called information-centric networking (ICN) has been proposed that aims to transform today's Internet from a host-based communication model to a content-centric one. An important feature spanning across ICN designs is incorporating in-network caching at storage-enabled routers. In ICN, while content is permanently housed at a few origin servers (content custodians), storage-enabled routers can cache content as it passes through them. Therefore, in contrast to today's host-centric Internet, in ICN, en route routers, in addition to the content custodian, can also satisfy requests generated by users. The main advantages of serving content to a user from a router are: i) users experience lower latency as all requests need not be routed to the content custodian; ii) congestion in the network core is alleviated as a significant fraction of requests can be serviced by routers located close to the user (i.e., network edge).



(a) Host-centric network (b) Information-centric network (1) (c) Information-centric network (2)

Figure 1: Comparison between host-centric networks and information-centric networks

Figure 1a depicts a typical host-centric network. Let us assume that user U requests the red piece of content that is housed at server C. The request is routed towards C along the red path according to the routing algorithm, as shown in Figure 1a. When C receives the request, it serves the red content to U. Let us contrast this scenario with that of an ICN, as shown in Figure 1b. All network routers are storage-enabled and capable of caching content; in general only a subset of routers in an ICN might have storage capability. We assume all content is available at custodian C. When U requests the red content, it is downloaded from C and cached at en route routers. Let's assume that after some time U requests the red content again. On its way towards the custodian, the request reaches router R that has the requested content in its cache (Figure 1c). The content is then downloaded to U from R and the request is not forwarded to C (as indicated by the curved black arrow in Figure 1c).

Therefore, in ICN, the primary focus is content; publishers generate content that is then permanently hosted at content custodians. Each piece of content is uniquely identified by a content name. In ICN, users generate *Interest* packets for a particular content. When the *Interest* packet reaches a router having that content, the router generates a *Data* packet to send the content back to the user. On its way back to the user, the content might be cached at en route routers, depending on the cache management policy. Hence, one of the main challenges in ICN is to design efficient cache management and routing policies that can enable the user to locate and obtain content along the fastest route.

In this article, we provide an overview of ICN data forwarding techniques, focusing mainly on the routing and cache management aspects. Our first objective is to provide the reader an understanding of the guiding principles behind the design of state-of-the-art caching and routing strategies, so that they can compare and contrast them. Secondly, we demonstrate via a simulation study that it is hard to pick a clear winner among the existing policies and that performance is dictated by a number of network parameters such as content popularity distribution, network topology, content universe and cache size. Finally, we highlight some of the shortcomings of existing research and discuss some open research questions.

II. Caching and Routing Policies

In this section, we outline the state-of-the-art cache management policies (that determine what content to cache and which ones to evict) and routing algorithms in ICN. For a detailed survey of ICN caching and routing policies we refer the reader to [15]. Intelligent caching and routing techniques provide superior performance in comparison to naïve strategies that cache all content that pass through routers and use simple shortest path algorithms for routing.

A. Cache Management Policies

We begin by describing some of the cache management policies designed for ICN. When a piece of content passes through a router, there are two major decisions to make. Firstly, it needs to decide whether to cache the content or not *(cache insertion)*. Secondly, once the router has decided to cache a particular content, it has to decide what content to replace from the cache *(cache eviction)*.

The simplest *cache insertion* policy is the Leave Copy Everywhere (LCE) policy, where every router on the path between the source (i.e., the router serving the content) and the user caches a copy of the content. The LCE policy stores multiple copies of the same content within the network and is not efficient. Recently, a number of *cache insertion* strategies have been proposed to improve performance by decreasing the redundancy of the same piece of content within the network. We discuss a few of them here.

