Strategic On-Path Cache Placement in Information-Centric Networks

Minseok Kwon, Ravi Kumar Singh, Himanshu Mendhe
Department of Computer Science
Rochester Institute of Technology
{mxkvcs,rk8000,hpm7152}@rit.edu

ABSTRACT

In-networking caching in Information-Centric Networking (ICN) introduces new challenges due to its name-based routing and geographically ubiquitous cache locations. To cope with these challenges, the state-of-the-art techniques cache frequently-accessed content along the route from the original server to the client probabilistically, based on server capacity as well as the benefits from caching the content. It is, however, observed that this approach leaves the core network mostly with rarely-accessed content while popular items are concentrated at the edge network. In this paper, we design an on-path cache placement technique in ICN that places popular items at strategic locations at the core network, so that more clients in widely dispersed areas can benefit from those cached items. Our technique is completely distributed and lightweight to be easily deployed in the real network. Our results show that our caching technique increases cache hit ratio and reduces overall network traffic in comparison to ProbCache.

1. INTRODUCTION

The IP-based addressing mechanism impedes users' capabilities to search content simply by name or keyword. Information-centric networking (ICN) (also known as content-centric networking) has emerged as a way to overcome this hurdle by separating contents from hosts and enabling network routing based on content, not address [12, 26]. One challenge is how to provide in-network caching that facilitates to lower the traffic around the original server and to reduce the delivery time of content to the client [28, 6]. In-network caching has extensively been studied in other contexts, e.g., Web caching [27] and CDN [19]. Yet, those caching techniques cannot be applied to ICN directly as ICN poses new challenges, namely cache transparency, ubiquitous caches, and fine-granular content [29, 30].

In a popular approach to in-network caching in ICN called on-path caching, copies of content are stored at cache servers along the path between the content provider and the client. In early work on on-path caching, each server on the path caches all of the contents that pass through [12], or with a specified probability in order to reduce the amount of redundant data [4]. Content popularity is also considered in moving content closer to the client in [14]. More recently, ProbCache [20] determines data caching probabilistically based on two criteria: available cache capacity and gains from caching data at the server. Here, the gains are computed as a function of the number of hops saved because of cached content. ProbCache is simple, easy to deploy requiring little coordination among servers, and is able to save the original server hit by 20% and the number of hops by 8%. Despite its advantages, ProbCache does not utilize contents off-path much even when those contents are available outside the path between the original server and the client. Motivated by this, off-path caching in ICN is investigated in [24, 25]; however, such schemes often entail extra communication overhead to keep track of cache locations.

In this paper, we show that on-path caching can be designed more efficiently by highly utilizing contents at the core network, while preserving its benefits including optimal delivery paths. One drawback observed with the earlier on-path caching solutions is that popular items are placed at the edge of the network close to clients, while leaving rarely used objects at the core network wasting the cache capacities there. We mitigate this drawback by placing popular contents at strategic locations at the core network serving clients broadly from widely dispersed areas. Specifically, we design a distributed and lightweight algorithm in which each cache server can determine whether it is in a strategic location, and based on that decision, each server can decide whether content should be cached or not. The ultimate caching decision is made probabilistically by considering the content popularity, the cache capacity of the server, and the benefits from the caching. We also design a mechanism that helps maintain consistency between contents cached at different servers.

We evaluate the performance of the proposed algorithm via simulation to measure how often cached content is accessed, how much response time is reduced, and how much traffic is saved overall. Our results indicate that 1) cache hit ratio is increased by 10%, 2) the response time is reduced by 20%, and 3) overall network traffic is reduced by 25%, in comparison to ProbCache. More importantly, the results also reveal that content at the core network is utilized 30% more than ProbCache. We also compare our scheme with off-path caching, and the results imply that ours has significantly lower response time and control information traffic.

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The rest of the paper is organized as follows. In Section 2, we motivate our study with measured data as evidence. In Section 3, we describe our two-level cache placement scheme with its architecture and cache consistency mechanism. Section 4 discusses the experimental setup and results. Finally, we summarize related work in Section 5, and conclude in Section 6.

