Improving Cache Efficiency of Content-Centric Networking by Using Encoded Addresses

Yuma Kitade
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
y-kitade@ist.osaka-u.ac.jp

Shingo Ata
Graduate School of Engineering
Osaka City University
Osaka, Japan
ata@info.eng.osaka-cu.ac.jp

Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
murata@ist.osaka-u.ac.jp

***Abstract*—Content-Centric Network (CCN) is expected to become the network architecture in the future for exchanging content without specifying the addresses of nodes. An advantage of adopting CCN is that it improves the availability of network resources by using in-network caching. However, since servers that have a lot of popular content may suffer heavy traffic loads, which lead to frequent updates of caches in neighboring routers, the efficiency of cache usage may be degraded. To solve this problem, we propose a method for widely distributing content by using random encoded addresses. From our simulation results, we show that our method can reduce cache sizes by about up to 75% while still achieving the same cache hit ratio when the access frequencies are biased to a specific server.**

Keywords–Future Network; Content-Centric Network; Encoded Addresses; Cache Efficiency.

I. INTRODUCTION

In recent years, Information-Centric Network (ICN) has been considered as a network architecture for the future, and one implementation of ICN is Content-Centric Network (CCN). Today, communication on the Internet is performed by specifying the identifier of the terminal node (IP address), namely, by node-oriented routing. However, recently, most communications are service-oriented, i.e., end users do not care about which node provides a service or content. It is so usual that end users specify some key words of content/service at search engine like Google, then click the URL returned by the engine. There is thus a gap between how the Internet performs routing and how the Internet is used. CCN is expected to solve this problem by implementing routing that uses information about the content, unifying content-oriented communication.

One of the functionalities of CCN is that it can implement routing without being aware of the location of nodes by routing using information about the content. Performance of the CCN can be improved by creating duplicates of content (a *content cache* hereafter) in any router along the delivery path, and by routing to the nearest node that has the desired content. Furthermore, the content cache can relax from spatial and temporal constraints on content by uncoupling the combination of content and location because content can be arranged to be in any location. That is, the content cache makes it possible to acquire content regardless of where the server with the original content is located and whether the server is running.

To increase the advantages of CCN, efficient creation and placement of content caches is important. Since the content is expected to become more diverse and larger in size in the future, the efficiency of the content cache is essential for effectively finding and using network resources. Therefore, efficient cache algorithms for CCN have been studied. Because research into CCN is still in the early days, evaluations of cache performance are mostly at the level of basic studies. For example, in [1] [2], cache performance at the chunk level in CCN was determined by simulation. However, the network topologies examined in that study were limited to the cascade topology and the tree topology.

In recent years, cache performance has been evaluated in more realistic topologies [3]. In [4], the cache performance in a more general network topology was evaluated. In [5], the cache performance was evaluated by using Mixed-Integer Linear Programming problem. In [6], Bernardini et al. proposed a method for caching only popular content. In [7], a method for enhancing cache robustness was proposed. In [8], a collaborative caching scheme to improve cache performance was proposed. In [9], Rossi et al. conducted a thorough simulation considering network topology, multi-path routing, content popularity, caching decisions and replacement policies. In [10], when individual CCN router had different cache size, the caching performance was evaluated. In [11], Rossini et al. evaluated the influence of multiple servers for the same content.

Although variations in the access frequencies to content are taken into account in many evaluations, such as by modeling with Zipf's law, as exemplified by [9], variations in the access frequencies to servers are often not taken into account. Specifically, the content at a given node is typically assigned randomly. In practice, however, the access frequencies to content servers are heterogeneous, just as access frequencies to content are. In short, the servers (web pages) that attract a lot of access are popular because they have a lot of popular content, and the content that is accessed frequently is concentrated in only some servers. When the locality of content is high, cache updates frequently occur along the peripheral paths of servers that are frequently accessed, and the cache performance is thereby greatly reduced.

The aim of this paper is to take both heterogeneities (not only on access frequency of content but on access frequency of node) into consideration. We propose a network architecture for widely distributing content in CCN to increase the efficiency of content caches. More specifically, this method

 ISBN: 978-1-61208-392-6

does not use content names, but instead uses random encoded addresses that are adapted to the addressing architecture of the network layer protocol for content search and routing. Additionally, content is initially placed in the nodes indicated by the random encoded addresses. We find by simulation that our method improves cache performance.

