Keyword-Based Breadcrumbs: A Scalable Keyword-Based Search Feature in Breadcrumbs-Based Content-Oriented Network

Kévin Pognart, Yosuke Tanigawa, Hideki Tode
Dept. of Computer Science and Intelligent Systems
Osaka Prefecture University
Osaka, Japan
{pognart@com., tanigawa@, tode@}cs.osakafu-u.ac.jp

***Abstract*—The Internet shows limited performances for users' needs especially on content sharing and video streaming. Content-Oriented Networks (CONs) are efficient approaches for such uses. They abandon the location-based routing of the Internet (IP routing) for a content identifier-based routing. In CONs, users must know the exact content identifier to request it. To give users an easier use of CONs, we propose the Keyword-based Breadcrumbs (KBC), a scalable keyword-based retrieval function for CONs based on Breadcrumbs (BC). Our work focuses on BC because of its simplicity, scalability, particularity and because it can be deployed in today's network, even partially. KBC uses stored information about contents in routers to retrieve several possible answers to a keyword-based request while keeping original behavior of BC for content identifier-based request. We present in this paper the working scheme of KBC, some KBC request managing rules to collect answers, and simulations results to show its performances.**

Keywords-Breadcrumbs; Content-Oriented Network; search; keyword; cache.

I. INTRODUCTION

The current Internet has a host-to-host architecture for allowing an easy communication between two machines. But in the recent years, people use the Internet not for direct communication but principally for sharing contents with a lot of people and viewing video streams. As a result, the current Internet has limited performances. Nowadays, some systems such as peer-to-peer (BitTorrent) can improve content sharing performances by coordinating several users by the contents they have.

Inspired by peer-to-peer systems, Content-Oriented Network (CON) is an alternative to the current Internet with some special features. Content identifier is used instead of location identifier for routing messages. Also, content identifiers are unique. The main characteristic of CON is that contents can be copied and cached in the network while keeping the same content identifier. It allows answering a request by one of the content copies located over the network.

As for CON, several routing methods have been proposed to realize it [1][2]. In our work, we particularly focus on Breadcrumbs (BC) [3][11] due to its attractive features described in Section II. This BC-based CON has simple content caching, location and routing systems. In BC, we assume that users and possibly routers have a content cache. Routers have also a BC table used to route requests. When content passes through a router, a BC entry is created in its BC table to indicate the direction of the cached content. If the content goes through a node having a content cache, the content is cached. Requests are firstly sent to a server to download contents by using IP routing. When a request arrives at a router where a BC entry for the same content identifier exists, the request is redirected to follow the direction shown in the BC entry. Each next node will redirect the request according to the direction in BC entries until finding the content in a content cache. If an issue occurs during the redirection, the BC entries are invalidated and the request is forwarded again to the server by IP routing.

To perform the routing, content identifiers must be unique. This uniqueness makes the requests difficult from a user's point of view. This problem also exists in the current Internet with URLs, and it leads to the need to use web search engines. Current web search engines are not an efficient solution because they use location and they cannot use cached content information. Hence, we propose Keyword-based Breadcrumbs (KBC). We extensively designed the BC framework to complement it with a keyword-based search feature while keeping the way of working of BC and its advantages. Also, we have implemented different KBC request behaviors to retrieve answers and we compare their performances.

In this paper, we present CONs and some unknown content search features. Then, we describe principle, specifics, and settings of KBC, and we compare KBC and the unknown content search features presented. We continue by evaluating KBC performances with some simulation scenario. To conclude, we summarize the main points about KBC and we talk about our future work.

II. RELATED WORK

A. Related CON schemes

To create a CON, several schemes have been proposed. The Data oriented Network Architecture (DONA) [5], the Network of Information (NetInf) [6], the Publish-Subscribe Internet Routing Paradigm (PSIRP) [7], and the Content-Centric Networking (CCN) [4] are the main approaches. In DONA, sources publish contents into the network and their information is spread to the nodes called resolution handlers. A request goes to a resolution handler to be routed to the

