A Comparative Study of Keyword-Based Search Features in Content-Oriented Networks

Kévin Pognart, Yosuke Tanigawa, and 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 quantitatively compare two keyword-based search features for CON: the existing Independent Search and Merge (ISM) and Keyword-based Breadcrumbs (KBC) we propose. While ISM uses routers to store mapping information between a content and its locations, and between a keyword and its corresponding contents, the proposed KBC simply uses routers to store information about contents went through them, in CONs based on Breadcrumbs (BC). We present in this paper the working schemes of ISM and KBC, and we compare their advantages and inconvenience, and their performances using simulation results.**

Keywords-Keywords-based Breadcrumbs; Independent Search and Merge; Content-Oriented Network; search; keyword; cache.

I. INTRODUCTION

The current Internet was made for an efficient communication between two machines by its host-to-host architecture. Nowadays, main use of the Internet is to watch video streams and to share contents, but the host-to-host architecture has limited performances. That is the reason why we present Keyword-based Breadcrumbs (KBC) [1], another network architecture with a user-friendly content searching feature exploiting its specificities. The inspiration comes from peer-to-peer (BitTorrent), which improves content sharing performances by coordinating several users by the contents they have. Content-Oriented Network (CON), which is a network architecture based on peer-to-peer features, is an alternative to the current Internet. In CON, messages are routed using content identifier instead of location identifier. In-network caches can store copies of contents while keeping the same content identifier. A content and its copies are considered identical when requesting it.

As for CON, several routing methods have been proposed to realize it [2][3]. In our work, we particularly focus on Breadcrumbs (BC) [4][5] due to its attractive features described in Section II. BC is a feasible hybrid solution that simply provides content-oriented capability over the current IP network. 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 KBC [1]. 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. We introduced different KBC request behaviors to retrieve answers. Also, we have implemented another keyword-based retrieval function called Independent Search and Merge (ISM) [6] for comparing their architecture and their performances.

In this paper, we present CONs. Then, we propose principle, specifics, and settings of KBC. After that, we describe also principle, specifics, and settings of ISM. We evaluate KBC and ISM performances by comparing them with some simulation scenario. After, we summarize the advantages and inconveniences of KBC and ISM, we conclude about our choice to work on KBC, and we talk about our future work.

II. RELATED WORK

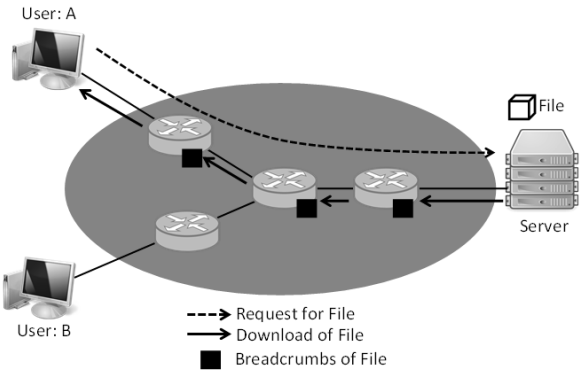
Our work is an enhancement of CONs from a user's point of view. Hence, we compared several CONs and chose the one with interesting characteristics.

A. Related CON schemes

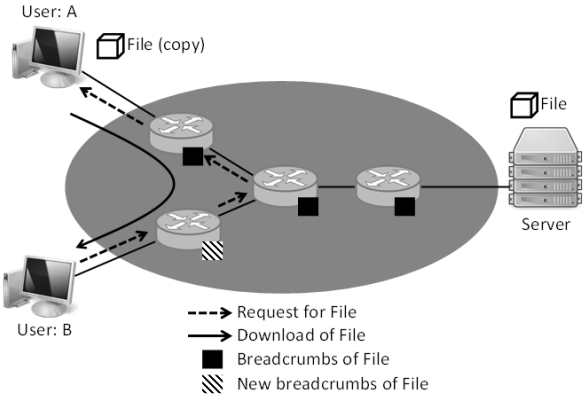
To create a CON, several schemes have been proposed. The Data oriented Network Architecture (DONA) [7], the Network of Information (NetInf) [8], the Publish-Subscribe Internet Routing Paradigm (PSIRP) [9], and the Content-Centric Networking (CCN) [10] are the main approaches. In DONA, sources publish 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 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. It uses the scope identifier requested and the rendezvous identifier to form the name of the content. And 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 [4][5], 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