This paper is organized as follows. First, we give an outline of CCN in Section II, and then propose a method for widely distributing content in Section III. Our simulation results and the effectiveness of our method are given in Section IV, and we present our conclusions and discuss future work in Section V.

II. Introduction to CCN

Recently many architectures for implementing ICN have been proposed (e.g., DONA [12], PURSUIT [13], SAIL [14] and COMET [15]). The ICN that we target in our work is based on CCN/NDN [16], which is being studied mainly in the US communities.

Communication in CCN consists of *Interest* packets requesting content and *Data* packets supplying content. Thus, whereas content is referenced by a uniform resource locator (URL) in an IP network, it is referenced by hierarchical content names in CCN. For example, an image of an apple can be assigned a name such as /picture/fruit/apple.png.

CCN routers implement forwarding control by content name, and contain three data structures: the content store (CS), pending interest table (PIT), and forwarding interest base (FIB). These are referenced and updated whenever Interest packets arrive. First, the CS is consulted based on the content name given in an Interest packet. The CS contains the list of content names about the contents cached in the router. If the CS contains the content that an Interest packet requires, the router returns the cached content to the client. If not, the FIB and PIT are used to forward Interest and Data packets respectively. The FIB contains information about the next hop for reaching the target content as well as Internet routing tables. CCN routers search the FIB by using the content names in Interest packets, and forward the Interest packets to the next router based on information about the next hop found in the FIB. It also adds and updates the information about the next router in the PIT in order for the Data packets corresponding to the Interest packet to be delivered to the client correctly.

III. CCN Architecture for Content Distribution

In this section, we propose a CCN architecture for implementing more efficient content distribution. In this architecture, random encoded addresses are first assigned to content, and then the content is placed and Interest packets are routed by using these random encoded addresses.

A. CCN by Using Encoded Addresses

Figure 1 shows an outline of the proposed CCN that uses encoded addresses. In the proposed architecture, routing is implemented by mapping content names into the address space used for routing in the network layer (e.g., IPv6 or IPv4) instead of using the content names themselves as the target addresses of routing. We call this process “encoding,” and call addresses that have been mapped into the address space used for routing in the network layer “encoded addresses.”

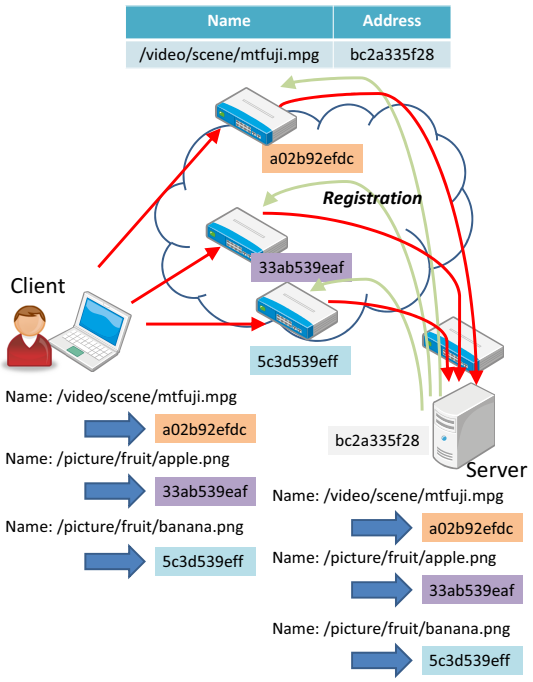

Figure 1. Outline of CCN using encoded addresses

The advantages of routing using encoded addresses instead of content names include (1) the ability to implement CCN without making significant changes to the existing network layer; and (2) the ability to gain additional value from the method of generating encoded addresses. Point (1) means that, if we use IPv4 or IPv6, which are already used on the Internet as the lower layer, and use 128-bit IPv6 addresses as encoded addresses, then we can take advantage of IPv6 routing functions for packet forwarding. Taking full advantage of existing technology is expected to lead to earlier transition to CCN. Point (2) means that, for example, we can use an encoding method to manage the placement and routing of content to be able to widely distribute content by using randomly encoded addresses. The improvement in cache performance by distributing content as described in this paper is achieved by distribution using randomly encoded addresses. More information is given in Section III-B. Also, we note that our architecture has an advantage that the strategy of cache placement is dependent on how to generate the encoded address from the specified content. In other words, we can easily change the strategy by only changing the function of address encoding in the network.