2. MOTIVATION

In order to understand the effects of on-path caching clearly, we perform experiments on the locations of popular content and the utilization of cached data at the core network. We use the terms cache server and content router interchangeably throughout this paper to refer to intermediate systems that cache content in the network. Our simulation topology consists of 100 content routers connected with each other at random. One content router is connected to $\log n$ other routers where n is the number of routers; hence, a router is connected to around 6-7 other routers. We ensure that all the content routers are indeed connected to the network since some content routers may be disconnected to the network depending on how the topology is generated. We are only interested in where data are cached and their utilization, not file transfer time (or response time), so link bandwidth does not matter and thus is not assigned. A total of 50 clients are connected to edge content routers issuing requests for content every five seconds where one time tick in the simulation is equal to one second.

We simulate two different distributions for content popularity: One is a skewed distribution where only a few contents are popular while the rest is not accessed frequently by the clients, and the other is a uniform distribution in which nearly all contents are requested equally. Each client issues requests following the two distributions independently. Figure 1 shows these two distributions in which only 20% of the contents (popular contents) account for 90% of the total requests in the skewed distribution.

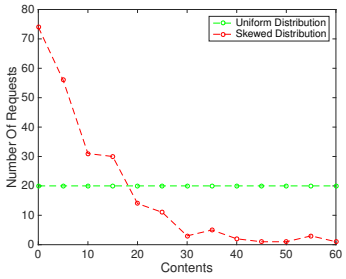


Figure 1: Distributions for content popularity: skewed versus uniform distributions. The x-axis denotes contents in descending popularity in percentage and the y-axis represents the number of requests for the contents

In Figure 2, we show how the locations of content routers that cache data change as content popularity changes. The figure shows that 37% of the contents are obtained from content routers in hop count = 1, and 3% from the routers in hop count = 2, and the rest 60% are retrieved from the original server, which is hop count = 21. It also shows that approximately 60% of the popular contents (from 0 to 20) are cached far from the origin server but close to the clients (hop count = 1 or 2) while unpopular contents (from 20 to 60) are mostly retrieved from the origin server. The results confirm our hypothesis that popular contents are served mostly by content routers close to the clients while the data cached at the core network are left unused, thus wasting their cache space.

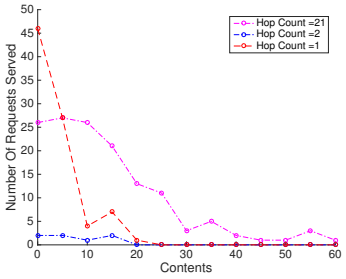


Figure 2: Locations of content routers as content popularity changes.

The results reveal potential drawbacks of the current in-network caching techniques that do not exploit the location information of content routers. We will employ network-aware caching strategies to address these drawbacks. In the following section, we discuss the architecture of our on-path caching technique, and the algorithms for cache placement, replacement, and cache consistency.

3. STRATEGIC ON-PATH CACHE PLACEMENT

In this section, we will give an overview of our network caching architecture including how cache servers are connected, how a content name is identified, how a request is routed to a server, and which content is stored in which cache server.

3.1 Architecture

In our ICN architecture, cache servers (or content routers) are connected, and a content request travels from a cache server to another along the route toward the original server. A content file is located by the name that has a URL-like structure, e.g., $/a/b/c/d$. We assume that the lengths of names are bounded. For routing, we use either an OSPF-like protocol or distance-vector routing without modifications to other components. Packets are forwarded based on longest prefix matching in which a prefix is used as the key to look up the routing table to find the next hop. In the previous example, $/a$, $/a/b$, $/a/b/c$, and $/a/b/c/d$ can be used as prefixes, and if the incoming packet is matched to multiple prefixes, the entry with the longest prefix is selected as the next hop. We can use hashing for the routing table and a Bloom filter to find out name prefix length similar to the mechanism used in [8].