- Leave Copy Down (LCD): The LCD policy was originally proposed for overlay caches, and researchers have adopted it for ICN. In the LCD policy, whenever there is a cache hit, content is replicated only at the router that is one hop downstream towards the user [1]. The LCD policy decreases redundancy and at every step moves the content one step closer to the user. We will see later that the LCD policy, though simple, yields performance comparable to and in some cases better than some state-of-the-art policies.
- Randomly Copy One (RCO): In RCO [14], the content is cached randomly at only one node along the path through which content is downloaded. The authors demonstrate that even such a simple caching policy is a promising policy for ICN and can provide significant performance benefits.
- Cache Less for More (CL4M): CL4M [2] decreases content redundancy by leveraging the concept of centrality to determine the router most suited to cache content en route between the source and user. The centrality of a router is calculated based on the number of content delivery paths that pass through it (i.e., number of shortest paths traversing it) to make caching decisions. Content is cached at the router between source and user with the highest centrality. The intuition is that caching content at a router that lies at the

intersection of a large number of content delivery paths will help satisfy a significant number of *Interest* packets and avoid unnecessary content replication.

- Probabilistic Caching (ProbCache): In [3], the authors adopt a probabilistic model to determine if a router should cache a piece of a content passing through it. They determine the 'caching capability of a path' that provides an indication of the number of times a particular piece of content should be cached on the path between the source and user. For each router on the path, they also estimate a weight and the probability of caching a piece of content at a router (ProbCache) is the product of the caching capability and the weight.
- Latency-aware Caching (LAC): LAC leverages local latency measurements to decide whether to cache a piece of content [4] at a node. The latency for a piece of content at a node is the time elapsed between *Interest* forwarding by that node and the corresponding *Data* reception. It adopts a distributed approach where the probability of caching a piece of content at a given node is proportional to the popularity and the latency locally observed at that node.
- PopCache: PopCache [5] utilizes content popularity to determine whether to cache a particular content or not. The main premise is that popular content should be stored in the routers close to the users, while unpopular content can be cached in routers away from the users. Additionally, it spreads unpopular content evenly among far away caches to prevent content duplication. Several other research papers have also advocated for caching the most popular content at the network edge. In [6], the authors demonstrate that in a tree topology caching effectively at the network edge provides majority of performance benefits in ICN.
- Congestion-aware caching (CACS): The policies described so far do not explicitly take network congestion into account when deciding what content to cache. The authors in [7] propose a network congestion based utility metric to determine what content to cache. The utility function operates at each router and estimates the savings in download latency obtained by caching a particular content at that router. The download latency savings is defined as the difference between the time taken to download the content from this router, and the time it takes to download it from the source. The latency savings are estimated by considering the content popularity, the number of flows through that router and the end-to-end throughput. The end-to-end throughput is dependent on network congestion and is determined based on the bottlenecked link.

The other important aspect of cache management is *cache eviction*. The most widely used cache eviction policies are First in First out (FIFO), RANDOM, Least Recently Used (LRU), Least Frequently Used (LFU) and Time-to-live (TTL) based caching. Note that until a cache is full, no content is evicted from the cache; content gets evicted when a new piece of content arrives and there is no space available in the cache.

- FIFO operates on the First in First Out policy and so content is replaced in the order in which it arrives.
- In the RANDOM replacement policy, when a new piece of content is added to the cache at a router, an existing piece of content is randomly selected and evicted from the cache.
- In the LRU policy, the least recently used content (i.e., content that has not been accessed for the longest duration of time) is evicted from the cache. The LRU caching policy thus ensures that popular content is not easily evicted from the cache. LRU caching policy adapts effectively to temporal changes in network traffic. Several modifications to LRU caching (e.g., p-LRU, where content is inserted into a cache with probability p, with the eviction policy being LRU) have also been proposed in literature.
- The LFU caching policy evicts content that is least frequently used. Assuming that the content popularity is known a priori, the LFU caching policy can be implemented by statically caching the most popular content in the cache (i.e., if the cache size is 'C', then the 'C' most popular content pieces are cached).
- In TTL based caches, each piece of content in the cache has a time stamp associated with it. Once the time stamp expires, the content is evicted from the cache. TTL based caches are preferred because of the ease of implementation, but the biggest challenge lies in determining the expiration time stamp. Recent work [8] outlines an analytical approach to model the performance of TTL based caches.

B. Routing Algorithms

The majority of prior work in ICN has been focused on designing novel caching techniques while keeping the routing simple. One of the simplest routing algorithms is Dijkstra's shortest path routing algorithm, which has been widely used in ICN prototyping and testing. Variations of shortest path routing are widely adopted in the Internet (e.g., OSPF). If the Internet were to be transformed to an ICN, protocols such as BGP and OSPF can be easily used with some minor modifications. We next outline some of the ICN routing algorithms proposed in literature.