(a) Download of a content from a server



(b) Download of a content using Breadcrumbs information

Figure 1. Breadcrumbs system overview

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). 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 the outdated 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 as shown in Figure 1 (b). 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. Hence, BC trails can be followed in both directions: one is used for finding content, and the other one is used to invalidate the BC trail. 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

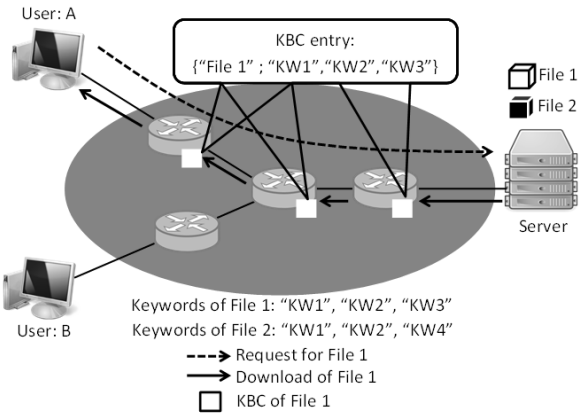
implement coordination protocol for cached contents. Also, it combines IP-routing for the first destination of request and BC trail routing when a right BC is found on path by requests. In terms of feasibility and scalability, BC is very interesting. It combines location-based routing and content name-based 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 [11]. Nevertheless, it has been shown that overlay can be used to improve these performances too.

C. Unknown contents search feature in CON

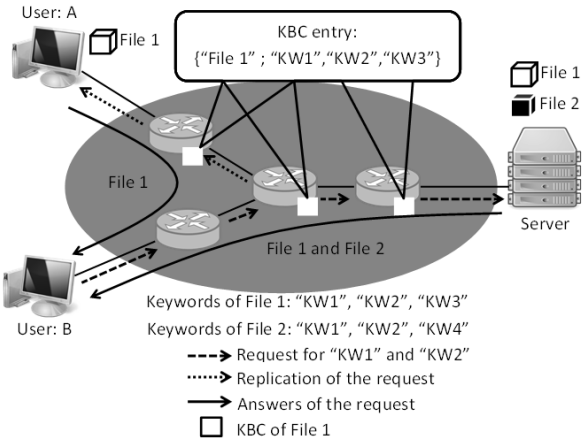
Regarding the keyword-based search feature for CON, some approaches have been proposed. It is important to add this feature in CON because the current web search engines use centralized data centers, and they cannot access caches. Hence, some advantages due to basic concepts of CON are not used. One approach to provide such a feature is to implement a system similar to typical multimedia search engines into CCN [12]. 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 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 consists in giving each keyword (or group of keywords) a numerical keyword ID as in Independent Search and Merge and Integrated Keywords Search [6]. Using these keyword IDs, the system can retrieve the answers for a keyword-based request by finding contents related to the keyword. This information is stored in router tables. We will develop Independent Search and Merge mechanisms in Section IV.

The approach we work on is Keyword-based Breadcrumbs [1]. The main idea is to use the specificities of Breadcrumbs [4], content information stored in routers on-path, to store also keywords related to content in routers on the content path. When a keyword-based request arrives at a router, all content information is checked and if some



(a) Download of a content from a server



(b) System behavior to a keyword-based request

Figure 2. Keyword-based Breadcrumbs system overview

contents have all the requested keywords, the request is copied to find these contents using BC specifics. These mechanisms will be explained in the next section.

III. KEYWORD-BASED BREADCRUMBS – OUR PROPOSAL

In order to have a feasible and scalable keyword-based search feature for CON, we propose 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.

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 as shown in Figure 2 (a). 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 (b). 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. 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. 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. 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. We also made some optimization for reducing the number of request replications:

- Routers have a KBC table and a KBC request table 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.