Our mechanism can also be deployed incrementally even when information-centric networking is not fully available. For incremental deployment, cache servers need to be connected as a fully distributed overlay network, and a client needs to know at least one cache server as proxy server to be an entry point to the network. While a router can also serve

as a cache server, it is not recommended with the concern that using the router as a cache server adversely affects the router's core performance such as packet forwarding. Note that DNS can be utilized for this purpose as well. When the DNS server receives a DNS request for the original server (domain name) from a client, the DNS server probes the entry point cache servers, and returns the IP address of the server with the content that is geographically closest to the client. This will allow the subsequent requests to be redirected to the entry point cache server, not the original server.

3.2 Two-level Cache Placement

Content routers compute the best paths to original servers by running a routing algorithm. A request is forwarded along this route starting at the first hop router until the target content is found. In the worst case (or first time), the request is forwarded to the original server that returns the content directly to the client. As the content becomes popular, the server or a content router copies the content file to another content router toward the client, so that the content is delivered to the client faster. The popularity of a content file is computed based on how often and recent the file has been retrieved by clients. Since the clients may access the file from every possible route and interface at the content router, the router needs to keep track of the popularity for each interface (connection to other routers). We compute the average popularity of a given content file based on EWMA (Exponentially Weighted Moving Average) as follows:

$$P(f,t) = \alpha P(f,t-1) + (1-\alpha)Y(f,t) \quad (1)$$

where $P(f,t)$ represents the popularity of file f at time t, $Y(f,t)$ is the number of requests received for f during time $(t-1,t]$, and $0 \leq \alpha \leq 1$.

To mitigate the drawbacks of on-path caching presented in Section 2, we define content routers at two different levels– *hub* and *edge*. Hubs are the content routers located at strategic locations at the core network being able to provide both popular and unpopular contents to potentially a large number of clients. For example, a node of high degree or with many neighbors in the graph constructed as a collection of the shortest paths to the original server is a candidate to be a hub since the node has potential to serve many clients.

The first step of this two-level cache placement is to discover hubs in a distributed manner. Each client initiates the hub discovery by sending a discovery packet toward the original server. As a content router receives discovery packets, the router keeps track of the number that it has received by far. In case the number exceeds a predefined threshold, the router declares itself as a hub, and sends a fresh new discovery packet to its next hop to the original server as the destination. This helps create another hub as the parent of this newly discovered hub, but located sufficiently far. See Figure 3 for this hub discovery. If the number does not exceeds, the router simply forwards packets to the next hop router and the discovery continues. These discovered hubs continue to serve as hub as long as the original server does not change. Edges are the cache servers with content that are not hubs.

Now popular contents are propagated from the original server and are stored in hubs as well as edges. As discussed earlier, when content becomes popular, a copy of the content is transferred to the neighboring router from which requests

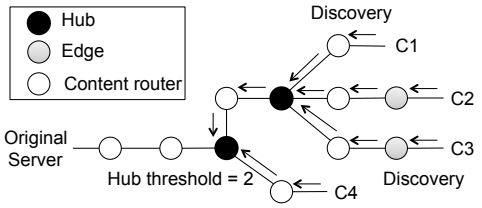


Figure 3: Discovering hubs in the cache network.

have arrived. This way, the content becomes closer to the clients that want the content. In the content propagation, if the content router is a hub, the content is *copied*, which means the content router still keeps a copy; otherwise, the content is *moved*, so that the content router does not keep a copy. In addition, hubs cache all the data passing through, but edges cache the content selectively based on the following probability same as in ProbCache [20]:

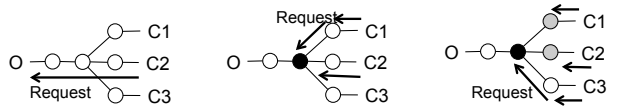
$$Prob(s) = \gamma \frac{x}{c} \quad (2)$$

where $Prob(s)$ is the probability to cache the content at content router s, γ indicates the cache capacity available on the path from s to the client, x is the distance from the original server to s, and c is the distance from the original server to the client, and thus $\frac{x}{c}$ represents the benefit from caching this content locally. The cache capacity (γ) is computed as the ratio of the capacity remaining from s to the client against the total cache capacity. Such cache capacity implies that the cache storage near clients become more competitive. In the end, most hubs and some edges have copies. This facilitates that popular content files are stored in strategic locations, not only on the edge network, and those files can serve a large number of clients along the shortest paths.