- Breadcrumbs: In this protocol [9], each cache maintains a table to keep track of routing information. Each entry in this table, called a breadcrumb, keeps track of the time when a piece of content was downloaded, the ID of the cache that provided the content and the cache to which this content was forwarded. *Interests* are forwarded at each cache based on breadcrumb entries available at that cache.
- Hash-routing: The Hash-routing scheme described in [10] requires both edge routers and core routers within the domain to implement a hash function. Whenever an *Interest* arrives at an edge router, it computes a hash function to determine the router where the content is available. This policy requires that each piece of content be stored at exactly one router (let's say R) within the domain. The *Interest* is then forwarded to R, as returned by the hash function. If the content is available at R, then it is downloaded; otherwise the *Interest* is forwarded to the content custodian. When the content is downloaded from the custodian, there are multiple options. We mention one of the policies, Hash-routing symmetric, where the content is downloaded from the custodian to the user via router R. Router R caches the content as it passes through it on its way to the user. We refer the reader to [10] for a detailed description of the other Hash-routing policies. Note that a Hash-routing policy eliminates the need for routers to maintain per-packet state. However, it requires a hash function to be implemented at each router and allows for only a single replication of any content within a domain.
- Characteristic time routing (CTR): In CTR [11], users forward their *Interests* based on a time-to-live (TTL) based lookup table. The lookup table contains information regarding the content, the cache that can serve this content and the time after which this entry will be purged. The TTL information is populated based on the characteristic time information for a piece of content. Characteristic time for a piece of content in a cache indicates the amount of time in the future a recently accessed content is likely to remain in that cache. In order for users to explore additional paths apart from the direct one to the custodian, the lookup table is populated periodically via local search. Whenever a user generates an *Interest*, it first queries its lookup table to determine if there is an entry in the lookup table corresponding to that content. If there is an entry, the user forwards the *Interest* towards the node obtained from the lookup table, otherwise it forwards it towards the custodian.
- Scoped Flooding: With in-network caching, same content might be available at multiple locations within the network. Therefore, it is possible that a user might find the content it is interested in, in its neighborhood. In [7], the authors propose augmenting the native shortest path routing algorithm with congestion-aware local search to improve performance (CACS). The search begins with the user flooding the *Interest* packet within a specified neighborhood. Any router that has content matching the *Interest* sends a *Reply* back to the user along the reverse path. Each router calculates the available bandwidth of the link it forwards the *Reply* on, and attaches it to the *Reply* message. When the *Reply* reaches the user, it can determine the bottleneck link of the path and the bandwidth available. The user collects all *Reply* messages and then downloads the content along the least congested path (i.e., the path with the maximum bandwidth). If the user does not receive any *Reply* messages within a fixed amount of time, it sends the *Interest* packet to the custodian along the shortest path. In Pro-diluvian [12], the authors study the benefit of scoped flooding for content discovery in ICN. They demonstrate theoretically, as well as through empirical evaluation, that flooding within a small radius helps achieve most of the performance gains. They then determine the optimal flooding radius for two scenarios: i) where a node has prior knowledge of where a piece of content might be available and ii) where the node does not.

- Cache-aware routing (CAR): In CAR [13], the authors propose a dynamic programming algorithm to compute the routes followed by each request from each network node. The algorithm computes these routes based on the transportation cost metric, which is governed by factors such as content popularity, content locality, and the caching strategy used. Their objective is to optimally compute the routes for each request such that the overall network traffic is minimized.
- Potential Based Routing (PBR): In PBR [14], the authors leverage the idea of a potential field to facilitate routing for a desired content. When a node receives an *Interest*, it selects the neighbor that maximizes the potential difference and forwards the *Interest*. In Cache-Aware Target idenTification (CATT) [14], the authors combine PBR and RCO to design a superior forwarding strategy for ICN.

