Content/Location Mapping with Cache-Location Resolution for In-network Guidance

Hiroki Kawabata, Kensuke Hashimoto, Tsutomu Inamoto, Yumi Takaki, Chikara Ohta, and Hisashi Tamaki
Graduate School of System Informatics, Kobe University
1-1 Rokkodai-cho, Nada-ku, Kobe 657–8501
Email: {kawabata, hashimoto, inamoto, yumi, ohta, tamaki}@al.cs.kobe-u.ac.jp

***Abstract*—In recent years, the popularity of large contents such as high-definition video and music is increasing content server loads and the amount of traffic on the backbone networks. To tackle this problem, we propose an Mapping Server with Cache-location Resolution (MSCR), which resolves prospective cache locations (PCLs) close to the requesting user as well as content server location. Sending a query to a nearby PCL enables the user to obtain the content more efficiently. Our simulation shows that MSCR can reduce content server loads and the amount of traffic on core networks.**

***Keywords*-mapping server; cache location; query induction; breadcrumbs; in-network cache.**

I. Introduction

Nowadays, because of the increased demand for large contents such as high-definition videos and music, content servers holding popular contents suffer from high access loads, and core networks also suffer from the massive amount of traffic thus generated. More efficient content delivery is, therefore, one of the most important issues for future as well as current networks.

One traditional solution is web caching, generally content caching. In this context, a content cache means a temporary content copy (content cache) in a user terminal (local cache), content server (web accelerator), or proxy server (cache server). In web caching, if a content query happens to find the appropriate cache on the way to the content server, the content is fetched from there, which reduces content server load and the amount of traffic on core networks. Note that only content caches on the path towards the selected content server are utilized even though other content caches exist just off the path. In this sense, content caches are not utilized efficiently.

Other approaches include the Content Delivery Network (CDN) and Peer-to-Peer (P2P), which are considered as overlay networks based on the current Internet. In recent years, attempts to fundamentally overhaul the Internet architecture have been made in order to optimize it for content-delivery based on “clean slate” approaches [1], [2], [3], [4], [5]. In such “Content-Oriented Networks (CON),” how to identify content, i.e., content naming, how to locate (or find) content, and where and how to store contents are major issues.

In this paper, we propose an mapping server, named “Mapping Server withCache-location Resolution (MSCR)”; designed to utilize distributed cache storages efficiently, it resolves the locations of prospective content caches (PCL: Prospective Cache Location) as well as original servers from content name. Here, a PCL for a certain content may be a node ID or network domain ID of a user who recently got the content.

Currently, the user on-line storage service “Pogoplug” is supported [6]. If part of such storage space is open for public access, content cache storage can become more widely deployed across the network. Further, in the future, routers might actually be equipped with content cache storages (router cache) like Transparent En-Route Caches (TERC) [7], [8] and its improved version called Breadcrumbs (BC) [2]. Based on the above considerations, in this study, we assume that content cache storages are widely distributed across the network and are publically accessible. We also assume that a unique ID (content ID) is allocated to each content. In practice, by some means or another, the user needs to acquire the ID of the content desired. Currently, in order to get the URL (Uniform Resource Locator) of what we want, we most often enter keyword(s) into search engines such as google and yahoo. Some portal sites such as yahoo provide lists of contents and redirect users to actual web sites. On this occasion, they collect users’ access histories. A user who accesses a certain content via such a portal site is expected to have the content, and it is also highly possible that a corresponding content cache exists near the user for some time. By using functions similar to those of search engines and portal sites, MSCR lists candidate contents that users might want to obtain by keyword(s), resolves the content ID, original server location, and PCLs of each content, and collects users’ access histories. As a result, upon sending a query toward a PCL, the user can obtain the expected content; this will lessen access loads on the corresponding content server. Further, if the PCL is close to the requester, the amount of traffic On the core networks might be also reduced. In this sense, the preferred PCL is the location of a past user who is close to the current requester and who has recently accessed the content.

A concern about privacy, however, arises since it is possible for other users to discover who obtained which content. This problem, however, can be mitigated by obscuring PCLs, that is, by using network domains instead of user locations. In particular, the BC scheme is effective even in the case of dim PCLs since a query is guided on the way to the location of the past obtainer.