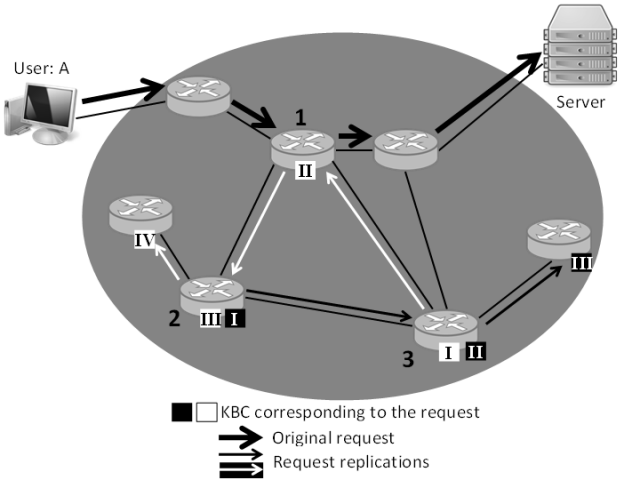


Figure 3. KBC request replications loop

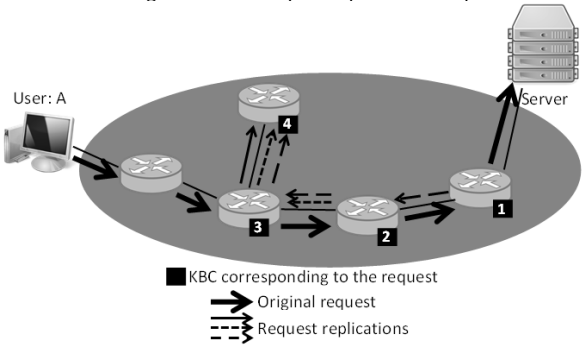


Figure 4. KBC request flooding by following a KBC trail

- In KBC request, the information of the last node ID on path is used to reduce useless 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 as shown in Figure 4. Then, a lot of replications are created for only a single KBC trail. Only the first replication to follow the KBC trail is enough, others flood 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.
- In KBC answer, the addition of the content location assures to choose the closest content between the answers. And also, the following request for a content found by this KBC search will not perform another search in the network (BC request) because this work was already done with the KBC search.

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. For "1 server extended", we also made some optimizations:

- 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 past history 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.
- For a KBC request, original destination server and other destination servers found on-path are distinguished. The threshold of minimum number of answers found in servers is tried to be reached only with the original destination server.

IV. Independent Search and Merge – Compared method

To evaluate the efficiency of KBC, we implemented another similar existing approach: Independent Search and Merge (ISM) [6]. In ISM, content ID and keywords describing the content are used to publish content. During a keyword-based search, some routers retrieve content IDs corresponding to the requested keywords. Then, at the user device, only the intersection of all content IDs retrieved are shown as the answers of the search. Then, user can perform a search using the content ID he selected.

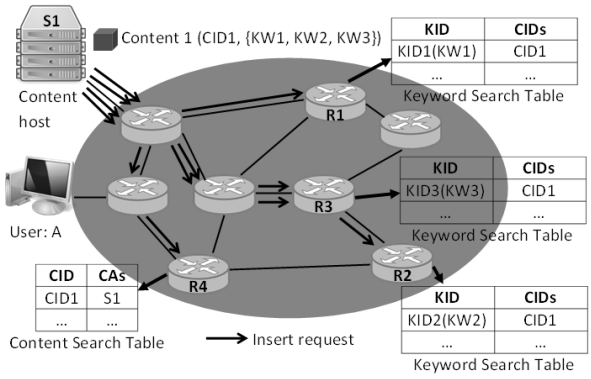
A. Principle of Independent Search and Merge Feature

The main idea is to link content ID to content locations, and to link keyword to corresponding content IDs. Hence, knowing a content ID allows the network to find easily all locations of the content. Also, by searching for some keywords, the network can retrieve easily content IDs corresponding to each keyword, and by taking into account only the intersection of all lists of answers, the user has its final list of results. One characteristic point in ISM is that each keyword is managed independently. To store all this information, routers are used. They have 2 tables: A Content Search Table, which stores for each content ID the corresponding content locations, and a Keyword Search Table, which stores for each keyword the corresponding content IDs. Also, using some algorithms, these 2 tables are not redundant between all routers. Each router is assigned to specific keywords and content IDs.