We next compare the existing cache management and routing policies according to the following attributes: on-path caching, alternate-path caching and neighbor search in Table I[1]. In this table, the attribute on-path caching denotes if content is cached at intermediate nodes while it being downloaded along the shortest path, as determined by the routing algorithm. Alternate-path caching denotes that *Interest* packets are forwarded along paths other than the shortest one. While content is downloaded along these alternate paths, it could be cached at en route nodes. The neighbor search attribute indicates if a user leverages its neighbors to discover new paths for a piece of content.

Table I: Comparison of cache management and routing policies

Name	On-path caching	Alternate-path caching	Neighbor search	Mobility Testing
LCE	**Yes**	No	No	**Yes**
LCD	**Yes**	No	No	No
CL4M	**Yes**	No	No	No
ProbCache	**Yes**	No	No	No
LAC	**Yes**	No	No	No
PopCache	**Yes**	No	No	No
CACS	**Yes**	No	**Yes**	**Yes**
Breadcrumbs	**Yes**	**Yes**	No	No
Hash-routing	No	**Yes**	No	No
CTR	**Yes**	**Yes**	**Yes**	No
CAR	**Yes**	No	**Yes**	No
RCO	**Yes**	No	No	No
CATT	**Yes**	**Yes**	**Yes**	No

An important point to note in the case of ICN is the close association between caching and routing. For example, the path along which content is obtained in turn dictates where the content is likely to be available. Though most strategies described so far treat caching and routing independently, the maximum gains in performance can only be achieved by treating both these factors jointly. Therefore, it is necessary to investigate the routing and caching problem jointly to determine the optimal strategy, i.e., path along which a user should forward its *Interest* and the content to be cached at enroute routers.

III. Comparing State-of-the-art Algorithms

In this section, we present results comparing some of the state-of-the cache management and routing policies, namely, LCE, LCD, CL4M and ProbCache caching policies (all of which use Dijkstra's algorithm as their routing algorithm) and Hash-routing, CTR routing (that uses LCE and LRU as its cache insertion and eviction policy, respectively) strategies. We simulate all algorithms using the Icarus simulator, a simulator designed exclusively for

[1] This table is similar to Table I presented in our earlier work [11].

ICN research. The simulation parameters described next have been carefully chosen to highlight the importance of network parameters on performance. Depending on the simulation parameters, one approach can outperform the other, thereby calling for more rigorous experimentation by the networking community to understand the entire parameter space. We refer the reader to [11] for more extensive simulation results.

Our simulation results, though not exhaustive, serve two main purposes: i) provide the reader an understanding of how the different policies compare to each other when simulated with the same set of common assumptions and for the same real world network topology; ii) highlight that no strategy is a clear winner.

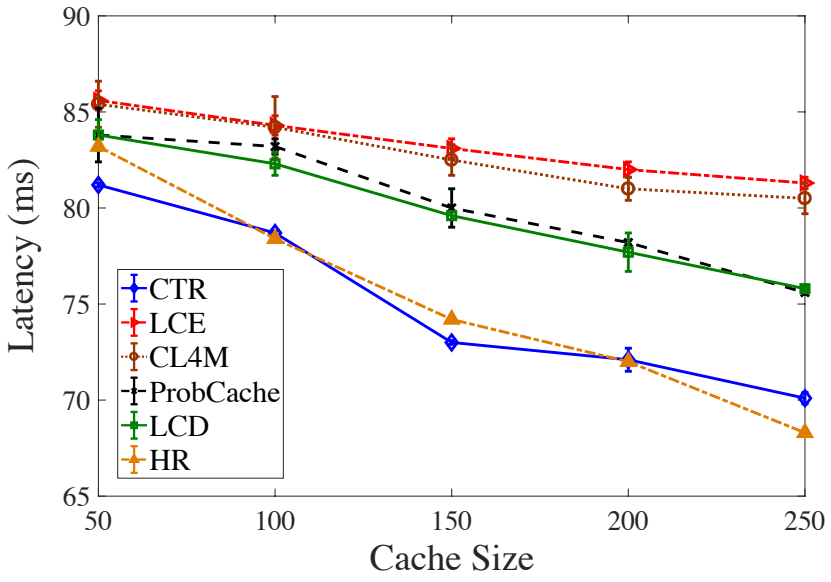


Figure 2: WIDE: Comparing latency performance of different policies with cache size (custodian = 1)

