User-side Flooding for Query Distribution in Breadcrumbs-based Content-Oriented Network and its Experimental Evaluation

Christophe Michard*, Yosuke Tanigawa† and Hideki Tode‡
Department of Computer Science and Intelligent Systems, Graduate School of Engineering,
Osaka Prefecture University, 1–1 Gakuen-cho, Naka-ku, Sakai-shi, Osaka, 599–8531, Japan
Email: *michard@com.cs.osakafu-u.ac.jp, {†tanigawa,‡tode}@cs.osakafu-u.ac.jp

***Abstract*—Recently, access loads on servers are increasing due to larger size of content distribution via network. Breadcrumbs, which guide queries to content caches, are designed to reduce server loads and to form content oriented network autonomously in cooperation with cached contents. However, this method only searches guidance information on the nodes which are parts of the query forwarding routes. Therefore, in this paper, we propose the extensive searching control of guidance information called User-side Flooding. Moreover, we implement the extensive control on a real machine, and demonstrate its effectiveness by experiment using the JPN25 topology.**

I. Introduction

Recently, access loads on servers are increasing due to larger size of content distribution via network. Breadcrumbs (BC), which guide queries to content caches, are designed to reduce server loads and to form content oriented network autonomously in cooperation with cached contents. Specifically, each router on a content download path registers guidance information in order to allow later requests for the same content to be guided in direction of the previously downloaded content. However, this method only searches guidance information on the nodes which are parts of the query forwarding routes from the users to the corresponding servers. In this paper, we propose the extensive searching control of guidance information in BC method called User-side Flooding. Each user requesting a content distributes searching queries on a limited flooding area before transmitting the corresponding query to the corresponding server. In addition, we implement the extensive control on a real machine in which the JPN25 topology is constructed by linking several Linux Containers together. Experimental results demonstrate its effectiveness.

II. Breadcrumbs and Active Breadcrumbs

In Breadcrumbs method [1], each router on a content download path makes a BC entry, which is a minimal information allowing to route queries to a content cache, which has previously been created at some user ends. Note that, due to higher feasibility, we assume that only user end—for instance edge router, ONU, STB or user PC—caches contents. When a content query encounters a BC entry for the corresponding content at a router on the way to the server by conventional IP forwarding, the query is forwarded to its neighboring router, to which the content was forwarded previously. This forwarding log is specified in the BC entry. This query guidance, based on BC, is done at each router until the query reaches a user with the intended cache. Thus, through tracing a series of BC entries, each query can follow the corresponding content downloaded previously. This series of BC entries is defined as BC-trail.

In order to guide more queries to caches, we have proposed Active Breadcrumbs (ABC) [2]. In this method, users possessing cached contents actively distribute ABC entries to neighboring nodes in order to guide queries reaching nearby routers. When a content query reaches a router with an ABC entry for the corresponding content, the query is forwarded to the user-cache specified in the ABC entry, based on IP routing.

III. Distribution Control of Query and Guidance Information

In BC method, queries search guidance information (BC) on the routes between users and servers.

In contrast, we study distribution control of queries whose distribution areas are expanded from lines to plain surfaces. While ABC [2] corresponds to the distribution of guidance information from user-side edge with cache, this distribution is originated from users transmitting queries. In the following, we describe this new distribution method.

A. User-side Flooding: Query distribution from users

In the method called User-side Flooding, each user requesting a content distributes searching queries on a limited flooding area before transmitting the corresponding query. This corresponds to (1) in Fig. 1.

