Local Tree Hunting Scheme to Find the Closest Content in In-Network Cache

Hiroshi SHIMIZU and Masahiro JIBIKI

CCN/ICN with in-network cache capability has been expected as Future Internet. This paper proposes a novel content discovery scheme called Local Tree Hunting for cached contents. It provides almost the same hunting performance as the entire hunting although the hunting area is localized into a tree whose root is the node with caching history. As an option, when caching the state is notified to neighbor nodes to make a short-cut of request forwarding. The effectiveness of the proposed scheme is verified with computer simulation with a Zipf's law content popularity distribution and the Least Recently Used eviction model.

1 Introduction

In the Future Internet, networks with various information processing functions, such as computation and storage functions, have been expected recently, while networks hitherto were mainly designed as a means for transmission of information. Above all, the Information Centric Network (ICN) in which intermediate nodes cache content is in the spotlight[1]-[5]. Content was downloaded from web servers in conventional networks, but now with ICN, it will be downloaded from a caching node near the user, thereby reducing the response time and the transmission links (hop count). Consequently, the issue of how to find desired content from the in-network caches is more important in ICN than how to transmit the content. At present, the IP address of the web server storing the content is resolved from the content name (for example, the URL address) in the Domain Name Server (DNS). However, with ICN, since the contents are widely disseminated to network nodes so as to be dynamically cached and evicted, it becomes difficult to manage the content location centrally by the DNS.

In view of this situation, this paper aims to propose a scheme to hunt and find the requested content without central management. The client, who does not know the content location, requests the content using the content name instead of the location address such as IP address, and the network guides the request to the caching node of the content. Here, in order to obtain a shorter response time and fewer transmission links, it is necessary to find the caching node closer to the request node.

The Content Centric Network (CCN)[2][5] is a prospective ICN architecture, and its characteristics are explained with the help of Fig. 1. The most significant characteristic is that the network nodes have content caching functions, and find the closest caching node using the request signal named Interest with the content name. User R1 sends a request signal. The request signal is forwarded to the web server along the name base path (solid black line) which is the path identified by the content name. Content downloaded from the web server is forwarded in the opposite direction from the path taken by the Interest (dashed black line) and cached in intermediate nodes A, B, and C. The content name which has a prefix structure similar to an IP address enables specifying the path. Next, when User R2 requests the same content, the Interest is forwarded to the closest caching node B along the name base path (solid red line) whose endpoint is User R2 and node B is the download source of the content (dashed red line). Unlike the conventional download scheme from the web

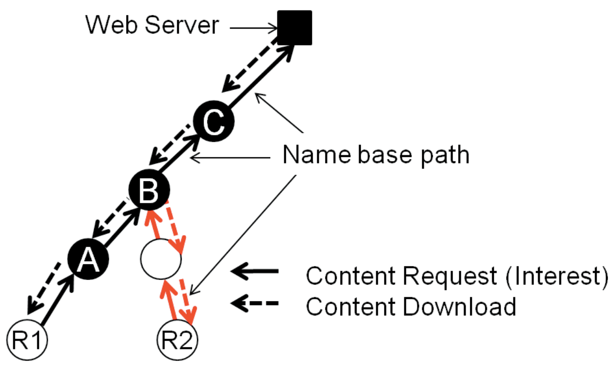


Fig. 1 CCN scheme

server, a shorter forwarding path is obtained. However, this scheme includes the problem that the closest caching node is not always hunted. On the other hand, the scheme of flooding request signals to find the closest caching node is also being studied[6], but it has a major problem that overhead caused by exhaustive hunting increases. This paper proposes a hunting scheme called Local Tree Hunting (LTH), which rivals full hunting even though the hunting area is partial, that is to say, a distributed hunting scheme is used to find the closest caching node in a local tree, and verifies its effectiveness by computer simulation. In Section **2**, the conventional content hunting scheme is described, in Section **3**, the proposed LTH scheme is explained in detail, in Section **4**, the access protocol for hunting is explained, and in Section **5**, the comparative schemes are described and the performance is compared and evaluated by computer simulation so as to verify the effectiveness of the proposed scheme.