 ISBN: 978-1-61208-377-3

content. Then, the content is sent back to the requester by the reverse path or by a shortest route. NetInf can retrieve contents by name resolution and by name-based routing. Depending on the model used, the publication of a content uses a Name Resolution Service (NRS) by registering the link between the name and the locator, or it uses a routing protocol to announce the routing information. A node having a content copy can register it with NRS and by adding a new name/location binding. If an NRS is available, the requester can first resolve a content name into several available locators and find a copy from the best source. Alternatively, the requester can send a request with the content name for finding a content copy by name-based routing. Then, content found is sent back to the requester. In PSIRP, contents are published into the network but publications receive a particular Name Scope. Users can subscribe to contents. Publications and subscriptions are linked by a rendezvous system. The scope identifier requested and the rendezvous identifier form the name of the content. By a matching procedure, the corresponding forwarding identifier is sent to the content source. Then, the content is sent to the requester. In CCN, contents are published at servers and nodes, and routing protocols are used to distribute the content location information. Requests are forwarded toward a publisher location. CCN router maintains a Pending Interest Table (PIT) for outstanding requests. PIT maintains this state for all requests and maps them to the requester network interfaces. Contents are then sent to the requester interfaces. CCN can perform on-path caching: when a content arrives at a router, this router can cache a content copy. It allows subsequent received requests for that content to be answered from that cache. While the namespace of DONA, NetInf and PSIRP are flat and names are not human-readable, the CCN namespace is hierarchical and the names can be human-readable. Flat namespace allows persistent names while the hierarchical one is IP compatible. With flat namespace, the routing is structured and the control overhead is low. With hierarchical namespace, the routing is unstructured based on flooding and the control overhead is high.

B. Breadcrumbs-based CON

We particularly focus on Breadcrumbs [3][11] which has been designed to reduce server loads and to form an autonomous CON in cooperation with cached contents. The network is a cache network where routers can cache contents and manage a table of BC entries which are guidance information to a node holding the corresponding content. Note that in our research, actually, not core nodes but edge nodes including STBs or terminals only have content caches for higher feasibility, though this limitation can be removed easily. When a content passes through a router, this router creates in its BC table a BC entry corresponding to the content as shown in Figure 1. A BC (BC entry) is data containing the content ID, the next node and the previous node on the content path, and the most recent time at which the content was requested and was forwarded via this router. BC is used for in-network guiding of request. Nodes information in BC is used to route requests. Time information is used to manage BCs in BC table and delete

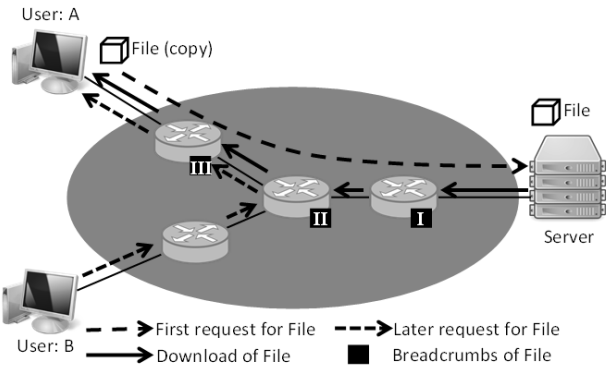

Figure 1. BC system overview

the oldest ones (since the last time update if any). When a request is created at a user node, its destination is set to a server containing the desired content in an ideal case. On its path, if the request encounters a router where a BC corresponding to the desired content exists, the router will redirect the request to the direction of the next node indicated by the BC entry, and the subsequent BC trail, series of BC entries, will guide the request until it finds the content in a cache. If a problem occurs during this redirection (content not cached at BC trail destination, lack of BC entry in the BC trail), the request is redirected to its initial server by IP routing while invalidating the whole corresponding BC entries. Namely, through tracing a series of BC entries, a request can follow the content downloaded previously. Some advantages are that the server loads are reduced and that there is no need to implement coordination protocol for cached contents. Also, it combines IP-routing for the first destination of request and BC trail routing when a right BC is found on path by requests. In terms of feasibility and scalability, BC is very interesting. It combines location-based routing and content name-based routing. Moreover, since location-based routing is the default routing system, BC can work in a partial deployment scenario allowing incremental deployment in the network. It has been demonstrated that this partial deployment is feasible but the performances highly depend on the deployment proportion [8]. Nevertheless, it has been shown that overlay can be used to improve these performances too.