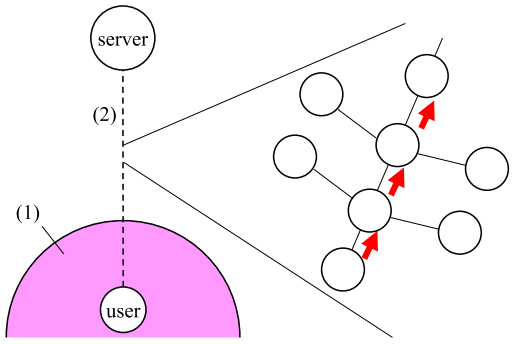

Fig. 1. Generalized query distribution

The searching queries are distributed to neighboring nodes, within a range of H_F hops from the user. They are looking for a cache or a BC. If a receiving node is caching the content or has a valid BC, it sends a reply to the user in order to announce itself. When the user receives this reply, it can send the content request directly to this node. The first reply is chosen by the user, others are then ignored for simplicity, though we can easily change the policy.

In the case no reply occurs, we designed a simple time out delay in order to eventually send the original content request towards the content server.

B. Design details of User-side Flooding

We assume that the user already knows the content server location and that the content is not already cached by the user. User-side Flooding occurs just before the transmission of the content request to the server.

1) The user distributes searching queries to the neighboring nodes within a range of H_F hops (=TTL). At this point, the user sets up a timer in preparation to the case no reply occurs. A unique identifier needs to be created, and information of the original content request needs to be stored;
2) When a node receives a searching query, it checks the existence of the content in its cache as well as the existence of a valid BC corresponding to the content. If either the content or a corresponding BC is found, the node sends a searching reply to the user, containing its routing information in addition to the flooding id. If the node is a router, if neither the content nor a BC is found and the query has not reached H_F hops (=TTL) yet, the searching query is transferred to all the other adjacent nodes;
3) If the user receives a reply before the time out, a special content request is sent directly to the replying node. From the user to the replying node, the request is simply forwarded by the intermediate nodes. This content request contains the same information as the original one, except for the routing information needed to be transmitted to the replying node. The timer and flooding information are deleted on the user side. If another reply reaches the user, it will be ignored;
4) When the special content request reaches the replying node, cached content and BC presences are checked again in case of invalidation. If invalidation occurred, the content request is simply forwarded to the original content server, checking the presence of content cache or BC along the route;
5) If the time out expires before the reception of a reply, the original content request is sent. The user deletes the timer and the flooding information. In the case a reply reaches the user after that, it will be ignored.

IV. Experimental Evaluation of the User-side Flooding

We implemented User-side Flooding and compared it to the classic Breadcrumbs method. We used a FUJITSU PRIMERGY TX300 S6 server equipped with 24 Intel Xeon X5675@3.07 GHz CPU units and 64 GB of DDR3 RAM memory for the experiment. We constructed the JPN25 topology as shown in Fig. 2 by linking several Linux Containers (LXC)—corresponding to routers, user terminals, content severs, and one mapping server—together. This topology links 25 routers; the maximum distance between users is 10 hops. The mapping server is located in Tokyo-East (node 131). Each node is provided with 50 Mbps network interfaces. Link latency due to distance is not implemented. Each router is connected to a LAN consisting of 2 users and 1 content server. Only the users cached the contents (cache size = 50). BC lifetimes were set to $T_f = T_q = 30$s. The content generation distribution followed a Zipf law with $\alpha = 0.8$. Users generated a request every second.