2 Conventional CCN/ICN schemes

The typical conventional schemes to be compared in CCN/ICN are explained here.

1) CCN scheme (Fig. 2 (a))[2]: The request signal sent from node R is terminated in the closest caching node C (default response node) on the name base path. And then, following the opposite path from node C, the content is downloaded to node R. The hunting area of the request signal from the caching node is limited to the name base path, thereby minimizing the overhead for hunting. However, as shown in the figure, the closest caching node is not always present on the name base path. For example, caching nodes 1 and 2 which are the closest from node C, are not found.
2) Breadcrumb scheme (Fig. 2 (b))[7]: Breadcrumbs[8] are left showing the history of the path used for forwarding the content in the past, and a request signal is transmitted according to the Breadcrumbs. If node A has a history of storing and forwarding the content in the past, then even after the content is evicted, such history is recorded as a Breadcrumb. In the case of this figure, the request signal from node R is forwarded in the direction of node 1, following the Breadcrumb of node A. The Breadcrumb is overwritten so as to record the most recent trail. Although this scheme presumes that the caching node near the node containing the Breadcrumb is closer to the request node than the default response node[7] (Node C on the name base path in this case), it is not guaranteed that the caching node discovered with Breadcrumbs is closer than the default response node C.
3) Flooding scheme (Fig. 2 (c))[6]: The request signal is flooded to all the neighbor nodes, and this operation is repeated until reaching a caching node. The caching nodes send the response signals to the request node, and then the request node checks the response signals. After identifying the closest node among them, the request node instructs to download. Although the content is exhaustively hunted beyond the name base path, the closest node with the shortest path is found, but the hunting overhead is large.

3 Local Tree Hunting (LTH) scheme

The proposed LTH scheme is for finding the sufficiently closest caching node without hunting exhaustively by utilizing the general behavior that the cached content dissemination area expands as time elapses. The basic and optional schemes are explained in reference to Figs. 3 and 4[8][9].

3.1 Basic scheme

The LTH scheme is classified into LTH (S) and LTH (M). First, the LTH (S) scheme shown in Fig. 3 (a) will be explained. The request node R sends the request including the content name to the name base path leading to the web server in the same way as the CCN scheme shown in Fig. 2 (a). This request signal is relayed by a node which