In this paper, we describe MSCL and conduct simulations to evaluate its effectiveness from the viewpoints of content server access load and the amount of traffic on core networks. Keyword search lies outside the scope of this paper. The remainder of the paper is organized as follows: Section II-A briefly explains the BC scheme. Section III introduce MSCR and the flow of content retrieval. Section IV conducts simulations to confirm the effectiveness of MSCR. Finally, Section V concludes this paper.

II. rredAuxiliary Scheme: breadcrumbs

In this section, we explain the Breadcrumbs scheme[2], which mitigates the privacy problem of MSCR.

A. Breadcrumbs Scheme

The Simple Best-Effort Content Search (S-BEACONS) scheme, described below, is a traditional implementation of BC [2].

In a BC scheme, like Internet Protocol (IP) scheme, a query raised by a requester for a content is transferred toward the (original) content server. Each content is assumed to be allocated a unique content ID. In the BC scheme, routers are assumed to cache not only contents but also their corresponding BCs. Both caches are assumed to have the replacement policy of Least Recently Used (LRU), which discards the least recently used item first.

A BC contains the following information:

- Content ID.
- Node ID from which the content arrived (ID of upstream node).
- Node ID to which the content was forwarded (ID of downstream node).
- Previous content transfer time: most recent time the content passed through the node.
- Previous query transfer time: most recent time the content was requested at the node.

A BC is generated in a router when a content is transferred through the router for the first time, and it is updated every time the content or a corresponding query traverses the node.

1) Query Guidance: In the BC scheme, if a query for a content traverses a router with a BC for the content, the query is diverted to the downstream node of the content which means that it backtracks along the corresponding BC trail. Suppose that a query for a content arrived at time t at a router and found that the content was not cached at the router. Then, with timeout thresholds T_{f} and T_{q}, the router forwards the query downstream if-and-only-if

- The content was cached or refreshed (via successful query) at the router within $[t - T_{\mathrm{f}}, t]$; or
- The previous query passed through the router within $[t - T_{\mathrm{q}}, t]$ and sent downstream.

2) BC Invalidation: If the query cannot find the content on the BC trail and reaches a dead end, the BC trail is regarded as being stale, and the invalidation procedure is invoked for the trail. More precisely, when the query encounters a node with its downstream entry null (i.e., dead end) and the content is not cached there, the query turns back along the trail and deletes the corresponding BCs along the trail.

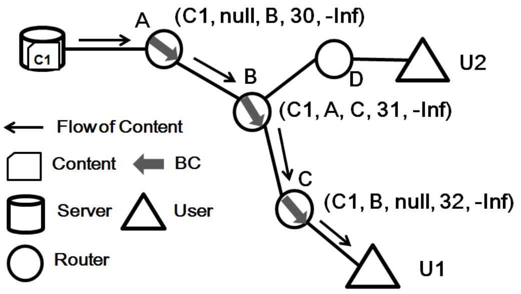

Fig. 1. BC trail and query transfer.

3) BC Update: If a content being transferred finds the corresponding BC in a router, then the BC entries, i.e., the upstream node ID, downstream node ID, and the most recent time the content passed through the node, are updated. If a query for a content finds the corresponding BC in a route, the BC entry of the previous query transfer time is updated.

4) BC Replacement: Since BC cache size is not unlimited, BCs need to have a BC cache replacement policy. In particular, core routers switch huge amounts of traffic, and their extremely rapid switching makes it difficult to provide sufficiently large BC caches. In other words, BC trails tend to frequently fail at the core routers.