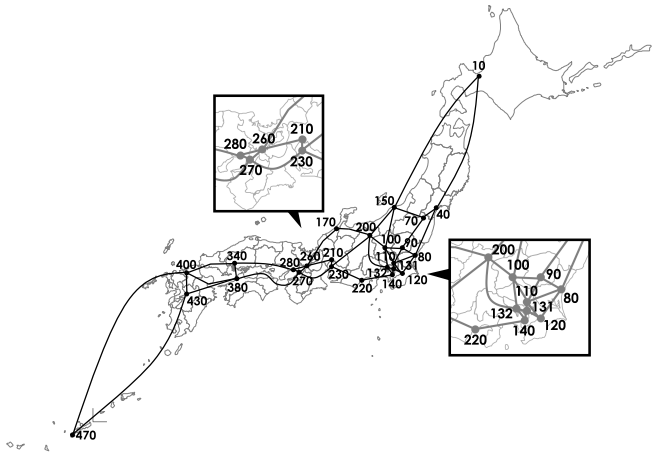

Fig. 2. JPN25 topology

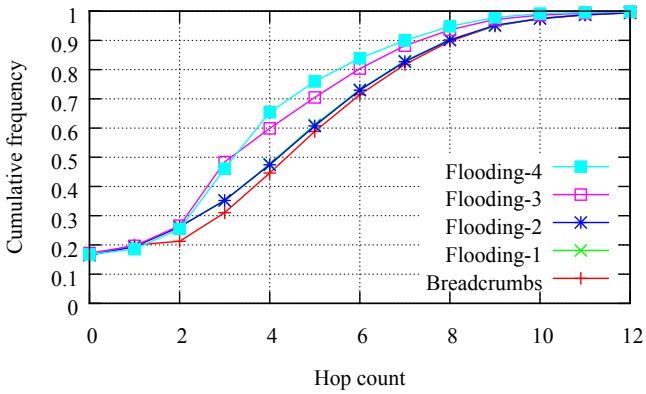

Fig. 3. Number of hops for a content request to reach its destination node

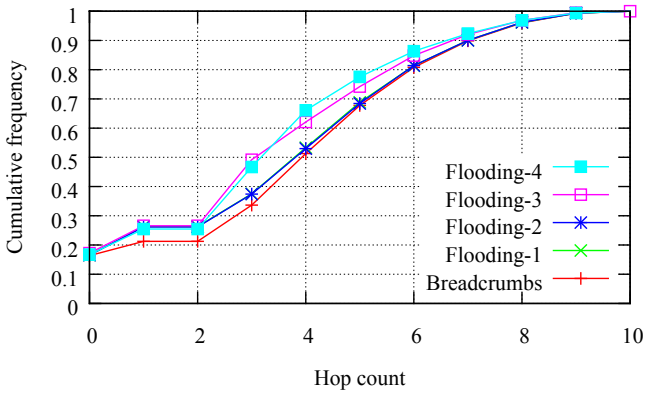

Fig. 4. Number of hops required to transfer the content to the requesting user from the server or the cache node

Each content server registered 100 contents, setting the total number of contents to 2500. Content size was fixed to 10 KB. The duration of the experiment was set to 300 seconds. As for User-side Flooding, we varied H_F from 1 to 4 hops, and the time out was set to 50 ms.

We measured the number of hops needed for a content request to reach the node containing the content as shown in Fig. 3 and Table I, and the number of hops needed for a content

TABLE I. NUMBER OF HOPS FOR A CONTENT REQUEST TO REACH ITS DESTINATION NODE

Proposal	min	mean	max	variance
Breadcrumbs	0	4.75096	18	339672
Flooding-1	0	4.5685	19	314082
Flooding-2	0	4.58306	19	316086
Flooding-3	0	4.01056	17	242050
Flooding-4	0	3.86897	16	225261

TABLE II. NUMBER OF HOPS REQUIRED TO TRANSFER THE CONTENT TO THE REQUESTING USER FROM THE SERVER OR THE CACHE NODE

Proposal	min	mean	max	variance
Breadcrumbs	0	4.22445	10	268558
Flooding-1	0	4.04292	10	245972
Flooding-2	0	4.05329	10	247235
Flooding-3	0	3.70897	10	207014
Flooding-4	0	3.67056	10	202750

to be transferred from the server or the cache node to the user as shown in Fig. 4 and Table II. We also measured for each user how many times our proposal successfully found the content in a cache or a valid BC as shown in Table III, and how many content requests were sent by the users, as shown in Table IV, in order to appreciate the previous table (no request is sent if the content is already cached in the requesting node).