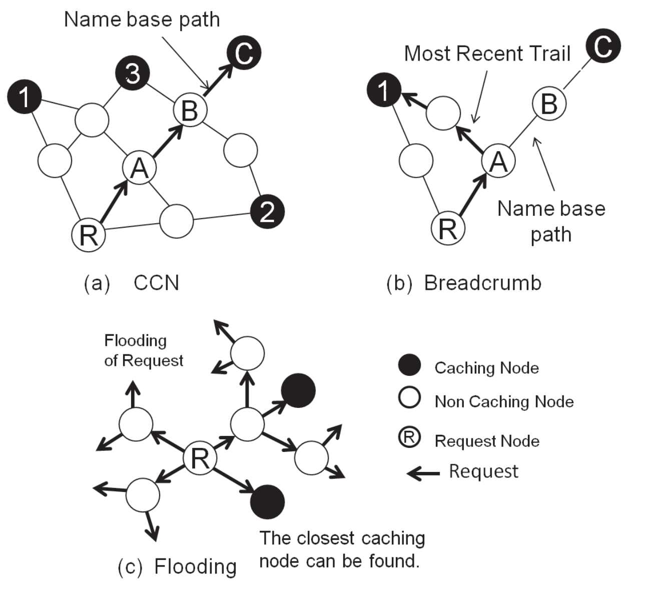


Fig. 2 Conventional scheme

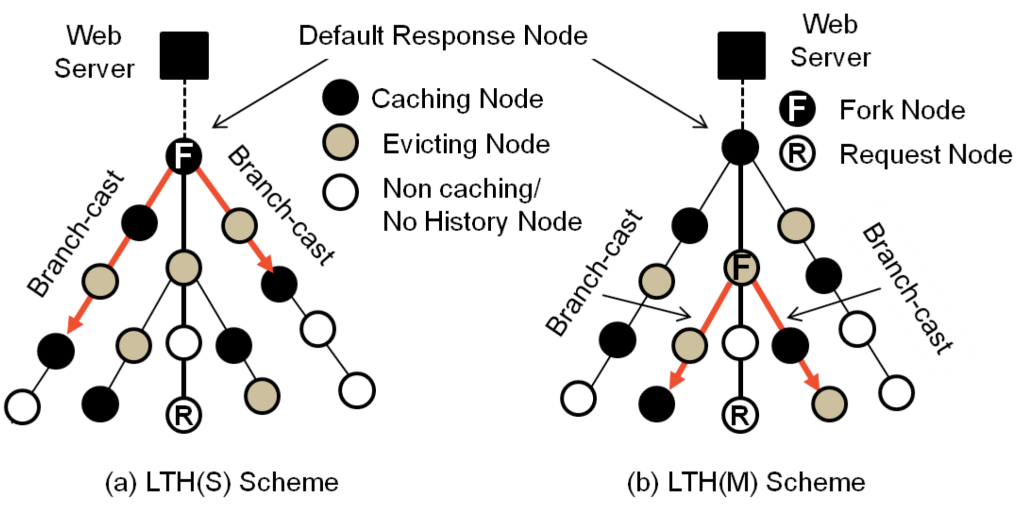


Fig. 3 Local Tree Hunting basic scheme

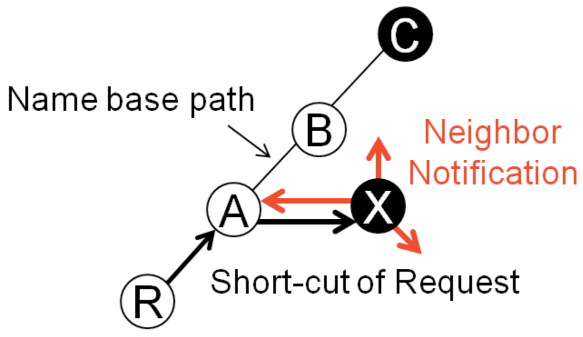


Fig. 4 Neighbor notification control

is not caching the content. The first caching node (default response node) responds to the request, and sends the response signal to request node R in reply. At the same time, this request signal is branch-cast to all the paths present in the forwarding history in the Local Tree whose root is the default response node. This branch-casting node is called the Fork node. The caching node which received the request sends the response signal to request node R. The node in which the requested content was evicted forwards the request according to the history. After selecting the response signal with the smallest hop count among those received, request node R sends the acknowledgment to the sender caching node. In this way, the caching node closest to the request node is found among the response nodes. There are the following three differences from the CCN scheme.

1) The Fork node branch-casts the request signals to the paths based on the forwarding history for the Local Tree.
2) Based on the response signals from the caching nodes which received the request signal, the request node selects the closest.
3) The response signal and content download are forwarded through the shortest path, using the location address of the request node such as IP address.

Next, the LTH (M) scheme will be explained using Fig. 3 (b). The caching node (default response node) which is the closest to the request node on the name base path, responds in the same way as LTH (S). The difference lies in the following. While only the closest caching node can be the Fork node on the name base path in the LTH (S) scheme, the LTH (M) scheme allows the closest node with the caching history to be the Fork node even though it does not cache. The Fork node operation to activate the branch-casting is the same as LTH (S).

Compared to LTH (S), since LTH (M) activates the branch-cast from a node nearer the request node, the Local Tree's size, that is, the hunting area, is smaller. Generally, a wider hunting area allows for a closer caching node to be found. Hence, according to the order of the hunting area breadth, the hunting ability becomes higher. That is, the order of hunting capacity would be Flooding first, then LTH (S), LTH (M), and finally CCN.

3.2 Optional control

Neighbor notification control[10] shown in Fig. 4 is added to enhance the hunting capacity. Node X which newly caches the content notifies this state to the neighbor nodes. When receiving the request signal, node A which has already received the notification from node X forwards it to node X instead of forwarding to node B along the name base path. With this, because the request signal is sent to the node which is sure to cache the content, the download path can be short-cut. Node X works as the Fork node and activates the branch-cast. Since neighbor notification control itself causes a new hunting overhead, two options are considered,

1) NN (AL): The request node and intermediate nodes perform the neighbor notification,
2) NN (RQ): Only the request node performs the neighbor notification in order to reduce the hunting overhead.