As an example, consider a network layer protocol with an address length of 40 bits, as shown in Figure 1. Now consider the acquisition of a video file named /video/scene/mtfuji.mpg. First, a server that has the object content (a content server) notifies the network that it has the content. We call this “Registration.”

Registration is performed with the encoded addresses of the content. That is, the address `a02b92efdc` is the encoded address of the content /video/scene/mtfuji.mpg, and also means that the network layer address of the node which treats the location of the content and receives the Registration message for the content. Therefore, the Registration message that was

 ISBN: 978-1-61208-392-6

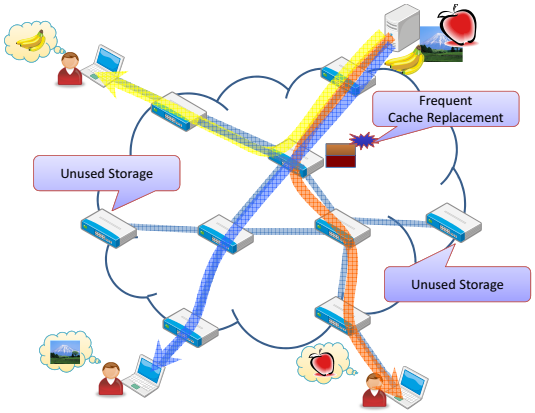

Figure 2. Problem in case of frequently updated caches

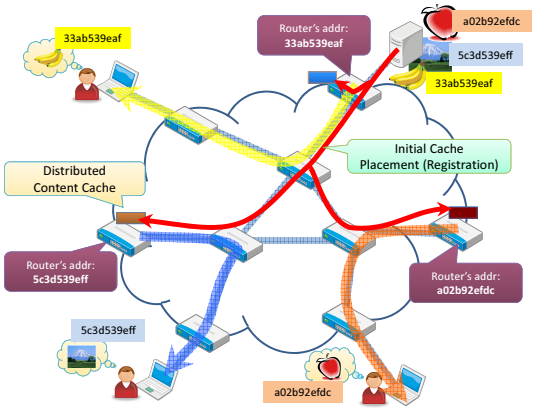

Figure 3. Content distribution with randomly encoded addresses

sent to the address arrives at the node `a02b92efdc` via the network layer protocol. The Registration message contains the content name and the network layer address `bc2a335f28` of the content server. When the node `a02b92efdc` receives the Registration message, it adds the information to a table (i.e., FIB) that maps content names to server addresses (content mapping table).

When a client wants to access the content /video/scene/mtfuji.mpg, it sends an Interest packet to the node that has the encoded address `a02b92efdc` as the network layer address. The Interest packet is sent to the node `a02b92efdc` by the network layer. The node that receives the Interest packet then searches the content mapping table for the content name contained in the message. If the content name is found, the node gets the address of the corresponding content server and forwards the Interest packet to the server. When the content server receives the Interest packet, it sends the content to the client.

B. Content Distribution by Using Random Encoded Addresses

When content that is accessed highly frequently is concentrated on a specific server, the cache updates occur frequently in neighboring nodes, and the efficiency with which the cache is used deteriorates (Figure 2). To solve this problem, we propose a method for scattering the nodes where content is initially placed by using randomly encoded addresses.

We use a random encoding that offers highly random encoded addresses for content names as a method that readily achieves the above-mentioned content dispersion. For example, this random encoding could employ a method of selecting encoded addresses based on a random hash value calculated from the content names by a hash function. Highly random hash functions, such as Secure Hash Algorithm 1 (SHA-1), are the most suitable for the random encoding.