5) Example of Query Guidance: Figure 1 shows an example of BC query guidance. From this figure, if content F1 is transferred via routers A, B and C and reaches user U1, BCs are newly generated on the path. Note that the entry of the previous query transfer time is set to $-\mathrm{Inf}$, and the entry of the upstream node ID in router A, to which the server is attached, and that of the downstream node in router C, to which user U1 is attached, are set to “null.” The BC of each router is updated every time the content or a query traverses the router. Next, suppose that user U2 requests the same content and sends its query towards the server. It is then transferred via routers D and B and finds an available BC for the content at router B. In this case, the query is diverted downstream toward router C instead of upstream router A. If the query finds the content on the way, the content is transferred from there instead of the server. This reduces the access load of the server. On the other hand, if the query reaches router C where the downstream BC entry is “null,” BCs are invalidated in the upstream direction from router C toward the server since the content is not cached on the path.

III. Mapping Server with Cache-location Resolution

We detail MSCR in this section.

A. Basic Functions

Like search engines, MSCR resolves a list of contents by the keyword(s) that the user is interested in. The user selects one of the contents from the list, and the selection is passed to

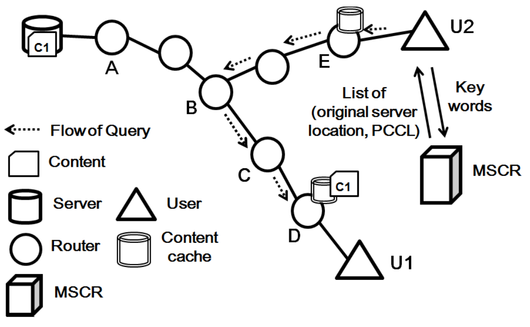

Fig. 2. Query transfer with BC and MSCR.

MSCR. MSCR then replies with the corresponding three-tuple information (content ID, original server location, PCL(s)) to the user, and, at the same time, records the user's node ID (or network domain ID) and the current time in an access history. Here, we assume that MSCR selects and replies with one (or several) proper PCL(s) although it stores multiple PCLs for each content. After getting the three-tuple information, the user sends a query towards the PCL. However, if the user fails to obtain the content desired, the user sends another query toward the original server.

In order to realize the above, MSCR stores up to N_{cont} Content/Location Mapping Information (CLMI) entries whose elements are

- Content ID,
- Original server location (Node ID),
- PCL list,
- Recent request time, and
- Access frequency,

where the PCL list consists of up to N_{PCL} two-tuple elements:

- PCL (Node IDs), and
- Access time.

Thus, MSCR stores information about N_{cont} contents, and up to $N_{\text{cont}}N_{\text{PCL}}$ PCLs. "Recent request time" and "access frequency" are used as the cache replacement policy as mentioned in Section III-B.

Let us explain the above procedure using Figure 2. In this figure, we consider the case that there is only one MSCR in the network, and only access routers have content cache stores.

As a precondition, it is assumed that no node has a content cache and MSCR knows only the original server location of each content, i.e., not PCLs. Furthermore, user U1 sent keyword(s) to MSCR and chose content C1. At that time, MSCR replid (C1's content ID, C1's original server location, null) to user U1 since it had no PCL for content C1. At the same time, MSCR recorded U1's node ID with the access time as a PCL for content C1, and counted up the access frequency for content C1. After getting the information, user U1 sent a query toward the server, got content C1 over path route A through D, and a copy of content C1 was newly cached at access router D.

Next, user U2 also requests content C1. At this time, since MSCR already holds the PCL for content C1 (i.e., user U1's node ID), user U2 obtains a PCL (i.e., U1 node ID) as well as the original server for content C1 from MSCR. To start with, user U2 sends a query toward the PCL (i.e., user U1), and obtains content C1 from router D. Further, a new copy of content C1 is stored at access router E.

B. PCL Replacement

In this study, we assume that an MSCR can have up to N_{cont} CLMIs, each of which can have up to N_{PCL} PCLs since MSCR cannot store unlimited quantities of CLMI. Thus, replacement policies for CLMI and PCL are necessary.

Generally speaking, more popular contents place heavier loads on the content server and core network, since they are more frequently requested. Thus it is better to store their information. In this sense, LFU (Least Frequently Used) is preferable as the CLMI replacement policy.

On the other hand, with PCL, stale information may lead to cache miss since content caches themselves are also replaced. In other words, the newer the PCL is, the better it is. Therefore, LRU (Least Recently Used) is preferable as the PCL replacement policy.