Figure 4 illustrates an example of how contents are copied to hubs as well as edges, and requests from clients are forwarded and served. Initially, a request from client $C2$ goes all the way to the original server O since no other content router caches the content as shown in Figure 4(a). As the content router in the middle receives a sufficiently large number of requests from $C1$–$C3$, the router declares itself as a hub, and caches the content locally (Figure 4(b)). Once the router becomes a hub, the router can respond to all of requests from $C1$–$C3$. In Figure 4(c), two edges are created as $C1$ and $C2$ send more requests to the hub. Now the requests from $C1$ and $C2$ do not have to go to the hub, but can be served by those edges, whereas the requests from $C3$ still need to travel to the hub, which is the closest router with the content.

3.3 Cache Consistency

Maintaining cache consistency without incurring high overhead is another challenge, especially when content is frequently updated, to prevent a client from retrieving outdated content. This requires a mechanism for how and when to update, refresh, and remove cache content easily, and even which content to replace when cache space is full. To support cache consistency, the original server as well as cache servers keep track of the version of content stored locally to determine whether the content is the latest. When the content is



(a) Request goes to the original server. (b) Content is cached at the hub. (c) Content is served from the edges.

Figure 4: Example of cache placement in hubs and edges.

updated, the original server forwards the updates to all of its child cache servers. If those servers that receive the updates have the content, they compare the version, update, and forward them to their own child cache servers recursively. In case the servers do not have the content or have already received the latest version, the updates are simply discarded and forwarding stops there.

We use TTL (Time-To-Live) to determine when to discard cached content and to ensure that no child or descendant caches the content if that occurs. When content is created or copied, the initial TTL value is assigned, decremented as time goes by, and the content is discarded when the TTL expires. When a child server reports an updated TTL, the TTL is reset to the maximum of the current TTL and the newly updated TTL of the child. This mechanism assures that the TTL of a cache server is never smaller than the TTL of its children or descendants, and the cache server has always its own copy if a descendant does. The mechanism, however, does not require that the cache server needs to know whether a child cache server has the content and which version. When the server itself experiences changes in file popularity, the TTL is increased in proportion to the popularity.

4. EVALUATION

4.1 Setup

We use ndnSIM [3] to evaluate our strategic cache placement algorithm and compare its performance against other caching methods. The ndnSIM simulator is built on top of NS-3 [2], and provides the NDN routing and forwarding mechanisms including Content Store, PIT, and FIB. Our simulations are performed on a network topology in which two arbitrary content routers are connected with a specified probability. The topology consists of 500 content routers and 1,000 clients that issue requests at a rate of 100–1,000 per second following the Zipf-Mandelbrot distribution [22]. All the simulations are tested on a Ubuntu machine with a quad-core 2.2 GHz processor and 16 GB memory.

We use the following performance metrics for evaluating our caching mechanism:

- **Cache Hit Ratio:** Cache hit ratio is the primary metric that demonstrates the effectiveness of data caching in our method. It is defined as the percentage of responses from cache out of the total responses. The higher cache hit ratio, the more useful the cached data, implying that data retrieval time is reduced.
- **Cache Location:** Cache location shows how far a cached object is from a client when the object is retrieved from cache. It is defined as the number of hops that the object in the cache is away from the client.
- **Cache Eviction:** Cache eviction represents the frequency of cache replacement. This metric is related to cache hit ratio, and indirectly indicates the popularity of content in the cache.
- **Response Time:** Response time is the time elapsed from when a user request is issued until the complete content arrives at the user.