The randomly encoded addresses also retain their meaning as addresses in the lower routing layer. Moreover, since the addresses are dispersed randomly, a random node can be selected independently of the network topology. Conventionally, it has been necessary to know the topological structure to achieve uniform dispersion of content. However, our method is able to obtain the address of a random node by using randomly encoded addresses, and this enables highly dispersed placement of content. Furthermore, we can also support a locality of content access. If encoded addresses are fully randomized by the name of content, contents would be widely distributed as a whole. On the other hand, if a content is preferable to distribute regionally, we can use a random function with preserving some length of prefix.

In the CCN shown in Subsection III-A, the server that has the content (the content server) does not notify (via a Registration message) the network of the existence of the content, but instead initially places the content in the node that has the address `a02b92efdc`, and the content isn't replaced by other contents. This makes it possible not only for the client to acquire content directly without passing through redundant paths, but also to distribute the content. Figure 3 shows an outline of the CCN for content distribution.

IV. Performance Evaluation

In this section, we compare the case where content is distributed in a network by using randomly encoded addresses with the case where popular content is concentrated on a specific server without randomly encoded addresses. In this paper, caching is assumed to be performed in chunks, which are portions of the content, rather than in entire content units. Communication is also performed in chunks.

A. Simulation Scenarios

Performance was evaluated by computer simulation. We use a modified version of ccnSim [9] as the simulator. Specifically, although ccnSim considers the distribution of popularity of content, it also assumes that this content is evenly dispersed across nodes in the network. Thus, because it does not assume that popular content is concentrated on a specific server, which is the subject of this paper, we revised the content placement algorithm in ccnSim so that popular content would be concentrated on a specific server. For comparison, we evaluated cases in which 50% and 70% of content demand is concentrated on a specific server; we also evaluated the case where content is dispersed by using randomly encoded addresses.

The distribution of popular content is implemented by assigning the frequency of demand for each piece of content according to Zipf's law, as follows:

$$f_r = \frac{c}{r^\alpha}, \quad (1)$$

 ISBN: 978-1-61208-392-6

where f is the number of times content is requested, r is the order of popularity of all content, c is a constant, and α is a tuning parameter.

The distribution of popular content greatly depends on the parameter α. In papers that evaluate cache performance in CCN [17] [18] [19], values such as 0.8 and 1.0 are used. In addition, the value of α in DailyMotion is about 0.88 [20]. In this paper, we use 0.9 for the value of α.

We assume that Interest packet generation is modeled by a Poisson distribution with 100 requests/s, that the frequency of requests for each piece of content obeys Zipf's law, and we also suppose that the nodes do not request content which they themselves own. If the content acquisition for a recent request has not yet completed, the same request is not made again.

Intermediate nodes always cache Data chunks they receive that are not already contained in their caches. Chunks are discarded according to the Least Recently Used (LRU) scheme in the event that the cache becomes full.

After the simulation results had become stable enough, we evaluated our simulation. That is, we calculated the performance metrics after the caches of all nodes had become full and the cache hit ratios had converged. The simulation was finished after the cache hit ratio had converged and a certain period of time had elapsed. Simulations were performed five times while changing the seed of the pseudorandom number generator, and the mean evaluation results were considered.

We assume that the average number of chunks is 100. We varied the cache size of the nodes (the number of Data chunks each node can cache) while comparing the case where content is distributed in the network by using randomly encoded addresses with cases where a specific server contains popular content without randomly encoded addresses. Furthermore, to examine the influence of the number of pieces of content, we performed simulation of the cases of 10^5 and 10^6 pieces of content.

A Level 3 Network Topology consisting of 46 nodes [21] [9] was used as the evaluation topology. However, because this topology, shown in Figure 4, consists of only the core network, we used the topology with three end nodes connected for each core node to include the case where end nodes are connected. As a result, the number of nodes is 184 and the graph diameter is 6. The end nodes consist of both servers and clients. The shortest paths by number of hops are calculated beforehand, and the paths between pairs of nodes do not change during the simulation.

Since the efficient utilization of caches is the main topic of this paper, all nodes are assumed to have caches. We also assume that link capacity is sufficient to prevent congestion.

Hereinafter, we refer to nodes that primarily contain content placed using randomly encoded addresses as *repositories*. Our method distributed content across the nodes in a network, and the cache size of each node to have decreased by 10%.