C. Unknown contents search feature in CON

Regarding the keyword-based search feature for CON, some approaches have been proposed. It is important to add this feature in CON because the current web search engines use centralized data centers, and they cannot access caches. Hence, some advantages due to basic concepts of CON are not used. One approach to provide such a feature is to implement a system similar to typical multimedia search engines into CCN [9]. This system searches the contents similar to the content the user includes inside its request. When a search by content name is performed, the search interest is flooded over the network. Each node sharing searchable content performs a feature extraction task: a feature vector containing information about the content characteristics is extracted for each content, and an index is

 ISBN: 978-1-61208-377-3

formed for each content type. When a request for similar contents is received by a node, it performs similarity search by comparing the request descriptors (the feature vector of the content requested) against the index to find a set of the most similar contents. When a node has similar contents, a new content is created: it is a collection of corresponding CCN links constituted by a label name for the similar object descriptors and a target name for the CCN name. An interest is sent to the requester to inform him about its availability. He requests the collection objects from each interest received. Then, data packets carrying the collections of names of similar objects are sent to the requester. This approach seems to be extendable to a keyword-based search feature but it does not seem feasible for a network such as the current Internet because of the flooding of messages all over the network. Another approach uses keywords as a secondary content identifier by converting them to IDs. Independent Search and Merge (ISM) and Integrated Keywords Search (IKS) are two solutions based on the same general settings [10]. Content is identified by its ID, its location and its list of keywords. Each keyword has an ID by using a pre-defined hash function. The whole content IDs and keyword IDs form the general IDs. Another hash function maps general IDs to an IP address. Each node has a content search table which stores the mapping information between content ID and content location. When a user requests a content by its ID, the hash function generates the destination IP address. Once a node received a content request, it adds the new mapping entry in its content search table. Also, each node has a keyword search table which contains keyword IDs (the ones which generate the node address by the hash function) and the list of the corresponding content IDs. With ISM, for each keyword requested, it is converted into keyword ID and then into IP address. The answers are the list of content IDs for each keyword ID. With IKS, the keywords requested are first sorted and then each subset of keywords is converted into keyword ID and then into IP address. In these two methods, keywords are at the same level as content IDs. It improves the search efficiency but either results are too numerous and irrelevant (if each keyword is handled independently) or the number of keyword IDs is too large (if each subset of keyword lists has one keyword ID).

III. KEYWORD-BASED BREADCRUMBS

In order to have a feasible and scalable keyword-based search feature for CON, we introduce Keyword-based Breadcrumbs (KBC). Our goal is to add an intrinsic keyword-based search feature to BC system while preserving the BC advantages in terms of simplicity, scalability, feasibility and working. For this purpose, we add elements to BC system to allow two ways of working: the standard working using content name-based request and sending back of content, and a new one using keyword-based request, where KBC entries are used to find other contents in other location than server, and where answers are information about content and not the content itself. To distinguish BC system and KBC system, BC entry will be renamed to KBC entry from now when it concerns KBC system.