4.2 Results

- cache hit ratio vs. request rate
- cache location (number of hops away from client) vs. request rate
- cache eviction vs. request rate
- response time vs. request rate
- cached data vs. popularity

5. RELATED WORK

The work that comes closest to ours is presented in [24]. In that work, Saha *et al.* investigate the suboptimal behaviors of non-cooperative caching in information-centric networks. They utilize localized off-path caching to reduce the number of duplicates for popular objects while eliminating unused objects usually placed on common paths inside the network. Their results show that spreading data objects widely helps increase cache hit ratio and improves data availability, which is considered more important than hop count in ICN. In their work, however, data are cached at random places, and thus data can be transferred over suboptimal paths requiring the nodes to keep track of multiple paths to find the cached data. ProbCache [20] is concerned with cache placement that promotes the fair sharing of available cache capacity among flows in a decentralized and uncooperative manner. They estimate the caching capabilities of paths, distribute caching space, and determine where to cache data in proportion to the benefits that the cached data would bring. The approach, however, does not place popular content at strategic locations, so that it can be served to many clients reducing redundant copies.

As discussed in Section 1, the earlier work on on-path caching in ICN is studied in [12, 4, 14]. The performance of in-network caching in ICN has been studied in the literature as well [22, 23, 10, 17, 21]. In [22], in-network caching is tested under various conditions, and the authors show that content popularity is the dominant factor that determines the caching performance. In more recent work [23], the same authors conclude that diverse cache sizes do not help improve caching performance as content is cached along the shortest paths. A more realistic mixture of network traffic is used for testing cache performance in [10] showing that caching video content in access networks is more desirable while storing other types of content at the core network.

Caching in ICN is also solved as an optimization problem for various goals including the minimal content retrieval cost [13], the minimal latency experienced by users [16], low energy consumption [7], and reduced traffic [9]. One major drawback of these approaches, however, is that they require

a-priori knowledge of content popularities in order to solve the optimization problems. To remedy this drawback, online caching algorithms are proposed in which content popularity is computed as the algorithm runs [5, 18, 15]. More recently, Gharaibeh *et al.* propose an online caching algorithm for ICN that offers incentives to Internet service providers for caching contents [11].

6. CONCLUSION

We have studied caching strategies in information-centric networks where content is retrieved by its name, not by the address, and possibly from replicas, not the original file. As most popular contents are placed on the edge of the network, frequent replacement occurs, and requests need to travel all the way to the original server in case of cache miss. Our primary interest lies in whether placing content close to clients as well as strategic locations at the core network helps reduce content retrieval time in information-centric networks. Contents are moved closer to the clients as the contents become popular; contents are also cached at the cache servers that are located in a junction point receiving many client requests. We adopt a probabilistic caching algorithm that estimates the popularity and benefits of caching to determine whether caching contents or not. Our results indicate that placing popular content at strategic locations even at the core network help increase cache hit ratio, lower server response time, and overall cached data utilization.

7. FUTURE WORK

We are currently in the process of deploying and testing our caching algorithm in a large-scale network. Testing ICN plus caching is challenging for field testing because content routers are not widely deployed yet in the production network. We are considering two options: 1) using geographically diversely-located cloud data centers, and 2) using a more controllable network testbed, but on a large-scale, e.g., GENI [1]. The testing will shed light on how viable the algorithm is in practice, and expose potential pitfalls.

If content access patterns are predictable, using that information can help increase cache hit ratio even further. The access patterns depend on the location of a client and the original server, application and content types, e.g., highly dynamic, and content locality. Utilizing historical data as well as statistical analysis, we can predict which contents are likely to be accessed in the near future, and cache those contents in strategic locations accordingly. Another important problem is to design a more flexible algorithm for on-path caching that can adapt to route changes even when the changes are frequent like mobile networks. Currently, if routes change, path-dependent contents are at risk to become stale causing many cache misses suddenly. Since a content router keeps track of paths in ICN, it is feasible to cache content more intelligently considering frequently updated path information.

8. REFERENCES

[1] GENI. https://www.geni.net/.

[2] NS-3 Network Simulator. https://www.nsnam.org/.

[3] A. Afanasyev, S. Mastorakis, I. Moiseenko, and L. Zhang. ndnSIM: NS-3 based Named Data Networking (NDN) simulator. http://ndnsim.net/2.1/.

[4] S. Arianfar, P. Nikander, and J. Ott. On Content-centric Router Design and Implications. In *ACM ReARCH*, pages 5:1–5:6, 2010.