B. Performance Metrics

We use the cache hit ratio and the hop reduction ratio as performance metrics. The cache hit ratio is the probability that the desired content exists in a node on the path to a repository (or server). Among the nodes $1, 2, \ldots, n$, the node n is the repository (or server), and the number of cache hits in node i is H_i. The cache hit ratio Q is then given by (2). In addition, nodes that generate Interest packets first check their own cache for a hit. Interest packets are assumed to hit in node n when they arrive at the repository (or server).

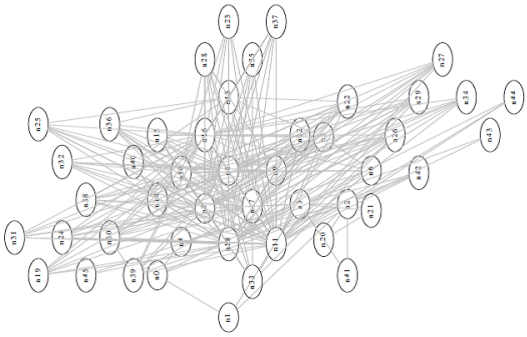

Figure 4. Level 3 topology

$$Q = \frac{\sum_{i=1}^{n-1} H_i}{\sum_{i=1}^{n} H_i} \tag{2}$$

The hop reduction ratio is the mean of the value obtained by dividing the number of hops d through which a Data packet passed by the smallest number of hops P between the node that generated the Interest packet and the repository (or server) that contains the desired content in cases where the node that generated the Interest packet received the Data packet it requested. Nodes are numbered $1, 2, \ldots, n$. The smallest number of hops between node i and node j is $H_{i,j}$. The number of cache hits that occurred in node l for Interest packets generated by node k requesting content c is $D_{c,k,l}$. The repository (or server) of content c is r_c. The total number of Interest packets requesting content c generated by node i is $I_{c,i}$. The total number of Interest packets requesting content c is I_c. The total number of Interest packets I is thus calculated as follows.

$$I_{c,i} = \sum_{l} D_{c,i,l} \tag{3}$$

$$I_c = \sum_{i} I_{c,i} \tag{4}$$

$$I = \sum_{c} I_c \tag{5}$$

The hop reduction ratio of Interest packets generated by node i requesting content c that have a cache hit in node j is given by (6).

$$p_{c,i,j} = \frac{H_{i,j}}{H_{i,r_c}} \tag{6}$$

The average hop reduction ratio of node i generating Interest packets requesting content c is then given by (7).

$$p_{c,i} = \frac{\sum_{j} p_{c,i,j} D_{c,i,j}}{I_{c,i}} \tag{7}$$

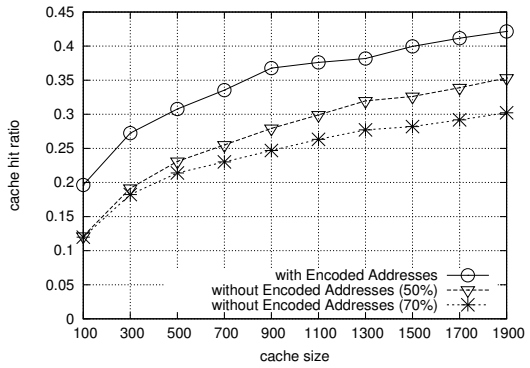

Figure 5. Cache hit ratio with and without randomly encoded addresses ($C = 10^6$)

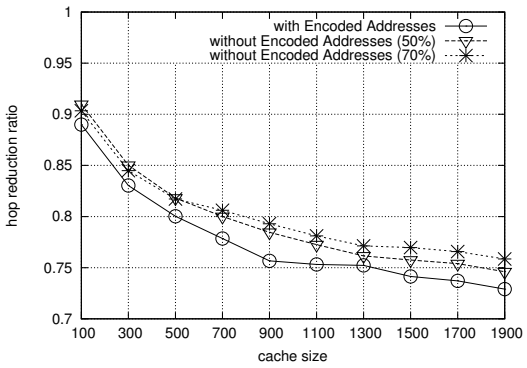

Figure 6. Hop reduction ratio with and without randomly encoded addresses ($C = 10^6$)

The average hop reduction ratio for content c is thus given by (8).

$$p_c = \frac{\sum_i p_{c,i}}{N} \tag{8}$$

The hop reduction ratio is given by (9). In addition, C is the number of pieces of content.

$$P = \frac{\sum_c p_c}{C} \tag{9}$$

Expanding (9) gives (10).

$$P = \frac{I}{NC} \sum_c \sum_i \sum_j \frac{H_{i,j}}{H_{i,r_c} I_{c,i}} \tag{10}$$

The smaller this value, the shorter the response time.