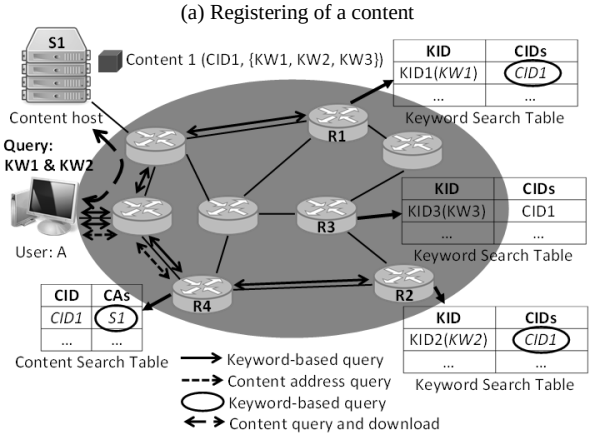
Registering a content: To register a content, we need to map the content ID and the content location, and we need to map each keyword and the content ID. For this purpose, we convert each keyword to a keyword ID, similar to content ID but belonging to a different domain, by using a pre-defined hash function. Also, another hash function is used to map any ID (content ID or keyword ID) to an IP address. The resulting IP address must correspond to an existing router IP address. If it is not the case, the hash function is applied again until generating a valid IP address. The generation of a valid IP address is assured by the use of DMap [13]. Regarding the content location, the previous hash function is applied to the content ID to generate the router IP address corresponding to this content ID. Then, an insert request containing the content ID and the content location is sent to this router. Once the router receives this insert request, the new mapping entry is added in the router Content Search Table. Regarding the keywords, each keyword is converted into a keyword ID, and for each keyword ID a router IP address is generated. For each keyword ID, an insert request containing the keyword ID and the content ID is sent to the corresponding router. Once the router receives this insert request, the mapping information is added to the router Keyword Search Table. Because content ID and keyword ID domains are different, routers can distinguish if the insert request concerns the Content Search Table or the Keyword Search Table. This scheme is explained in Figure 5 (a), where Content 1 is uploaded in the server S1. Once the upload is done, the content needs to be registered. Hence, S1 creates an insert request to link Content 1 ID CID1 and its location S1, and 3 insert requests to link the keywords KW1, KW2 and KW3 to CID1. For the first insert request, the hash function is used on CID1, and the result is the IP address of the router R4. Hence, this insert request is sent to R4. For the keywords, they are first converted to keyword IDs KID1, KID2, and KID3 by another hash function, then the previous hash function is used. KID1 leads to R1, KID2 leads to R2, and KID3 leads to R3.

1) Requesting a content by its content ID: To request a content using its content ID, we just need to apply the hash function to the content ID to obtain the router IP address where the information about the content locations is stored. A request containing the content ID and the requester IP address is sent to the corresponding router. Then the router sends back the content locations found in its Content Search Table. Finally, the requester needs to send a similar request to one of the content locations answered to download the content.

2) Keyword-based request: The user makes a request by using several keywords. At the user node, each keyword is



(a) Registering of a content



(b) System behavior to a keyword-based request followed by a content download request

Figure 5. Independent Search and Merge system overview

converted to a keyword ID using the same pre-defined hash function as the one used for registering contents. Then, for each keyword ID, the other hash function is applied to have the IP address of routers having information about these keywords. Hence, for each keyword ID, a request containing the keyword ID and the requester IP address is sent to the corresponding router IP address. At the router, the list of content IDs corresponding to the keyword ID is retrieved from the router Keyword Search Table and sent to the requester. Once the requester received all answers, only the content IDs shared by all lists are kept and shown to the requester. Finally, the requester can select one of the content IDs to perform a request using the content ID. This behavior is explained in Figure 5 (b). User A makes a keyword-based request with the keyword KW1 and KW2. They are converted into the keyword IDs KID1 and KID2. By using the hash function, KID1 leads to the router R1, and KID2 leads to the router R2. The content ID information is sent back to user A: CID1 by KID1, and CID1 by KID2. Only the content IDs in common are kept: CID1. Then user A requests the content CID1. By using the hash function, it leads to the router R4. CID1 location (S1) is sent back to user A. Finally, user A sends a request for CID1 at server S1, and S1 replies the content CID1.

V. EVALUATION

To evaluate KBC, we used different simulation scenarios for KBC and ISM. We use the simulation results to compare them.