C. PCL Activation Delay

Just because a user gets a reply from MSCR does not mean that the content has already been cached anywhere. This is because there is a delay from when the user sends the query to when at least one copy of the content is cached. To make sure that a user has already obtained a content, and thus the content is cached, the user notifies the completion of content receipt to MSCR. This, however, increases control overhead and the amount of traffic. The simplest and least burdensome way is for MSCR to activate a PCL after some delay even to the cost of certainty.

D. PCL Selection

MSCR stores multiple PCLs in a CLMI for a certain content. Recall that, for a requester, a newer PCL and a closer PCL are more effective in reducing content server access load and the amount of traffic on core networks, respectively. Thus, from the requester's viewpoint, some PCLs are more effective than others. Thus, it is important which and how many PCLs should be returned to the user.

If a requester has routing information, he/she can probably decide which PCL is close to himself/herself more easily than MSCR. In this case, it is simplest for MSCR to return all PCLs to the requester. This, however, increases the amount of control traffic. If MSCR is to judge the closeness between a requester and a PCL, it needs to hold some sort of information on network topology. This is possible if network domain IDs are hierarchically allocated, as assumed in Section IV. Under this assumption, the closeness between two nodes can be roughly estimated. This approach means that the MSCR need return only a few PCLs to the requester.

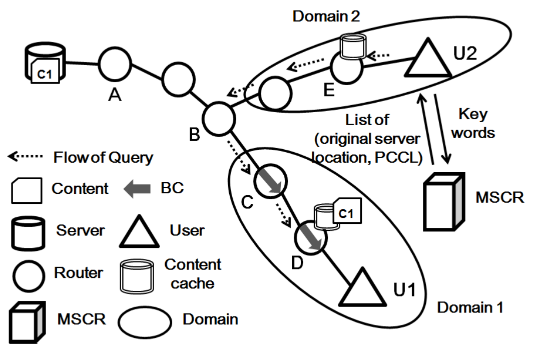

Fig. 3. Query transfer with BC and MSCR.

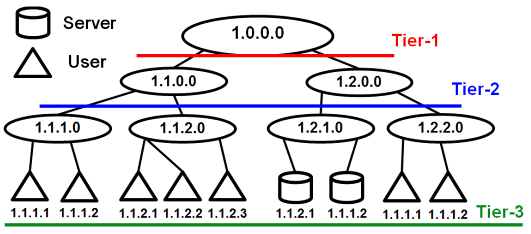

Fig. 4. Example of 3-Tier router topology.

E. Privacy

As mentioned in Section I, privacy concerns arise since other users can know who obtained which content. This concern can be mitigated if MSCR returns network domain IDs instead of user node IDs as PCLs. A query issued toward a PCL, however, may not be able to find any content cache from just the network domain ID. In such a case, in-network guidance by the BC scheme is effective, especially if network domain IDs are hierarchically allocated.

Figure 3 shows an example with in-network guidance by the BC scheme. This figure is almost the same as Figure 2 except that all routers are BC routers and network domain IDs are used as PCLs. Note that, as in Figure 2, only access routers have a content cache storage. As preconditions, we assume that just after user U1 obtained content C1 from the server, a BC trail was established from the server to user U1, and BCs on routers A through B on the trail have been pushed out due to BC cache replacement as mentioned in Section II-A4.

At this point, suppose that, for the request from user U2, MSCR returns the network domain ID of domain 1 as a PCL in addition to the original server location for content C1. Then a query is sent toward domain 1. At the entrance of domain 1, the query happens to find the fragment of the BC trail to router D which has the content cache of the content C1. Note that we assume that a query to find a corresponding BC trail is always guided along the trail. Thus, even vague information on cache location is still useful if in-network guidance by the BC scheme is applied.

F. Unique Content ID

Each content needs to be indexed uniquely, i.e., assigned a unique content ID. For example, a fixed length content ID can be generated by a hash function from the content itself or its URL. A content is at first uploaded to a particular content server and its URL is the only one piece of information related to the content at that time. So when a content is uploaded for the first time, we generate a hash from its URL which uniquely identifies the content thereafter.