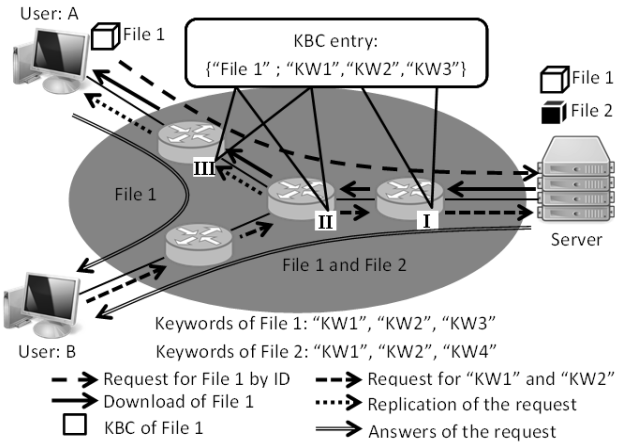

Figure 2. Keyword-based Breadcrumbs system overview

A. Principle of the Keyword-Based Search Feature

The basic idea is to use KBC to find closest corresponding contents. In the initial state, there are no cached contents and no guidance information. When a content is downloaded, KBCs are created on-path like in the BC system. The difference appears for a keywords-based request. For the KBC request, the first destination remains a server. If the request reaches a node with one or more KBC entries whose keywords correspond to the requested ones, the request will be replicated as shown in Figure 2. Replicated requests follow their KBC trail while the original request continues its path to the server. Then, when a right content is found, an answer containing the content ID, its list of keywords and its location is sent back to the requester. By this method, the requester can get a large number of answers with information for choosing the one he wants and if there are several identical contents, he can select the closest one. Also, IP-routing is used for downloading a content found by such a request because the answer gives the content ID and the location, and so performing another BC request for this content ID is unnecessary.

B. Specificities of KBC

In the proposed KBC system, we created new messages type: requests by keywords (KBC request, in opposition to BC request for a request by content ID) and answer (to KBC request because for BC request the answer is the content itself). We set rules for managing the behavior of KBC request. Also, some additions have been done to nodes to allow the use of keywords. As described previously, content has its list of keywords in addition to its ID for the creation of KBCs. Each server contains contents and a server table which contains some of its closest other servers. This information is used to redirect KBC request for having enough answers. KBC entry contains the content ID, the content keywords, the next node and the previous node on the content path, and the most recent time the content was requested by its ID and was seen at this node. Time information is used to manage KBC in KBC table. If a KBC timer reaches the time out limit because of inactivity, it is deleted. Routers have a KBC table and a KBC request table

 ISBN: 978-1-61208-377-3

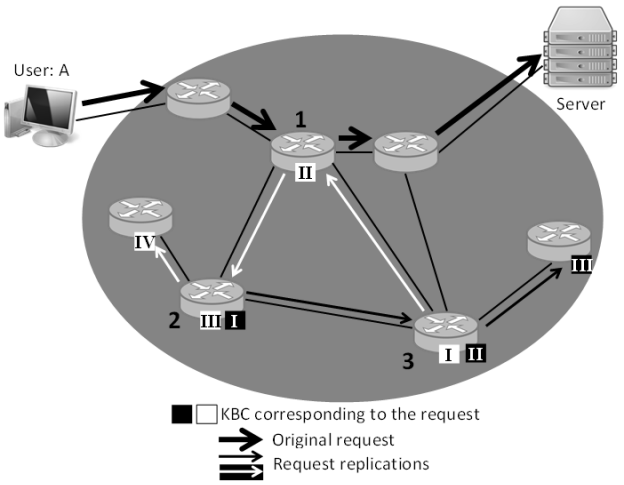

Figure 3. KBC request replications loop

containing the request IDs of recent KBC requests that went through them. This table exists to avoid an issue of KBC request replications loop. This issue happens when a triangle of routers is as follows: one KBC trail follows two edges and another one with the same keywords list follows the last edge in the same way. Figure 3 presents such a situation. The two nodes containing these two KBC entries (nodes 2 and 3) create KBC request replication whenever KBC request for the corresponding keywords list goes in. In node 2, a replication is made for the black KBC trail, and in node 3 a replication is made for the white KBC trail which will go again in node 2 via node 1, and so on. A KBC request contains the list of keywords set by the requester, a request ID for managing answers and for avoiding the previous issue, and the last node ID on its path. This information is used to optimize some request replications. When a request follows even partially a KBC trail on its reverse path, each router will replicate the request to follow this KBC trail. Then, a lot of replications are useless. Only the first one is enough, others are flooding the network. Hence, if the next node shown in KBC is equal to the previous node on the request path, the request is not replicated. An answer contains the content ID and its list of keywords for allowing the user to know if this answer corresponds to the content he wants. Also, it contains the request ID for linking the answer to its request. And it contains the location of the content for allowing the requester to select the closest one he wants between several identical contents. By this addition, the request for a content found by a KBC search will not perform another search in the network (BC request) because this work was already done with the KBC search. Note that an answer must not contain the content itself. The goal is to search corresponding contents, but the user has to select the content(s) he wants from the answers list before the download. Thus, answers to KBC requests are only information about contents and not the contents themselves.