A. Simulation Scenario

1) General settings:

- Network Topology: To evaluate the proposed schemes, we use a flat router-network based on the Waxman model on a lattice points of 1000x1000, α=0.1 and β=0.05 [14]. There are 1000 routers, 5000 users and 50 servers. Each router is linked 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: 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", etc.). 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.
- 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 requests by content ID are made for initializing the network and for spreading contents information. Then, we study the systems for 55,000 content ID-based requests and a variable number of keyword-based requests depending on the wanted ratio between these two request types.
- Answers: When answers for keyword-based requests are received, one of them is selected to download the content.

We have four requests patterns to switch between content ID-based (CID) requests and keyword-based (KW) requests. For 1 KW request, 2 CID requests are performed (2 CID), 4 CID requests are performed (4 CID), 10 CID requests are

performed (10 CID) or 15 CID requests are performed (15 CID). Also, we set four thresholds for the minimum number of answers found in servers: 5, 10, 15 and 20. For each threshold value, we average the results for each different requests pattern to focus on the threshold values.

2) Keyword-based Breadcrumbs: To evaluate KBC, we use a modified version of Breadcrumbs+ (BC+) simulator [5] for implementing KBC. Hence, BC+ with adaptive invalidation is used instead of BC. 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.

- Servers: Each server has address information 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, which 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.

3) Independent Search and Merge: In ISM, all information is in router tables. All content IDs are equally distributed between all routers, and all keyword IDs are equally distributed between all routers. The function used to link IDs to router addresses is pre-defined. Also, at the beginning of the simulation, the information of contents in servers are known by the corresponding tables (content ID-content location, and keyword ID-content ID).

B. Performances

1) Keyword-based Breadcrumbs: Figure 6 presents the efficiency of KBC system for retrieving right contents. The setting of 1 Server has limited performances because requests are restrained to a close area of the first destination server. 1 Server Extended shows high efficiency for finding different but right contents.

Figure 7 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 8 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

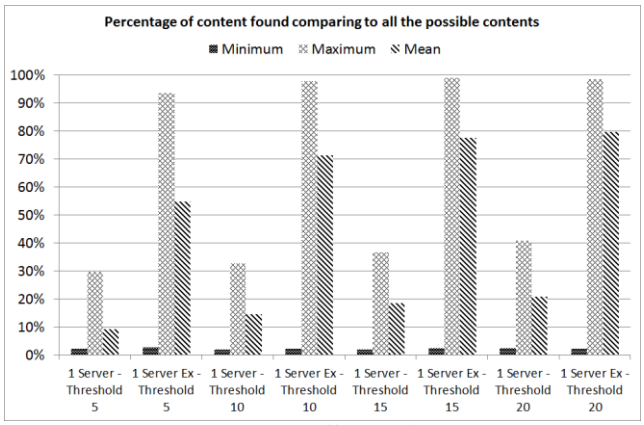


Figure 6. Content retrieval efficiency for KBC requests

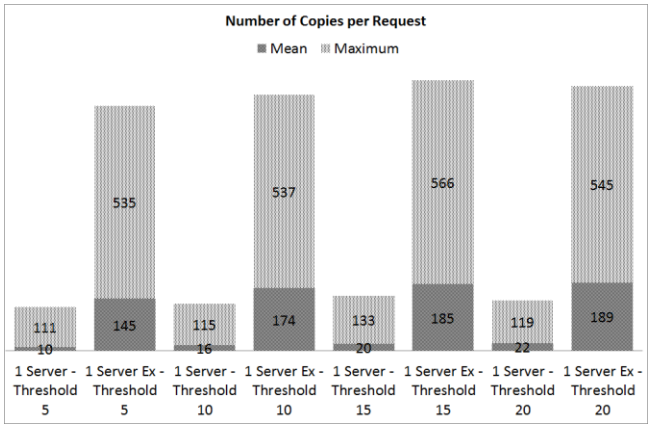


Figure 7. KBC request replications

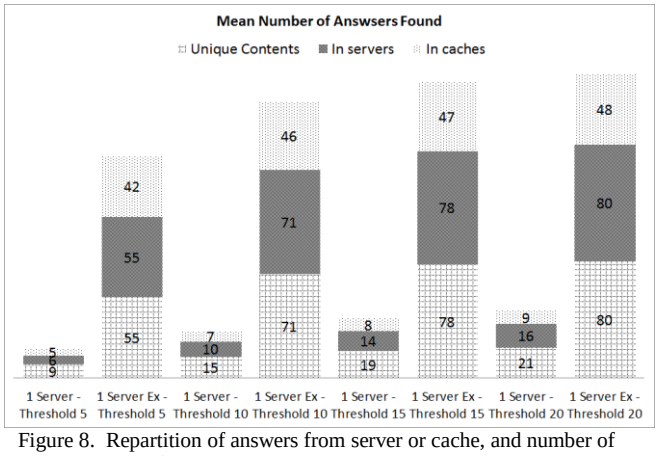


Figure 8. Repartition of answers from server or cache, and number of unique contents found (without taking into account identical contents)

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.