C. Simulation Results

In Figures 5 - 9, the percentage of content requests that are concentrated on a specific server in the "without Encoded Addresses" cases are indicated in the legend. That is, the label 70% indicates that 70% of all requests are concentrated on a specific server. In addition, cache sizes are given in units of content. That is, a cache size 2000 means each node has capacity for 2000 pieces of content. Hereafter, the cache size in units of content is S, and the number of pieces of content is C.

From Figure 5 and 6, it is clear that the cache hit ratio and hop reduction ratio improve when randomly encoded addresses are used. This is because the caches of frequently accessed nodes and their neighboring nodes are not updated frequently because the content was placed in random nodes beforehand by using randomly encoded addresses. Specifically, our method of random encoded addresses improves the cache hit ratio by a maximum 9% over the case where 50% of content requests are concentrated on a specific server and by a maximum of 12% over the case where 70% of content requests are concentrated on a specific server. Moreover, randomly encoded addresses can reduce the required cache size significantly by achieving the same cache hit ratio. For example in Figure 5, the cache hit ratio is 0.3, for the case when the cache size is 500 and encoded addresses is used, which is almost the same when the cache size is 1900 without encoded addresses (70%). In other words, encoded addresses can reduce the cache size to about 1/4 to achieve the same performance on cache hit ratio.

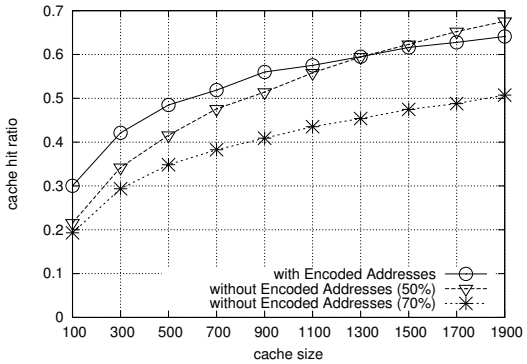

Figure 7. Cache hit ratio with and without randomly encoded addresses ($C = 10^5$)

The hop reduction ratio is improved by a maximum of 3% over the case where 50% of content requests are concentrated on a specific server and by a maximum of 4% over the case where 70% of content requests are concentrated on a specific server. In [17], a scale-free topology of the same order as we used was evaluated, and cache hit ratio and hop reduction ratios were both improved by about 3%. We therefore consider our method to offer significant performance improvement.

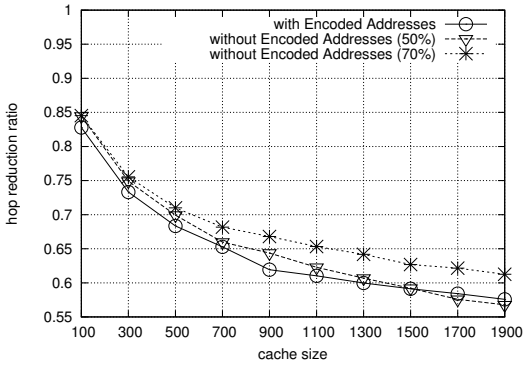

Figure 8. Hop reduction ratio with and without randomly encoded addresses ($C = 10^5$)