C. KBC request settings

An important challenge for keyword-based search feature in CON is to be efficient while not overloading the network and limiting the messages flooding. We propose here two KBC request settings to manage their behavior. As explained previously, a request needs a server destination at its creation. In "1 Server", the KBC request is sent to one server only, but we set a threshold of minimum number of answers found in server. If this threshold is not reached, the request is redirected to another server which was not reached yet by the request, thanks to the servers table. In "1 Server Extended", we keep the settings from "1 Server" but we propose to add new information in contents and KBCs, the origin server location of the content. If a KBC request finds a KBC entry whose origin server is not one of the destination servers, a request replication is created to go to the new server. We add also in KBC request a list of destination servers which is updated at each replication for a server to avoid useless replications. We add to server a request ID table to avoid several answers for the same KBC request ID (the server sends answers for a KBC request only one time for each KBC request ID).

D. Comparison between KBC and other unknown contents search features

KBC has the fact that keyword is at the same level as content name in common with ISM, IKS, and similarity content search. ISM looks like KBC with one notable exception, the entry used to search contents contains one keyword and a list of contents associated, while in KBC only the KBC table is used and so KBC contains one content name and its keywords list. This difference makes more complicated KBC search in routers but it avoids the large number of irrelevant answers in ISM. IKS being close to ISM, the same comparison can be made. However, IKS solves the irrelevant answers issue by giving to each subset of keywords list a unique ID. It causes a feasibility problem because the number of keyword IDs is too large to be implemented. The similarity content search does not use keywords but descriptors to find similar contents. It is why we focus on comparable elements in the search mechanisms. Requests flood the network to find right contents. Unlike in KBC, only cache nodes have the indexes used for the search. Hence, the search is less structured than in KBC.

IV. EVALUATION

A. Simulation Scenario

To evaluate KBC, we use a modified version of Breadcrumbs+ (BC+) simulator for implementing KBC. Hence, BC+ with adaptive invalidation is used instead of BC [11]. It is an improvement of BC to avoid the issue in which some requests cannot reach the intended content in a particular situation. The differences with BC are that a BC+ entry has a list of the previous nodes on the content path instead of the previous one only, and if at the end of a BC trail, content is replaced or cannot be cached, an invalidation message is sent to all the nodes in this previous nodes list.

- Network Topology: To evaluate the proposed KBC, we use a flat router-network based on the Waxman model on a lattice points of 1000x1000, α=0.1 and β=0.05 [12]. There are 1000 routers, 5000 users and

 ISBN: 978-1-61208-377-3

50 servers. Each router is connected to five users and the server locations are chosen according to uniformly random distribution. Regarding caches, only edge nodes including STBs or terminals have content caches for higher feasibility, though this limitation can be removed easily. Each cache can have a maximum of two contents.

- Keywords: For evaluating KBC, we set three types of keywords (KW1, KW2 and KW3) which are hierarchically linked. All contents and requests contain one of each previous type of keywords (1 KW1, 1 KW2 and 1 KW3). In KBC system, a KBC request is initially routed toward a server. Keyword types are hierarchical for practicability of the initial routing. KW1 represents the main characteristic of the content (video, audio, etc.). Only keywords belonging to a single KW1 are used. KW2 represents a sub-domain of KW1 (if KW1 is "Video", KW2 can be "Action", "News", "Sports"…). There are 25 different keywords for KW2. KW3 is a more specific keyword describing more precisely the content. For each KW2, there are four keywords possible for KW3. In total, 100 keywords combinations are possible.
- Contents: Servers contain in total 10,000 contents which are all unique by their content ID and which are all defined by three random keywords (one of each keyword type). Hence, each keyword combination corresponds to around 100 contents. Also, servers have the same contents during all the simulation time and for each simulation.
- Servers: Each server has a set of its three nearest server neighbors to redirect the requests in the situation where the threshold number of answers from servers is not reached. Servers have also a list of request IDs of requests went to them. We did not set a size for this set. However, it can be easily done by setting a time out to entries.
- KBC table: It does not have limitation about its size but information about its size is collected during simulations.
- Requests: The two types of user request (by content ID and by keywords) are generated at an independent, identical and exponentially-distributed random interval. In a first time, 50,000 BC requests by content ID are made for initializing the network and for spreading KBCs. Then, we study KBC system for 55,000 BC requests and a variable number of KBC requests depending on the wanted ratio between these two request types.
- Answers: When answers for KBC requests are received, one of them is selected to download the content by IP routing.