G. Implementation

In the above examples, a single MSCR is used. In practice, however, such a centralized system is vulnerable and not scalable. In order to enhance dependability and scalability, we consider that MSCR should be implemented as a distributed system like Chord[9] which utilizes Distributed Hash Tables (DHTs). In our case, CLMI entries will be distributed to multiple servers, which will mitigate vulnerability and enhance scalability. Implementing MSCR in a distributed manner like Chord is left for future work, and in this paper, we evaluate the effectiveness of the basic MSCR function.

IV. Performance Evaluation

In this section, we evaluate how much MSCR can reduce the content server load and the amount of traffic on the core network. To do so, we developed an event-driven simulator in C++. In what follows, we compare four schemes from the viewpoints of cache hit ratio (ratio of the number of contents obtained from caches to that of totally obtained contents) and the amount of traffic in the tier-1 network:

- IP
- IP + Cache
- IP + Cache + MSCR
- IP + Cache + MSCR + BC

In the above, “IP” means that a user fetches a content from its content server as per the server/client model of basic IP networks. “Cache” means that access routers have content caches. If “MSCR” is stated, an mapping server resolves a PCL as well as the original server location. Otherwise, it resolves only a content server from a content name. “BC” means that every router has the BC function and only access routes have content caches.

A. Assumptions

In our simulations, we used 3-tier router topologies, which were created by combining multiple Transit-Stub (TS) topologies (i.e., 2-tier topologies), which are generated by gt-itm [10]. More precisely, in a 3-tier topology, tier-1 corresponds to a transit domain of a certain TS-topology, whose stub domains are replaced by transit domains of other distinct TS-topologies, so that their stub domains become tier-3 domains. Here, we removed links between the routers in the different tier-2 domains (originally set between the routers in different transit domains on the TS-topology). Figure 4 shows an instance of such a 3-tier topology. In this paper, a tier-1 domain is regarded as a core network in a 3-tier topology.

Each node is allocated a sixteen-digit hex node ID separated at every fourth-digit by periods, i.e., “0123.4567.89ab.cdef,” so

TABLE I
GENERAL PARAMETERS.

Item	Value
Link capacity	1 packet/unit time
Number of contents	10,000
Number of tier-1 domains / routers per domain	3 / 3
Number of tier-2 domains / routers per domain	9 / 8
Number of tier-3 domains / routers per domain	144 / 15
Number of servers	50
Number of users	5,000
Query size	1 packet
Content size	100 packets
Content cache size	10
Content cache replacement policy	LRU

that node ID is represented by 64 bits. Node IDs are organized hierarchically to express tier level. That is, the first four-digits express tier-1, the second tier-2, the third tier-3, and the fourth is allocated to a content server or a user. As a result, as shown in Figure 4, all tier-2s have the same first four-digits in their node IDs, and all tier-3s under a certain tier-2 have the same first eight-digits in their node IDs. This kind of address allocation can be realized by HANA (Hierarchical Automatic Locator Number Allocation Protocol) [11]. Here, let us denote the distance between two nodes whose node IDs are I_i and I_j by

$$D(I_i, I_j) = 64 - L(I_i, I_j), \tag{1}$$

where $L(I_i, I_j)$ is the size of the longest prefix match between two node IDs.

In addition to the above, we assume the following: As shown in Figure 4, content servers are located in a particular tier-3 domain, called “server domain,” that is, they are connected to routers in the server domain, while users are sited randomly in the other tier-3 domains. All routers are equipped with BC caches, while only access routes that accommodate users have content caches, since routers in the backbone networks (tier-1 and 2 domains) are expected to have a paucity of high-speed memory even in the future;

The query occurrence interval of each user follows an exponential distribution; Content selection follows a Zipf-like distribution with $\alpha = 0.75$ [12]; Queries and contents are transferred without any packet loss; This time, as we focus on cache hit ratio and amount of tier-1 traffic, not the retrieval delay, each link between nodes is assumed to have capacity sufficient to transfer one packet in each unit time (including a delay for packet processing at routers). Table I summarizes the parameters used in the simulations. With regard to BC parameters, we assume that each router can store up to 50 BC entries in a BC cache, and its cache policy is LRU; the timeout thresholds of BC are set to $T_{\mathrm{f}} = 3,000$ and $T_{\mathrm{q}} = 300$. Table II summarizes the BC parameters used in the simulations. With MSCR, we assume that only one MSCR exists in a 3-tier network. In the case of content/location resolution, MSCR selects one of the nearest (and newer) PCL as well as original server location. In our simulations, PCL is specified by user node ID. MSCR holds up to 100 CLMI entries, and each

TABLE II
BC PARAMETERS.