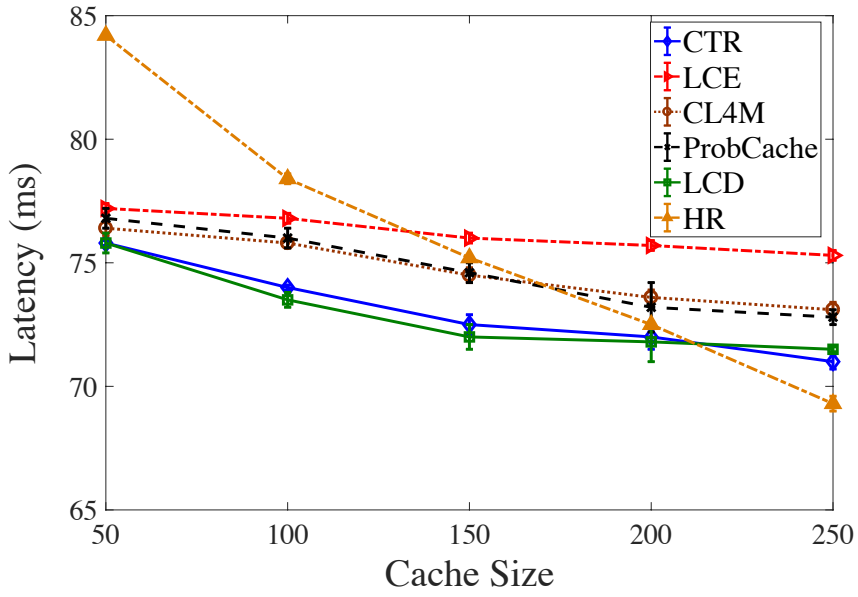


Figure 3: WIDE: Comparing latency performance of different policies with cache size (custodian = 2)

We assume that content universe size is 2000 and that content popularity is distributed according to a Zipfian distribution with skewness parameter α. We compare the latency for the various policies (Figures 2 and 3) by varying the cache sizes of the in-network routers from 50 to 250 for $\alpha = 0.6$. The results reported here are for the WIDE topology. WIDE is the first network established in Japan consisting of 29 network nodes. The performance numbers are calculated after the caches are sufficiently warmed up, so as to eliminate transient behavior. The vertical bars on the graphs indicate the confidence intervals calculated based on 5 runs of the simulation. Figures 2 and 3 are generated for a network with single and multiple (in this case 2) custodians respectively. The content is distributed uniformly at random among the custodians in the multiple custodian case.

We observe from the figures that the performance of CTR, CL4M, ProbCache and LCD is usually better than LCE, that indiscriminately caches content at all routers on the download path. The performance of Hash-routing (denoted by HR in the figures) varies drastically with the number of custodians and cache size. We note that the Hash-routing scheme presented in this article is the symmetric scheme discussed in [10]. Usually the hybrid multicast Hash-routing scheme outlined in [10] has performance comparable to ProbCache and LCD; the superior performance can be attributed to the multicast behavior and comes at the cost of increased overhead. In Figures 2 and 3, we observe that overall, CTR, one of the most recent routing strategies in ICN, performs the best. However, it is difficult to pick a winner among the existing strategies. Our simulations demonstrate that there is no clear ordering between the existing policies. In fact, from Figure 3, we observe that for the multiple custodian case studied here, a relatively simple strategy such as LCD provides the best performance.

IV. Discussion

In the previous section, we observed that a thorough exploration of the parameter space is necessary to understand where one strategy outperforms the other. Most prior work in ICN has been primarily tested on synthetic traces, relying heavily on assumptions such as content popularity following a Zipfian distribution, arrival rates being Poisson and request streams following an Independent Reference Model. Though these synthetically generated traces are useful for verifying a concept, past research has clearly demonstrated that these assumptions do not hold true for real Internet traffic. Therefore, going forward, ICN researchers should evaluate the proposed policies on real-world traffic traces to understand their efficiency. During the past decade, a large number of datasets have been made publicly available to the networking community (e.g., CRAWDAD, University of Massachusetts trace repository).