4 Access protocol

The LTH protocol consists of the four phases shown in Fig. 5 (a) (Content Request, Response, Acknowledgment, and Content Download). Each phase is identified from the Type field present in the frame header shown in Fig. 5 (b). In each node, a node ID is allocated which indicates the location. Request node R generates the Content Label, which is temporarily mapped to the content name (for example, the URL address). The requested content is uniquely defined by a set of Content Label and request node ID[11]. Although a similar operation was studied in ICN/SDN, the Content Labels were centrally managed[12], whereas in this scheme, each node manages them in a distributed manner.

The access protocol shown in Fig. 5 is classified into four phases and explained below.

1) Content Request: Request signal is branch-casted in the Local Tree, whose root is the Fork node (F). Each node checks the history and controls the forwarding based on the Content Name.
2) Response: Caching nodes in the Local Tree (Fork node and response nodes 1, 2, and 3) send the response signals using the request node ID as the destination address. The response signal is forwarded through the shortest path given by the request node ID.
3) Acknowledgment: The request node checks the hop count of each response signal from the caching node, and replies with acknowledgment through the input port which received the response signal with the smallest count. This acknowledgment is forwarded in the opposite direction of the response signal travelling path, and the caching node which receives it first becomes the download source.
4) Content Download: The content is downloaded through the shortest path given by the request node ID and cached in the intermediate nodes.

The characteristics of this access protocol are listed below.

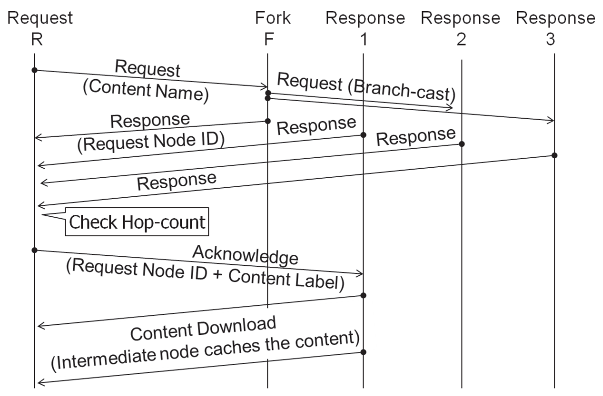
1) Content name is mainly used in the request signal phase.
2) Content Label used for the content download can be managed locally by each node.
3) Content forwarding control is implemented by the set of Content Label + node ID.

5 Performance evaluation with computer simulation

5.1 Evaluation items and evaluation objects

The objective of this research is to find the caching node closest to the request node. In order to verify the effectiveness of the proposed schemes, the download hop count from the discovered caching node, and the hunting overhead which is the sum of the request signal forwarding hop count and the neighbor notification hop count, were evaluated.

The six LTH related schemes of LTH (S) and LTH (M) with no neighbor notification control, LTH (S) + NN (AL), LTH (S) + NN (RQ), LTH (M) + NN (AL), and LTH (M)