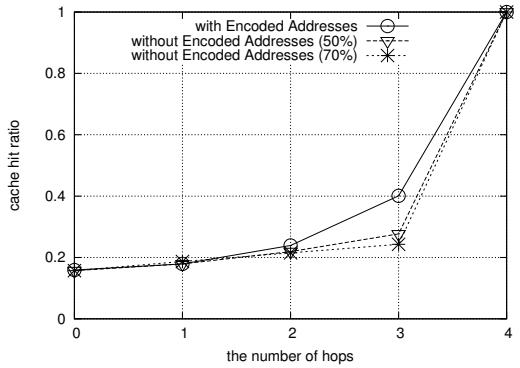

Figure 9. Cache hit ratio every number of hops from clients with and without randomly encoded addresses ($C = 10^6$, $S = 900$, $h = 4$)

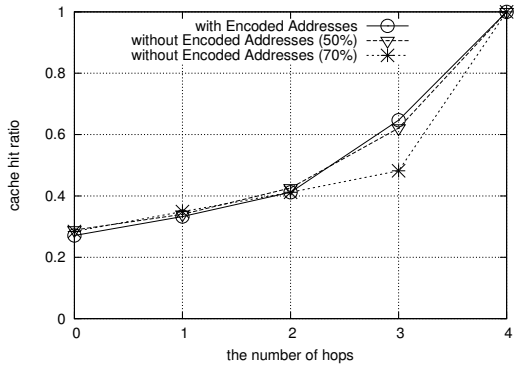

Figure 10. Cache hit ratio every number of hops from clients with and without randomly encoded addresses ($C = 10^5$, $S = 1300$, $h = 4$)

From Figure 7 and 8, for $S = 1300$, our method offers the same performance as the case where 50% of content requests are concentrated on a specific server. To analyze the reasons for this, we assume that the requested content was obtained from the cache of a node on the path to the repository. We therefore calculate the cache hit ratio every number of hops from a client when the smallest number of hops between a client and a repository is h. The number of Interest packets sent to repositories is then R_h, and the number of times content requested by Interest packets was found in the cache of a node that the number of hops from the client is i, is $E_{h,i}$. In addition, when content was acquired from a repository, we treat it as found in the cache of a node where the number of hops from the client is h. That is, the number of times the repository is accessed is $E_{h,h}$. The cache hit ratio $C_{h,i}$ from a client to a node i hops away is then given by (11).

$$C_{h,i} = \frac{\sum_{j=0}^{i} E_{h,j}}{R_h} \tag{11}$$

A comparison of the cache hit ratio every number of hops from the client with and without randomly encoded addresses is shown in Figure 9 for the case where $C = 10^6$, $S = 900$, and $h = 4$ (a case where performance is improved by our method), and in Figure 10 for the case where $C = 10^5$, $S = 1300$, and $h = 4$ (a case where performance was not improved by our method). In these figures, the horizontal axis indicates i, and the vertical axis indicates $C_{h,i}$.

From Figure 9, it is clear that the cache hit ratios of nodes $h-1$ hops away from the client are greatly improved by using randomly encoded addresses. Although caches in neighboring nodes were frequently updated because popular content was concentrated in a specific server, the content distribution by using randomly encoded addresses resulted in a reduction in the frequency of cache replacements. That is, the problem is solved by our method. The improvement in hop reduction ratio is limited to about 4% because the cache hit ratio up to $h-2$ hops remains almost the same. This is because replacement of caches originally occurred only rarely in nodes near the clients.

Figure 10 shows that the cache hit ratios of nodes $h-1$ hops away from the client are greatly improved by using randomly encoded addresses compared with the case where 70% of content requests are concentrated on a specific server, but the cache hit ratio is not significantly improved compared with the case where 50% of content requests are concentrated on a specific server. With $C = 10^5$, and $\alpha = 0.9$, the most popular 1300 pieces of content account for 50% of content requests. That is, when 50% of content requests are concentrated on a specific server, that server contains the 1300 most popular pieces of content. Therefore, for $S = 1300$, because the neighboring nodes can cache most of the content of the server, their caches are infrequently updated. Therefore, our method hardly improved performance in that case.

Consequently, if CCN routers can cache most of the content that is frequently accessed on a server, the caches in nodes near the servers updated infrequently. As a result, our method offers virtually no performance improvement in those cases. However, we expect that the cache size in CCN routers is not sufficient to cache most of the content that is frequently accessed on a server (such as a YouTube server). Therefore, we conclude that our method is useful.