Recent work in ICN has focused on simple forms of popularity-based in-network caching to improve user performance. These policies focus mainly on the network edge and fail to effectively leverage caches in the network core. In ICN, deployment of network-wide caches is likely to be expensive. Therefore, it is important to design efficient caching and routing policies that maximize cache utilization, both at the network edge and in the network core and minimize unnecessary content duplication. Additionally, most state-of-the-art caching techniques primarily focus on minimizing content delay, which is a relatively simple problem. An equally important, yet largely ignored problem is to design routing and caching algorithms aimed at maximizing throughput. Determining the throughput of a flow is harder in comparison to delay, as throughput is governed by the capacity of the bottleneck link.

Several other open research questions remain. With the growth and deployment of large networks in the Internet-of-Things era comprising of diverse nodes (e.g., mobile nodes, sensor nodes, smartphones) with varying capabilities to connect to the Internet (e.g., WiFi, 4G LTE), questions related to where and what content to cache and where to place computation (e.g., mobile cloud) will become even more important. These diverse nodes will differ in their energy capability (i.e., some might be battery powered while others might be connected to a power outlet) and the amount of cache memory they possess. Therefore, designing routing and caching mechanisms that take heterogeneity and resource constraints into account is an important direction of future research.

Scalability is another important issue that has received little attention in ICN. The ICN architecture will receive widespread adoption only if it can be the demonstrated that the caching and routing algorithms work at scale, and can provide superior performance in comparison to today's host-centric Internet. As network size increases, it is possible that some centralized and cooperative caching and routing policies become computationally expensive, thereby rendering them ineffective. Therefore, it is essential to test the scalability of the proposed routing and caching techniques. The SNAP database, designed by researchers at Stanford, contains examples of multiple real networks of varying sizes. The proposed algorithms can be tested on these networks to study their scalability.

In future, mobility will be a fundamental characteristic of most networks. Uncertainty and change in network connectivity, particularly due to mobility will pose new challenges to ICN researchers. For example, consider a network consisting of two mobile strongly connected components. We assume that these components meet each other at regular intervals. Thus the links that are formed between nodes in these two connected components are tenuous and fleeting. Let us assume that a piece of content (say A) is available in only one of these components. As a result, when the two components meet, it is necessary to cache this content A in the other component as well. To address scenarios like this, it is necessary to identify the 'importance' of content. This importance of content would need to be determined considering factors such as availability of content, quality of network links, mobility of the nodes and future network connectivity. To better understand the importance of mobility on performance, it is imperative to stress test algorithms proposed in future against varying levels of mobility, both pedestrian and vehicular, an aspect that has been largely overlooked so far.

There are several publicly available user request stream traces. Researchers at University of Massachusetts Amherst have made publicly available a collection of traces from a campus network measurement on YouTube traffic. This trace contains information about requests for specific YouTube content. Similarly, the web traffic dataset generated by researchers at Indiana University contains information for 60 million requests per day. This data was collected everyday between Sep 2006 and May 2010 and can be obtained by filing a request.

For greater acceptance of the proposed algorithms and for their widespread adoption, it is imperative that moving forward, the community tests the proposed algorithms on real ICN testbeds. One such effort is the NDN research testbed, that is a shared resource created for research purposes including software routers at multiple participating institutions. However, it is necessary that more such testbeds be deployed at different parts of the world to facilitate rigorous experimentation and testing.

Designing countermeasures against attacks that aim to comprise ICN security and privacy by exploiting caching and routing is relatively unexplored. ICN is vulnerable to cache pollution attacks where an attacker attempts to degrade network performance by polluting caches. Similarly, *Interest* flooding is a routing based Denial of Service (DoS) attack that is unique to ICN. But, in-network caching also presents unique opportunities for ICN to minimize the degradation of service caused by DoS attacks. A DoS attack in a host-centric network makes the server temporarily unavailable, thereby preventing any requests from being served. Contrasting this with the content-centric architecture in ICN, even if a DoS attack results in the content custodian being unavailable, requests for content could potentially be served from other in-network caches that have a cached copy of the content. Designing routing and search strategies for effectively locating content in the network when the custodian is unavailable due to a DoS attack is an interesting research challenge. In ICN, as routers that are independent of the content generators cache content, ensuring content authenticity and privacy is another important, yet largely unexplored problem.