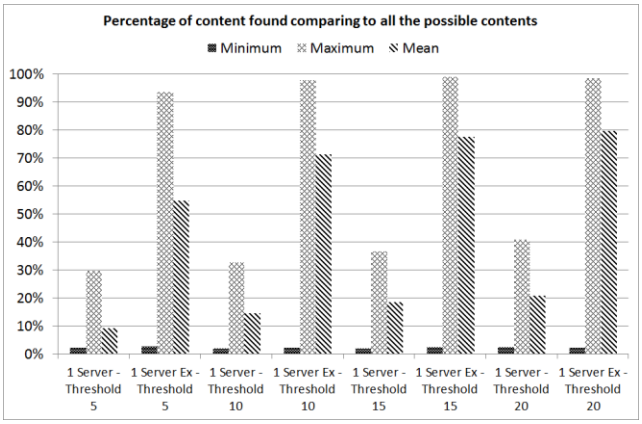

Figure 4. Content retrieval efficiency for KBC requests

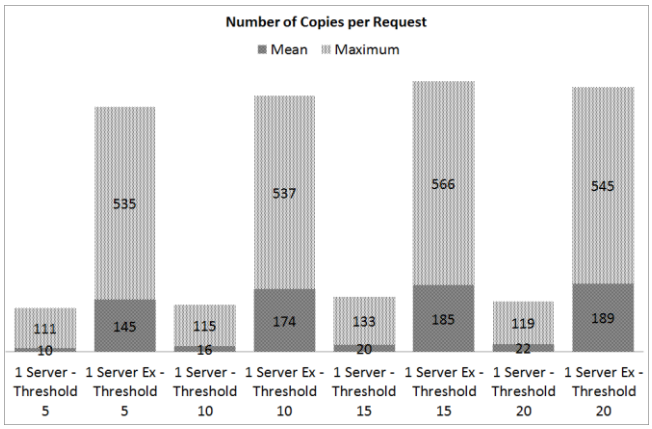

Figure 5. KBC request replications

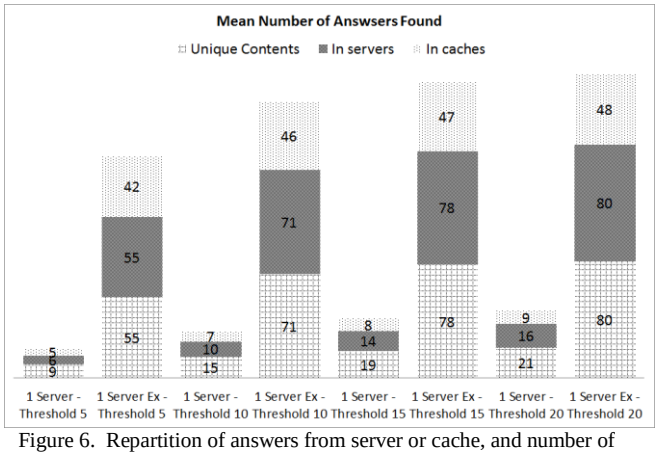

Figure 6. Repartition of answers from server or cache, and number of unique contents found (without taking into account identical contents)

We have four requests patterns to switch between BC requests and KBC requests. For 1 KBC request, 2 BC requests are performed (2 BC), 4 BC requests are performed (4 BC), 10 BC requests are performed (10 BC) or 15 BC requests are performed (15 BC). Also, we set four thresholds for the minimum number of answers found in servers: 5, 10, 15 and 20. For each threshold value, we take the mean of the results of each different requests pattern to focus on the threshold values.

B. *Performances*

Figure 4 presents the efficiency of KBC system for retrieving right contents. The setting 1 Server has limited performances because requests are restrained to a close area

TABLE I. KBC TABLE SIZE

Mean size of KBC table	Unbiased variance of the BC table size	Standard deviation of the BC table size
71	712	28

of the first destination server. 1 Server Extended shows high efficiency for finding different but right contents.