Fig. 3 shows that when $H_F = 1$ (Flooding-1) or $H_F = 2$ (Flooding-2), the User-side Flooding method does not show significant improvement compared to the classic Breadcrumbs one. About 5 more percents of requests reached the destination node within 2 hops in these cases. By contrast, $H_F = 3$ (Flooding-3) or $H_F = 4$ (Flooding-4) proposals show significantly improved results. Almost half of the requests reached their destination within 3 hops in the Flooding-3 (48%) and Flooding-4 (46%) cases, compared to the 31% of the Breadcrumbs method. If we consider a range of 4 hops, Flooding-4 shows even better results (65%) compared to Flooding-3 (60%) and Breadcrumbs (45%).

Fig. 4 shows similar results. Flooding-1 and Flooding-2 show little improvement within a short range. Since users are connected via LAN networks and cache is disabled in the routers, no content can be transmitted in exactly 2 hops. Almost half of the contents are downloaded within 3 hops of the requesting nodes in the Flooding-3 (49%) and Flooding-4 (47%) cases, compared to the 34% of the Breadcrumbs method. This is huge, since 3 hops means that contents are retrieved at the next location on the topology (user – router – router – user). Within 4 hops, the Flooding-3 (62%) and Flooding-4 (66%) proposals still prove significant results compared to the Breadcrumbs (51%) one.

Table I shows that content requests reach their destination node in about 4.01 hops in the Flooding-3 proposal and 3.87 hops in the Flooding-4 proposal, compared to 4.75 hops in the Breadcrumbs method, which is a great improvement. Table II shows that contents are downloaded at a distance of about 3.71 hops in the Flooding-3 proposal and 3.67 hops in the Flooding-4 proposal, compared to 4.22 hops of the user in the Breadcrumbs method. In both cases, the Flooding-1 and Flooding-2 methods show similar results and only little improvement compared to the Breadcrumbs method.

TABLE III. USER-SIDE FLOODING: NUMBER OF REQUESTS WHICH FOUND A CACHE OR A VALID BC

Proposal	min	mean	max	variance
Flooding-1	12	20.36	29	20295
Flooding-2	14	20.28	29	20138.2
Flooding-3	50	75.1	106	276225
Flooding-4	85	112.1	147	615556

TABLE IV. USER-SIDE FLOODING: NUMBER OF CONTENT REQUESTS SENT

Proposal	min	mean	max	variance
Flooding-1	230	251.2	273	3.0919e+06
Flooding-2	235	251.3	270	3.09436e+06
Flooding-3	231	249.18	263	3.04239e+06
Flooding-4	234	250.62	268	3.07766e+06

Tables III and IV show that out of an average 251 User-side Flooding requests, only 20 ones (8%) found a cache or a valid BC in both the Flooding-1 and Flooding-2 cases. Improvement can be observed in the Flooding-3 one, where 75 requests out of 249 (30%) were successful on average. The Flooding-4 proposal shows the best results with 112 requests out of 251 (45%) which found the information.

Both Flooding-3 and Flooding-4 show great results, the latter strictly proving a little bit superior. Nevertheless, we should be careful. Increasing H_F implies more network traffic and consequently increases the CPU loads of the routers, which was observed during these experiments. At $H_F = 5$ (not showed here), the results showed almost no improvement for a sudden increase of the CPU loads, which is harmful for the network performance. Flooding-3 showed almost no load for great results and little traffic. This would be the preferred choice for networks with little resources.

V. CONCLUSION

We proposed a method to allow the users to search contents actively on promiscuous nodes. We showed its effectiveness by comparing it to the Breadcrumbs method, and demonstrated how the distribution range H_F has a positive impact on the results.

ACKNOWLEDGMENT

This research is partly supported by the National Institute of Information and Communications Technology, Japan.

REFERENCES

[1] E. J. Rosensweig and J. Kurose, "Breadcrumbs: efficient, best-effort content location in cache networks," *Proc. IEEE INFOCOM 2009*, pp. 2631–2635, Apr. 2009.

[2] M. Kakida, Y. Tanigawa, and H. Tode, "Active Breadcrumbs: Adaptive Distribution of In-network Guidance Information for Content-Oriented Networks," *IEICE Transaction on Communications*, vol. E96-B, no. 7, pp. 1670–1679, July 2013.