V. Conclusion

In this article, we provided an overview of caching and routing algorithms in information-centric networks (ICN). We described the state-of-the-art cache management (e.g., CL4M, ProbCache, PopCache) and routing (e.g., Hash-routing, CTR) techniques in ICN, outlining their salient features. We compared some of the state-of-the-art policies via realistic simulations and demonstrated that their performance is strongly dependent on network parameters;

therefore, it is hard to pick a clear winner among the proposed techniques. This calls for more rigorous experimentation to understand the entire parameter space. We finally discussed some open research problems and possible ways to address them.

Acknowledgment

We acknowledge the contribution of Bitan Banerjee, graduate student from University of Alberta for his help with the simulations.

References

[1] N. Laoutaris, H. Che, I. Stavrakakis, “The LCD interconnection of LRU caches and its analysis”, Performance Evaluation 63 (7) (2006) 609–634.
[2] W. K. Chai, D. He, I. Psaras, and G. Pavlou, “Cache less for more in information-centric networks,” in proceedings of IFIP Networking 2012.
[3] I. Psaras, W. K. Chai, and G. Pavlou. “Probabilistic in-network caching for information-centric networks”, in proceedings of ACM ICN 2012.
[4] C. Giovanna, L. Mekinda, and L. Muscariello. “LAC: Introducing Latency-Aware Caching in Information-Centric Networks.” in proceedings of LCN 2015.
[5] K. Suksomboon, S. Tarnoi, Y. Ji, M. Koibuchi, K. Fukuda, S. Abe, N. Motonori, M. Aoki, S. Urushidani, S. Yamada, “Popcache: cache more or less based on content popularity for information-centric networking”, in proceedings of IEEE LCN 2014.
[6] D. Ali, B.M. Mirzazad, J.J. Garcia, “Understanding optimal caching and opportunistic caching at the edge of information-centric networks”, in proceedings of ACM ICN 2014.
[7] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, “Congestion- Aware Caching and Search in Information-Centric Networks,” in proceedings of ACM ICN 2014.
[8] N. C. Fofack, P. Nain, G. Neglia, D. Towsley. “Performance evaluation of hierarchical TTL-based cache networks”, Computer Networks, 65(2), pages 212-231, June 2014.
[9] E. Rosensweig, and J. Kurose. "Breadcrumbs: Efficient, best-effort content location in cache networks." in proceedings of IEEE INFOCOM, 2009.
[10] L. Saino, I. Psaras, and G. Pavlou. “Hash-routing schemes for information centric networking”, in proceedings of ACM ICN 2013.
[11] B. Banerjee, A. Seetharam, A. Mukherjee, and M. Naskar. “Characteristic Time Routing in Information Centric Networks.” Vol 113, pages 148-158, Computer Networks February 2017.
[12] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, J. Crowcroft, “Pro-diluvian: understanding scoped-flooding for content discovery in information–centric networking”, in proceedings of ACM ICN 2015.
[13] S. Tarnoi, W. Kumwilaisak, J. Yusheng, “Optimal cooperative routing protocol based on prefix popularity for content centric networking”, in proceedings of IEEE LCN 2014.
[14] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, N. Nishinaga, “CATT: potential based routing with content caching for ICN”, in proceedings of ACM ICN 2012.
[15] G. Zhang, Y. Li, and T. Lin. “Caching in information centric networking: A survey.” Volume 57, Issue 16, Pages 3128–3141, Computer Networks November 2013.

Biography

Anand Seetharam is an assistant professor in the computer science department at Binghamton University. His research encompasses internet-of-things, information-centric networks, wireless networks, and security. He co-organized the IEEE ICME 2017 MuSIC, IEEE INFOCOM 2016 MuSIC and the IEEE MASS 2015 CCN workshops. He has served on the TPC of multiple conferences including IEEE INFOCOM and IEEE ICC and as reviewer for multiple journals including IEEE Transactions on Networking and Elsevier Computer Networks.