Figure 9 shows how many KBC requests do not gather enough answers in servers. It indicates which thresholds are fitting or not, knowing the number of contents and their keywords. As expected, the lower the threshold is, the lower the failing rate is. But if the threshold is too low, user does not receive enough answers. About this trade-off, choosing a medium threshold (10 or 15) seems a good compromise.

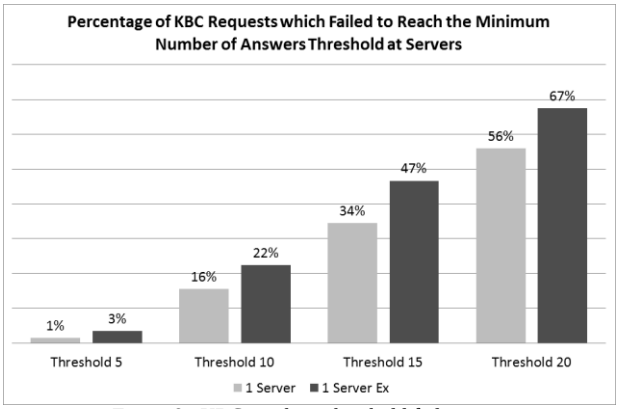


Figure 9. KBC reaching threshold failing rate

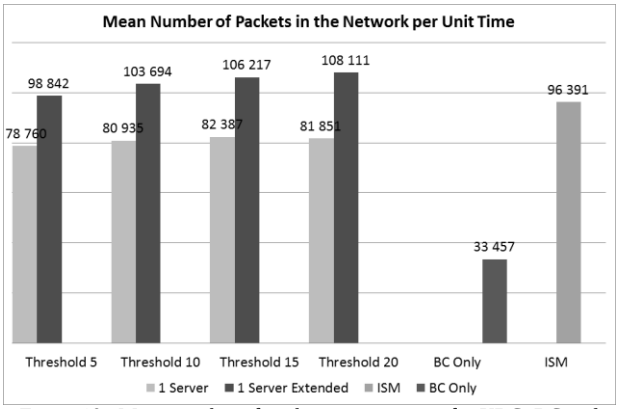


Figure 10. Mean number of packets per unit time for KBC, BC and ISM

TABLE I. KBC TABLE SIZE

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

TABLE II. NUMBER OF ENTRIES IN ISM TABLES

Mean number of entries in Content Search Table	Mean number of entries in Keyword Search Table
10	0.125

TABLE III. NUMBER OF CONTENT LOCATIONS FOUND IN ISM

Minimum number of locations found	Maximum number of locations found	Mean number of locations found	Standard deviation of number of locations found	Percentage of cases where only the server location exists
1	1694	7	49	15%

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.

2) Independent Search and Merge: Router tables being the center of ISM mechanism, we take a look at them. Content and keyword information are distributed between all routers tables. Table II shows primary information about ISM tables. The Keyword Search Table (KST) is conditioned only by the number of keywords and the number of content using these keywords. Here, we have 125 keywords, which correspond to 125 KST entries distributed in the 1000 routers KSTs. Hence, if there are too many keywords, this table becomes too huge and unfeasable. This is a limitation of ISM. About the Content Search Table (CST), its size depends on the number of contents and the number of their locations. Here, we have 10000 contents, which correspond to 10000 CST entries distributed in the 1000 routers CSTs. Table III shows the mean number of locations for a single content and its standard deviation. At least, each content can be found at a server. For all contents, there is in mean 7 user cache locations to find it, and in only 15% of keyword-based requests, only the server location was registered.