[5] C. Bernardini, T. Silverston, and O. Festor. MPC: Popularity-based caching strategy for content centric networks. In *IEEE ICC*, pages 3619–3623, June 2013.

[6] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi. A Survey on content-oriented networking for efficient content delivery. *IEEE Communications Magazine*, 49(3):121–127, 2011.

[7] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson. In-network caching effect on optimal energy consumption in content-centric networking. In *IEE ICC*, pages 2889–2894, June 2012.

[8] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor. Longest Prefix Matching Using Bloom Filters. In *ACM SIGCOMM*, pages 201–212, August 2003.

[9] J. L. et al. Popularity-driven Coordinated Caching in Named Data Networking. In *ACM ANCS*, pages 15–26, 2012.

[10] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *IEEE INFOCOM Workshops*, pages 310–315, 2012.

[11] A. Gharaibeh, A. Khreishah, and I. Khalil. An O(1)-Competitive Online Caching Algorithm for Content Centric Networking. In *IEEE INFOCOM*, April 2016.

[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *ACM CoNEXT*, pages 1–12, 2009.

[13] Y. Kim and I. Yeom. Performance Analysis of In-network Caching for Content-centric Networking. *Computer Networks*, 57(13):2465–2482, Sept. 2013.

[14] N. Laoutaris, H. Che, and I. Stavrakakis. The LCD Interconnection of LRU Caches and Its Analysis. *Performance Evaluation*, 63(7):609–634, July 2006.

[15] J. Li, B. Liu, and H. Wu. Energy-Efficient In-Network Caching for Content-Centric Networking. *IEEE Communications Letters*, 17(4):797–800, April 2013.

[16] Y. Li, H. Xie, Y. Wen, and Z. L. Zhang. Coordinating In-Network Caching in Content-Centric Networks: Model and Analysis. In *IEEE ICDCS*, pages 62–72, July 2013.

[17] L. Muscariello, G. Carofiglio, and M. Gallo. Bandwidth and storage sharing performance in information centric networking. In *ACM SIGCOMM Workshop on ICN*, pages 26–31, 2011.

[18] M. D. Ong, M. Chen, T. Taleb, X. Wang, and V. Leung. FGPC: Fine-grained Popularity-based Caching Design for Content Centric Networking. In *ACM MSWiM*, pages 295–302, 2014.

[19] G. Pallis and A. Vakali. Insight and Perspectives for Content Delivery Networks. *Communications of the ACM*, 49(1):101–106, Jan. 2006.

[20] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *ACM SIGCOMM Workshop on ICN*, pages 55–60, 2012.

[21] E. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In

IEEE INFOCOM, pages 1–9, 2010.

[22] D. Rossi and G. Rossini. Caching performance of content centric networks under multi-path routing (and more). *Relatório técnico, Telecom ParisTech*, 2011.

[23] D. Rossi and G. Rossini. On sizing ccn content stores by exploiting topological information. In *IEEE INFOCOM Workshops*, pages 280–285, 2012.

[24] S. Saha, A. Lukyanenko, and A. Yla-Jaaski. Cooperative caching through routing control in information-centric networks. In *IEEE INFOCOM*, pages 100–104, 2013.

[25] L. Saino, I. Psaras, and G. Pavlou. Hash-routing Schemes for Information Centric Networking. In *ACM ICN*, pages 27–32, 2013.

[26] K. Visala, D. Lagutin, and S. Tarkoma. LANES: An Inter-domain Data-oriented Routing Architecture. In *ACM ReArch*, pages 55–60, 2009.

[27] J. Wang. A Survey of Web Caching Schemes for the Internet. *ACM SIGCOMM CCR*, 29(5):36–46, Oct. 1999.

[28] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys Tutorials*, 16(2):1024–1049, 2014.

[29] G. Zhang, Y. Li, and T. Lin. Caching in Information Centric Networking: A Survey. *Computer Networks Journal*, 57(16):3128–3141, Nov. 2013.

[30] M. Zhang, H. Luo, and H. Zhang. A Survey of Caching Mechanisms in Information-Centric Networking. *IEEE Communications Surveys Tutorials*, 17(3):1473–1499, 2015.