(a) Operation Flow

Request Node ID	Content Label	Hop Count	Type	Content Name

(b) Frame Header Format

Fig. 5 Access flow and frame header

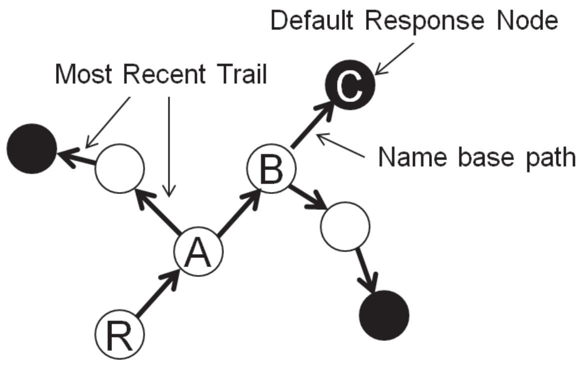


Fig. 6 Multiple Breadcrumbs scheme

+ NN (RQ) which add neighbor notification control (NN), were examined. These six schemes were compared with the CCN scheme and Flooding scheme shown in Fig. 2, and the Multiple Breadcrumbs scheme shown in Fig. 6. In the regular Breadcrumb scheme shown in Fig. 2 (b), only node A controls the Breadcrumbs. In contrast, in this Multiple Breadcrumbs scheme, the Breadcrumbs are controlled by multiple nodes A and B on the name base path which have the cache history. In addition to the Breadcrumbs operation, the request signal is also forwarded to node C (default response node) which is caching the content. This Multiple Breadcrumbs scheme has more hunting capacity than the regular Breadcrumb scheme.

5.2 Simulation models

Figure 7 shows the network model and the evaluation criteria. The network was studied as a future optical network[13]in Japan, in which two nodes were located in Tokyo, and one node each in other prefectures. The web server was installed in node S (Tokyo). Forty-seven other nodes besides node S randomly sent request signals 10,000 times each, and the 12,800-types content catalog was distributed in accordance with the Zipf rule.

$$p\,(i)=(1/i)/\textstyle\sum_{k=1}^{N}(1/k)$$

where N= Content Catalog Size

(Occurrence probability is proportional to the reciprocal of the content ranking order)[14].

The buffer size of each node was 32, 64, 128, 256 and 512. When the buffer was full, the content was evicted in accordance with the Least Recently Used (LRU) policy. Figure 8 shows the accumulated probability in the Zipf rule, in which the first ranking occupies 10% of all requests and the requests until the 85th ranking occupy 50%.

5.3 Simulation result

The simulation result with these buffer sizes of each node on the horizontal x-axis is shown in Fig. 9. When the request node was already caching the content, the download hop count and the hunting hop count were assumed to be zero. Figure 9 (a) shows the download hop count between the hunted caching node and the request node. The performance of the CCN scheme was the worst, followed by the LTH (M) scheme, then Multiple Breadcrumbs (MPLBCR). The Flooding scheme was the best for discovering the closest caching node. Figure 9 (b) indicates the relative download hop count values of each scheme, normalized by that of the Flooding scheme. LTH (M) + NN (AL), LTH (M) + NN (RQ), and LTH (S) schemes achieved almost the same performance. The best is the LTH (S) + NN (AL) and LTH (S) + NN (RQ) schemes with almost the same performances.

On the other hand, Fig. 9 (c) shows the evaluation result of the hunting hop count, which indicates the hunting overhead, given by the sum of request signal forwarding hop count and the neighbor notification hop count. The CCN and LTH (M) schemes had the fewest, followed by the LTH (M) + NN (RQ) scheme, then the Multiple Breadcrumbs scheme. As for the LTH (M) + NN (RQ) scheme in comparison with the Flooding scheme, the hunting overhead could be remarkably reduced to 1/15th to 1/10th allowing only a 5–10% performance degradation in the download hop count. On the other hand, while the LTH (S) + NN (RQ) scheme required the 2.5 times the hunting overhead compared to the LTH (M) +NN (RQ) scheme, the improvement factor of the download hop count was merely 4%. Therefore, the LTH (M) + NN (RQ) scheme would be best as the scheme to find the closest caching node while suppressing the hunting overhead.

Next, we evaluated the performance behavior for content ranking order. The 12,800 different types of contents were divided into 10 groups based on the distribution

Mean Request Frequency per Node	10,000 (Total 470,000)
Content Catalog Size	12,800
Buffer Size	32, 64, 128, 256, 512