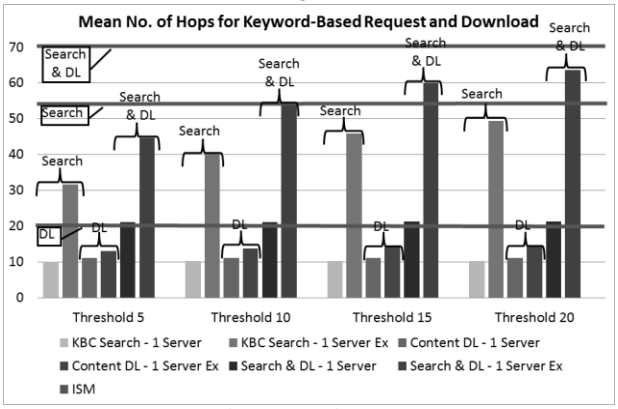


Figure 11. Mean no. of hops to perform a keyword-based request

Another aspect is the efficiency for retrieving results. ISM presents a 100% of efficiency thanks to its mechanism. However, it is counterbalanced by the necessity for routers tables to have information of all keywords and all contents.

3) Comparative results: To compare KBC and ISM, we focused on several aspects.

The mean number of packets in the network per unit time indicates which one implies the more messages creation and management in mean. As shown in Figure 10, KBC with the setting "1 Server" induces the least flooding where the setting "1 Server Extended" induces the higher flooding. ISM is between them, however, depending on the threshold, KBC "1 Server Extended" induces between 2% and 12% more packets than ISM, which is still acceptable.

Figure 11 shows the search time and the download time for a keyword-based request by using hop count. The lines correspond to ISM results. For KBC "1 Server Extended", the number of necessary hops to perform a keyword-based search depends on the minimum number of answers from

server threshold. The higher the threshold is, the higher the number of hops is. Even with a high threshold, performances of ISM are worse than KBC. It can be explained by the routers locations for gathering the information. In KBC, the effective network area is small. It is close to the user and to the original destination server of the request. In ISM, the whole network is used to store the information. Also, regarding the download time in ISM, a request for the content location must be made while in KBC this piece of information is included in the keyword-based request answers.

ISM emphasis is on efficiency. Hence, all contents and all locations can be found by a keyword-based request. However, it requires router tables capacity to be high enough to store information for all existing keywords and for all contents, which can be unfeasible. Also, if a router has an issue and becomes out of order, all information stored are inaccessible and/or lost because in ISM, network-wide coordination of routers is necessary. On the other hand, KBC focuses on scalability and capability of adaptation. We want to have enough answers, but only a small part of the whole network is used. Also, the KBC information is managed automatically with time-out in tables and with invalidation messages in case of no more valid KBC trail.

VI. CONCLUSION

We presented in this paper a comparative study of keyword-based search features for Content-Oriented Network. Such features are important from a user point of view to make the network accessible. We based our study on the proposed Keyword-based Breadcrumbs, our keyword-based search feature based on Breadcrumbs, and on the existing Independent Search and Merge, another similar feature performing differently. They are scalable and interesting. KBC 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. These features focus on different points. ISM is focused on the content retrieval efficiency. However, be careful about the size of tables used to store content information, because it depends on the number of routers, contents, and keywords. Also, a failure in the system results in a loss of data and so in efficiency. KBC is focused on the capacity of adaptation and on the user's neighborhood. Content information is stored on its path, and a keyword-based request will go to the closest server and find close content information. There is a trade-off between the network flooding and the search efficiency. It seems that KBC with the setting "1 Server Extended" is a good compromise with a high efficiency and a capacity of adaptation.

In our future work, we want to work more on KBC by changing 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). Also, we continue to see other possible ways to perform keyword-based search in CON.

ACKNOWLEDGMENT

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

REFERENCES

[1] K. Pognart, Y. Tanigawa, and H. Tode, "Keyword-Based Breadcrumbs: A Scalable Keyword-Based Search Feature in Breadcrumbs-Based Content-Oriented Network," in Proc. AFIN 2014, Nov. 2014, pp. 20-25.

[2] 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.

[3] 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.

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

[5] M. Kakida, Y. Tanigawa, and H. Tode, "Breadcrumbs+: Some Extensions of Breadcrumbs for In-network Guidance in Content Delivery Networks," in Proc. SAINT 2011, Jul. 2011, pp. 376–381.

[6] 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.

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] T. Vu, et al., "DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet," in Proc. IEEE ICDCS 2012, Jun. 2012, pp. 698-707.

[14] 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.