Figure 5 shows the number of KBC request replications. With 1 Server, requests are replicated only few times in mean, which means that it does not degrade the network performances. With 1 Server Extended, replications are numerous and can interfere with a good network working.

The repartition of answers between caches and servers shown in Figure 6 is also interesting because it indicates how many KBC trails are successfully followed. Once again in 1 Server, the results are low. On the other hand, 1 Server Extended can find a lot of corresponding KBC trails even if the threshold in server is low. In a small network area, KBC requests can easily find a KBC about a content from outside of this area. Hence with 1 Server Extended, KBC requests can go all over the network. It is confirmed by the equality between the number of different contents found (Unique Contents) and the number of contents found in servers.

Regarding the KBC tables, no limit was set because the needed size is important to know. Table I presents the mean size of KBC table with its unbiased variance and its standard deviation. Viewing these results, we can propose to have a KBC table of 100 entries, which is 1/100 of all contents in our simulated network.

V. CONCLUSION

We presented in this paper a keyword-based search feature using Breadcrumbs and some keyword-based request settings to control their behavior. The purpose is to make CON easier by a feature similar to web search engines from users' point of view. Our system is different from other approaches because it does not flood the network at each request. It is scalable not only in CON but also in partially deployed Breadcrumbs because keyword-based search is close in its working to content name-based one, and thanks to Breadcrumbs characteristics. We showed that the setting 1 Server Extended has a good potential even if there is a trade-off between the network flooding and the search efficiency.

In our future work, we want to use other keyword settings, and change our network for having non unique contents. Also, we will take into account the content popularity, and we want to implement an indicator of users' satisfaction (if a content is downloaded thanks to a keyword-based search, it means that for the keyword list used, the user is satisfied of this content).

ACKNOWLEDGMENT

This research was supported in part by National Institute of Information and Communication Technology (NICT), Japan.

REFERENCES

[1] J. Choi, J. Han, E. Cho, K. Kwon, and Y. Choi, "A Survey on Content-Oriented Networkinf for Efficient Content Delivery," IEEE Communications Magazine, vol.49, no.3, Mar. 2011, pp.121-127.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," IEEE Communications Magazine, vol.50, no.7, Jul. 2012, pp.26-36.

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

[4] V. Jacobson, et al., "Networking named content," in Proc. ACM CoNEXT 2009, Dec. 2009, pp.1-12.

[5] T. Koponen, et al., "A Data-Oriented (and Beyond) Network Architecture," in Proc. SIGCOMM '07, Aug. 2007, pp. 181-192.

[6] B. Ahlgren, et al., "Second NetInf Architecture Description," Deliverable D-6.2 in The Network of the Future - FP7-ICT-2007-1-216041, Apr. 2010.

[7] M. Ain, et al., "Architecture Definition, Component Descriptions, and Requirements," Deliverable D2.3 in Publish-Subscribe Internet Routing Paradigm - FP7-INFSO-IST-216173, Feb. 2009.

[8] T. Tsutsui, H. Urabayashi, M. Yamamoto, E . Rosenweig, and J. Kurose, "Performance Evaluation of Partial Deployment of Breadcrumbs in Content Oriented Networks," in Proc. IEEE ICC FutureNet 2012, Jun. 2012, pp.5828-5832.

[9] P. Daras, T. Semertzidis, L. Makris, and M. Strintzis. "Similarity Content Search in Content Centric Networks," in Proc. ACM MM '10, Oct. 2010, pp.775-778.

[10] Y. Mao, B. Sheng, and M. Chuah, "Scalable Keyword-Based Data Retrievals in Future Content-Centric Networks," in Proc. 2012 Eighth International Conference on Mobile Ad-hoc and Sensor Networks (MSN) , Dec. 2012, pp.116-123.

[11] M. Kakida, Y. Tanigawa, and H. Tode, "Breadcrumbs+: Some Extensions of Breadcrumbs for In-network Guidance for Inter-AS Content-Oriented Network Topology," in Proc. WTC 2012, Mar. 2012, pp. 1283-1292.

[12] B. M. Waxman, "Routing of Multipoint Connections," IEEE J. Sel. Areas Comms (Special Issue on Broadband Packet Communication), vol.6, no.9, Dec. 1998, pp.1617-1622.

 ISBN: 978-1-61208-377-3