Item	Value
BC cache size per router	50
BC cache replacement policy	LRU
T_{f}	3,000 unit time
T_{q}	3,00 unit time

TABLE III
MSCR PARAMETERS.

Item	Value
PCL metric	Network distance
PCL selector	MSCR
Number of returning PCLs	1
Number of total PCLs	3,000
Max. CLMI, N_{cont}	100
CLMI replacement policy	LFU
Max. PCL per CLMI, N_{PCL}	30
PCL replacement policy	LRU

TABLE IV
CACHE HIT RATIO AND
RELATIVE AMOUNT OF TIER 1 TRAFFIC IN IP SCHEME.

Scheme	Cache hit ratio	Relative amount of tier 1 traffic ratio
IP	0	100
IP + Cache	1.4	98.8
IP + Cache + MSCR	7.4	94.0
IP + Cache + MSCR + BC	13.0	90.2

has up to 30 PCLs for a certain content. In PCL entries, prospective locations are replaced as per LRU policy, and CLMI entries are replaced as per LFU policy. Table III shows the basic MSCR parameters used in the simulations.

B. Simulation Results and Discussions

Table IV show the cache hit ratio and the amount of tier-1 traffic relative to that in IP scheme. From the table, we can see that MSCR increases the cache hit ratio (i.e., decreases content server access load) and decreases the amount of tier-1 traffic compared with the IP and IP + Cache scheme. This is because MSCR enables users to more effectively utilize content cache storages at access routers other than their own. Moreover, combining the BC scheme with MSCR increases the cache hit ratio and decreases the amount of tier-1 traffic even more. This is because the BC scheme can guide a query for a content whose PCL has not been stored yet in MSCR to a cache.

In the simulation scenario, MSCR is expected to increase the cache hit ratio and decrease the amount of tier-1 traffic by nearly 21.9%. MSCR holds one hundred CLMIs under the LFU replacement policy. That is MSCR tends to store PCLs of the first through the 100th most popular contents, which accounts for 21.6% of content requests given the Zipf-like distribution. If a content cache pointed by a PCL surely exists

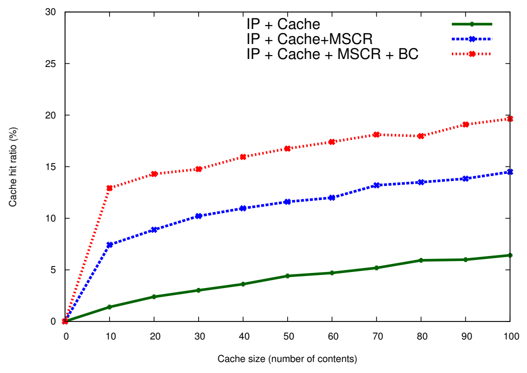

Fig. 5. Characteristics of cache hit ratio as a function of content cache size.

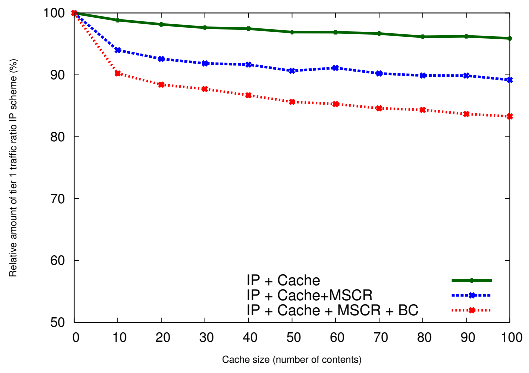

Fig. 6. Characteristics of the amount of tier 1 traffic relative to that of IP scheme as a function of contencache size.

somewhere, the cache hit ratio can be almost 21.6%, which reduces the amount of tier-1 traffic to the same degree.