V. Conclusions and Future Work

In this paper, we proposed and evaluated a method for widely distributing content by using randomly encoded addresses in a CCN. Using our method makes it easy to disperse content across a network, and improves cache performance. Furthermore, since our method performs routing by the existing network layer protocol, it is expected to lead to an earlier transition to CCN.

In this paper, we assume that a node that has a random hash value calculated from the content names by a hash function always exists. We need to think a specific method of selecting encoded addresses based on a random hash value (e.g., a case where a node that has calculated encoded addresses doesn't exist).

References

[1] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in Proceedings of 23rd International Teletraffic Congress (ITC 23), Sep. 2011, pp. 111–118.

[2] G. Carofiglio, M. Gallo, and L. Muscariello, "Bandwidth and storage sharing performance in information centric networking," in Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking (ICN-2011), Aug. 2011, pp. 26–31.

[3] G. Tyson et al., "A trace-driven analysis of caching in content-centric networks," in Proceedings of ICCCN 2012, Jul. 2012, pp. 1–7.

 ISBN: 978-1-61208-392-6

[4] E. J. Rosensweig, J. Kurose, and D. Towsley, “Approximate models for general cache networks,” in Proceedings of IEEE INFOCOM 2010, Mar. 2010, pp. 1–9.

[5] S. Wang, J. Bi, and J. Wu, “On performance of cache policy in information-centric networking,” in Proceedings of ICCCN 2012, Jul. 2012, pp. 1–7.

[6] C. Bernardini, T. Silverston, and O. Festor, “Cache management strategy for ccn based on content popularity,” in Proceedings of 7th International Conference on Autonomous Infrastructure, Management and Security, Jun. 2013, pp. 92–95.

[7] M. Xie, I. Widjaja, and H. Wang, “Enhancing cache robustness for content-centric networking,” in Proceedings of INFOCOM 2012, Mar. 2012, pp. 2426–2434.

[8] S. Wang, J. Bi, and J. Wu, “Collaborative caching based on hash-routing for information-centric networking,” in Proceedings of ACM SIGCOMM 2013, Aug. 2013, pp. 535–536.

[9] D. Rossi and G. Rossini, “Caching performance of content centric networks under multi-path routing (and more),” Telecom ParisTech, Tech. Rep., Jul. 2011.

[10] ——, “On sizing CCN content stores by exploiting topological information,” in Proceedings of IEEE NOMEN2012, Mar. 2012, pp. 280–285.

[11] G. Rossini and D. Rossi, “A dive into the caching performance of content centric networking,” in Proceedings of IEEE CAMAD, Sep. 2012, pp. 105–109.

[12] T. Koponen et al., “A data-oriented (and beyond) network architecture,” in Proceedings of ACM SIGCOMM 2007, Aug. 2007, pp. 181–192.

[13] “PURSUIT,” http://www.fp7-pursuit.eu/ [retrieved: 02, 2015].

[14] “Scalable and Adaptive Internet Solutions (SAIL),” http://www.sail-project.eu/ [retrieved: 02, 2015].

[15] “COntent Mediator architecture for content-aware nETworks (COMET),” http://www.comet-project.org/ [retrieved: 02, 2015].

[16] V. Jacobson et al., “Networking named content,” in Proceedings of ACM CoNEXT 2009, Dec. 2009, pp. 1–12.

[17] W. K. Chai, D. He, I. Psaras, and G. Pavlou, “Cache “Less for More” in Information-centric Networks,” in Proceedings of Networking 2012, May 2012, pp. 27–40.

[18] I. Psaras, W. K. Chai, and G. Pavlou, “Probabilistic in-network caching for information-centric networks,” in Proceedings of ICN’12, Aug. 2012, pp. 55–60.

[19] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, “Impact of traffic mix on caching performance in a content-centric network,” in Proceedings of IEEE NOMEN2012, Mar. 2012, pp. 310–315.

[20] Y. Carlinet, B. Kauffmann, P. Olivier, and A. Simonian, “Trace-based analysis for caching multimedia services,” Orange labs, Tech. Rep., 2011.

[21] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with rocketfuel,” in Proceedings of ACM SIGCOMM 2002, Aug. 2002, pp. 133–145.

 ISBN: 978-1-61208-392-6