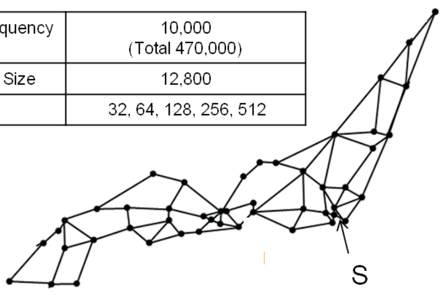


Fig. 7 Network model and specifications

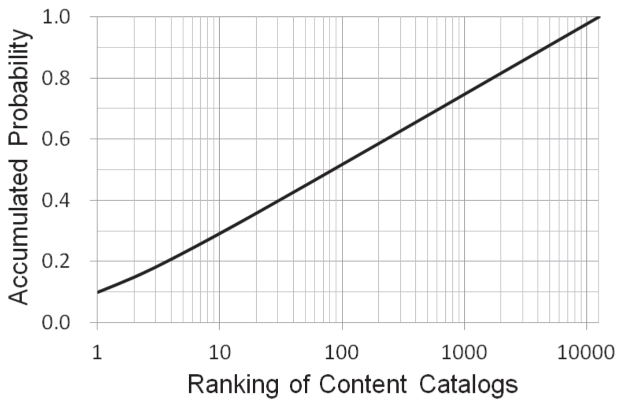
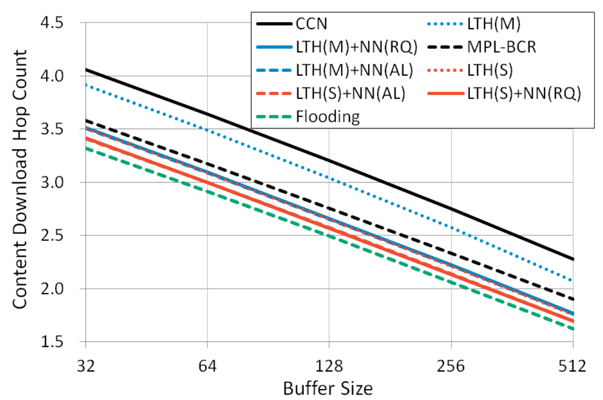
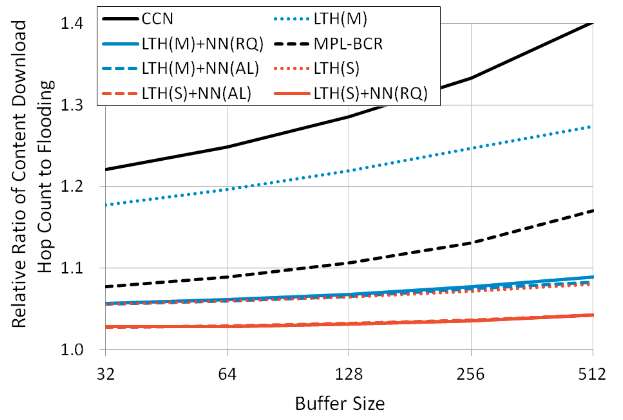


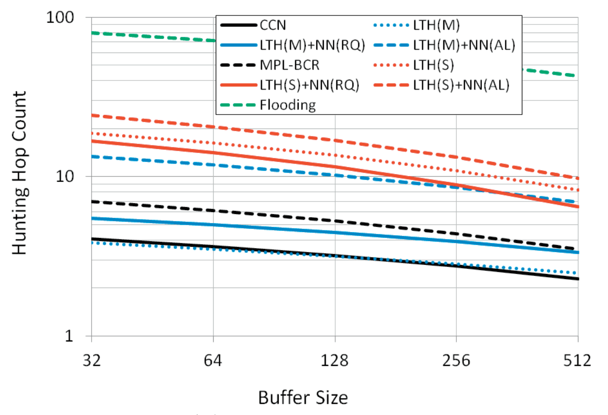
Fig. 8 Request signal probability distribution by Zipf rule