The reason of relatively poor performance is explained as follows. The content cache policy is LRU, and all users demand contents according to the same, static, Zipf-like distribution. As a result, popular and similar content caches occupy the cache spaces distributed across the network, which are less diversified. This implies that the effective number of PCLs is restricted to basically the content cache size of each access router.

To verify this, we varied the content cache size of each access router from 10 to 100. Figures 5 and 6 show the cache hit ratio and the amount of tier-1 traffic, respectively. In those figures, we can see that the cache hit ratio and the relative amount of tier-1 traffic approach to the expected values as content cache size increases.

In practice, users in different domains are expected to have different preferences due to locality. In this sense, the simulation scenario that content requests must follow the same and static Zipf-like distribution is an overly-stringent condition for MSCR. This can be alleviated by increasing the diversity of content caches among content cache spaces, which could be achieved by cooperative caching between MSCR and the content cache. This is left for future work.

V. Conclusion and Future Work

In this paper, we proposed MSCR that resolves content ID, original server location, and PCL(s) in order to reduce the content server load and the amount of traffic on core networks. Simulation results showed that MSCR is effective in achieving both goals. In our simulations, all users have the same and static preference, i.e., a Zipf-like distribution, for contents, and there is no cooperation between MSCR and the content cache policy. As a result, similar contents are cached in each tier-3 domain. In this sense, it is not so significant for MSCR to guide a query to a different tier-3 domain. In practice, however, preferences will be quite different in different domains due to locality, and will vary over time. In such cases, we expect that MSCR will be even more effective. Verifying this is one of our future works. Moreover, we also investigate establishing cooperation between MSCR and content cache policy.

Acknowledgement

We would like to express our deepest gratitude to Prof. Jim Kurose and Mr. Elisha Rosensweig who provided valuable comments and suggestions. We gratefully appreciate the financial support of Information and Communications Technology (NICT), Japan.

References

[1] J. Choi, J. Han, and E. Cho, "A Survey on Content-Oriented Networking for Efficient Content Delivery," IEEE Communications Magazine, pp. 121–127, March 2011.
[2] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks," Proc. IEEE INFOCOM 2009, pp. 2631–2635, April 2009.
[3] I. Stoica, D. Adkins, S. Zhunang, and S. Shenker, "Internet Indirection Infrastructure," IEEE/ACM Trans. on Networking, vol. 12, no. 2, pp. 205-218, April 2004.
[4] V. Jacobson, D. KSmetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," Proc. ACM CoNEXT 2009, Rome, Italy, Dec. 2009.
[5] T. Koponen, M. Chawla, B. C. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," Proc. ACM SIGCOMM 2007, Kyoto, Japan, pp. 181–192, August 2007.
[6] "Pogoplug," https://pogoplug.com/, April 2nd, 2012 (last accessed).
[7] P. Krishnan, D. Raz, and Y. Shavit, "The Cache Location Problem," IEEE/ACM Trans. on Networking, vol. 8, no. 5, pp. 568–582, Oct. 2000.
[8] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose, "The cache-and-forward network architecture for efficent mobile content delivery services in the future internet," Proc. Innovations in NGN: Future Network and Services 2008, K-INGN 2008, pp. 367–374, May 2008.
[9] I. Stoica, R .Morris, D. Karger, M. F. Kaashek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," ACM SIGCOMM Computer Communications Review, vol. 31, no. 4, pp. 149–160, Oct. 2001.
[10] "gt-itm," http://www.cc.gatech.edu/projects/gtitm/, April 2nd, 2012 (last accessed).
[11] F. Fujikawa, H. Harai, and M. Ohta, "The basic procedures of hierarchical automatic locator number allocation protocol HANA," Proc. the 7th Asian Internet Engineering Conference (AINTEC) 2011, pp. 121–131, Nov. 2011.
[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," Proc. IEEE INFOCOM 1999, pp. 126–134, March 1999.