(a) Content Download Hop Count

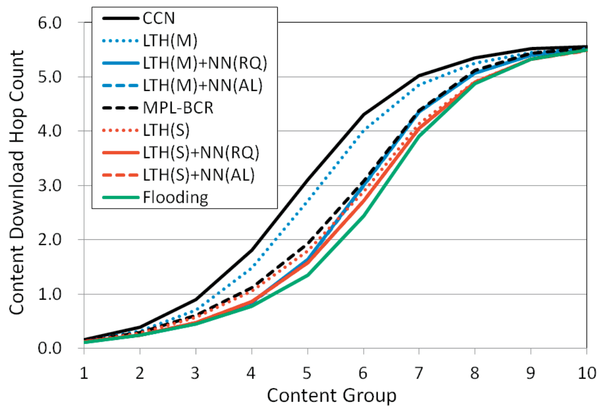


(b) Relative Content Download Hop Count

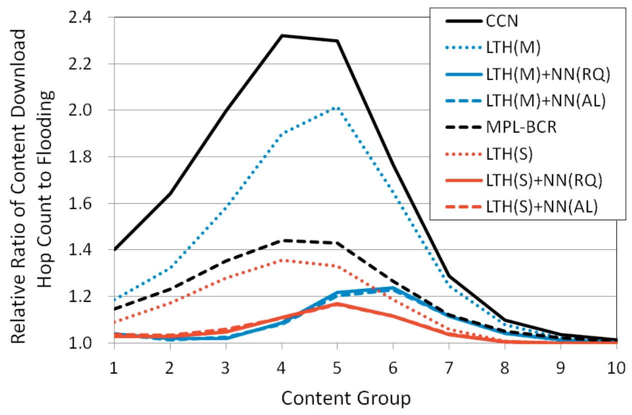


(c) Hunting Hop Count

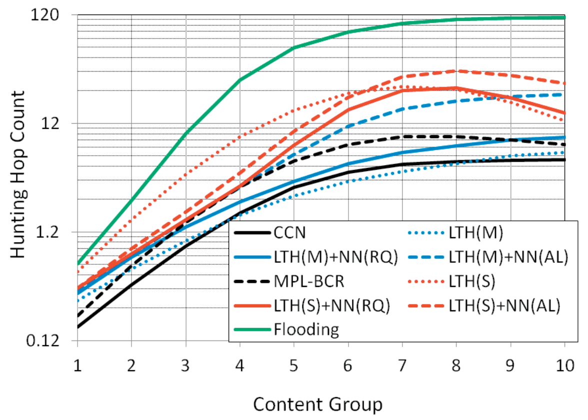
Fig. 9 Performance evaluation result



(a) Content Download Hop Count



(b) Relative Content Download Hop Count



(c) Hunting Hop Count

Fig. 10 Performance evaluation result (buffer size = 128)

shown in Fig. 8, so that the content group request probability for each group would be almost 10%. The first group (Group1) composed of only the first ranking content had 10% probability. And then the next groups of contents 2-4, 5-11, 12-31, 32-85, 86-231, 232-631, 632-1,720, and 1,721-4,693 rankings having 10% probability respectively were made into Group 2, 3, etc. The 10th group (Group 10) was comprised of content ranked at 4,694 and below.

Figure 10 shows the performances in buffer size 128. As shown in Fig. 10 (a), the download hop count performance did not differ greatly depending on the hunting schemes in the first three and the last three ranking groups. In the situation where the higher ranking contents were widely disseminated and the low ranking cached contents were largely evicted in a short time, there was not so much difference among these schemes. In terms of the LTH (M) + NN (RQ) and LTH (S) + NN (RQ) schemes, the download hop count performances were almost the same for the higher ranking groups 1 to 5 which occupy 50% of the total requests.

The recommended LTH (M) + NN (RQ) scheme is summarized below.

1) Out of the nodes containing the cache history, the Fork node (which is the first to receive the request

signal including the content name on the name base path) branch-casts the request signal to the Local Tree whose root is itself, based on the cache history. The caching node, which received the request signal by the branch-casting, sends the response signal to the request node through the shortest path.

2) When the Fork node is not caching, the request signal is forwarded until it reaches the first caching node on the name base path and this node (default response node) also sends the response signal.
3) The request node notifies all the neighbor nodes about the caching state when it was downloaded (neighbor notification).
4) In phase 1), the intermediate node that received the neighbor notification forwards the request signal to this neighbor node (short-cut). The neighbor node works as the Fork node.
5) The request node, which received the response signals from the caching nodes on the Local Tree including the default response node, checks the hop count of these response signals, and sends the acknowledgment to the response node which has the smallest hop count.
6) The response signal and downloaded contents travel through the shortest path specified with the node ID involved in the content name to the request node.

6 Conclusion

As for the scheme to efficiently hunt and find the caching node closest to the request node while suppressing the hunting overhead in the Information Centric Network with in-network caching operation, the Local Tree Hunting scheme with several options was proposed. By computer simulation, the most effective scheme was selected among the options. The location address of the request node was used in the signal frame header information for acquiring the shortest download path.

The problem still remains that the lower ranking contents cause a large amount of hunting overhead although they do not benefit from the caching so much due to the eviction as shown in Fig. 8 (c). We are going to advance the study to resolve this problem.

Acknowledgments

We deeply appreciate the guidance from Dr. Masayuki MURATA, Professor at Osaka University, and Dr. NISHINAGA and the colleagues of the New Generation Network Laboratory.

References

1 http://www.named-data.org/ndn-proj.pdf, “Named Data Networking (NDN) Project,” NDN-0001, Oct. 31, 2010.
2 Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard, “Networking Named Content,” Proc. of CoNEXT 2009, pp.1–12, Dec. 2009.
3 Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman, “A Survey of Information-Centric Networking,” IEEE Communications Magazine, pp.27–36, July 2012.
4 Md. Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, Raouf Boutaba, and Bertrand Mathieu, “A Survey of Naming and Routing in Information-Centric Networks,” IEEE Communications Magazine, pp.44–53, Dec. 2012.
5 H.S. Jeon, I.S. Choi, B.J. Lee, and H.Y. Song, “A Closer Look at Content-Centric Internet Research Projects,” ICACT2012, , pp.698–702, Feb. 2012.
6 Sen Wang, Jun Bi, Jianping Wu, Zhaogeng Li, Wei Zhang, and Xu Yang, “Could In-Network Caching Benefit Information-Centric Networking?” AINTEC '11 Proc. of the 7th Asian Internet Engineering Conference, pp.112–115, 2011.
7 Elisha J. Rosensweig and Jim Kurose, “Breadcrumbs: efficient, besteffort content location in cache networks,” Proc. of IEEE INFOCOM 2009, pp.2631–2635, 2009.
8 H. Shimizu, H. Asaeda, M. Jibiki, and N. Nishinaga, “Content Hunting for In-Network Cache: Design and Performance Analysis,” Proc. of IEEE ICC 2014, pp.3178–3183, June 2014.
9 H. Shimizu, H. Asaeda, M. Jibiki, and N. Nishinaga, “Local Tree Hunting: Finding Closest Contents from In-Network Cache,” IEICE TRANS. INF.& SYST.,Vol.E98-D,No.3, pp.557–564, March 2015.
10 Lijun Dong, Dan Zhang, Yanyong Zhang, and Dipankar Raychaudhur, “Optimal Caching with Content Broadcast in Cache-and-Forward Networks,” Proc. of IEEE ICC 2011, pp.1–5, June 2011.
11 H. Shimizu, H. Asaeda, M. Jibiki, and N. Nishinaga, “Service Expansible LabelFlow/SDN and Information Centric Operation,” IEICE Technical Report IN2013-55, pp.113–118, July 2013.
12 S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, “Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed,” Computer Networks, Vol.57, Issue 16, pp.3207–3221, Nov. 2013.
13 T. Sakano, Y. Tsukishima, H. Hasegawa, T. Tsuritani, Y. Hirota, S. Arakawa, and H. Tode, “A Study on a Photonic Network Model based on the Regional Characteristics of Japan,” IEICE Technical Report PN2013-1, pp.1–6, June 21, 2013.
14 L. Breslau, Pei Cao, Li Fan, G.Phillips, and S.Shenker, “Web Caching and Zipf-like Distributions: Evidence and Implications,” Proc. of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '99, Vol.1, pp.126–134.

Hiroshi SHIMIZU, Ph.D.
Researcher, New Generation Network Laboratory, Network Research Headquaters
New Generation Network, Information Centric Network, SDN

Masahiro JIBIKI, Ph.D.

Research Expert, New Generation Network Laboratory, Network Research Headquaters
New Generation Network, Information Centric Network, Very Large-scale Information Sharing Network