

# Performance Evaluation of Partial Deployment of Breadcrumbs in Content Oriented Networks

Tatsuhiro Tsutsui    Hiroyuki Urabayashi    Miki Yamamoto  
Faculty of Engineering Science  
Kansai University  
Email: { k960991, k475558, yama-m } @kansai-u.ac.jp

Elisha Rosensweig    James F. Kurose  
Dept. of Computer Science  
University of Massachusetts, Amherst, USA  
Email: { elisha, kurose } @cs.umass.edu

**Abstract**—In recent years, much work has been devoted to developing protocols and architectures for supporting the growing trend of data-oriented services. One drawback of many of these proposals is the need to upgrade or replace all the routers in order for the new systems to work. Among the few systems that allow for gradual deployment is the recently-proposed *Breadcrumbs* technique for distributed coordination among caches in a cache network. *Breadcrumbs* uses information collected locally at each cache during past downloads to support in-network guiding of current requests to desired content. Specifically, during content download a series of short-term pointers, called *breadcrumbs*, is set up along the download path. Future requests for this content are initially routed towards the server which holds (a copy of) this content. However, if this route leads the request to a *Breadcrumbs*-supporting router, this router re-directs the request in the direction of the latest downloaded, using the aforementioned pointers. Thus, content requests are initially forwarded by a *location ID* (e.g., IP address), but encountering a breadcrumb entry can cause a shift over to *content*-based routing. This property enables the *Breadcrumbs* system to be deployed gradually, since it only enhances the existing location-based routing mechanism (i.e. IP-based routing).

In this paper we evaluate the performance of a network where *Breadcrumbs* is only partially deployed. Our simulation results show *Breadcrumbs* performs poorly when sparsely deployed. However, if an overlay of *Breadcrumbs*-supporting routers is set-up, system performance is greatly improved. We believe that the reduced load on servers achieved with even a limited deployment of *Breadcrumbs*-supporting routers, combined with the flexibility of being able to deploy the system gradually, should motivate further investigation and eventual deployment of *Breadcrumbs*.

## I. INTRODUCTION

In the current Internet, content distribution is taking up a growing portion of Internet traffic. Content distribution style has changed over the past few years, from conventional web-server access to CDNs [1] and P2P, both of which focus on content ID instead of the location from which content is downloaded. For example, in CDNs a user identifies his/her required content by location-ID (URL or IP address), but DNS name resolution might reroute the content request to a different, replicated server. Thus, from a service viewpoint, content distribution has already become content-oriented, with requests being more content-centered. As a result, there has been a growing interest in the Networking community regarding the design of Content-Oriented Networks (CONs) [2][3][4][5], which would restructure the Internet architecture in order to support a set of content-oriented protocols.

In CONs, the centrality of content expresses itself in many ways. Content itself is identified by a content ID, and content can be downloaded from wherever it is found, which might not be an “official” content server/provider. This is made possible in many CON architectures by the extensive use of caching for performance improvement. While the different architectures differ in terms of content *transmission* once it is located - CCN (Content Centric Network) [2] uses a content-oriented method, while others [3][4][5] transmit content based on location ID - in all currently-proposed architectures content *discovery* is done in a content oriented manner: the end-user specifies only a content ID to locate. As a result, this shift in content-search protocol would require full deployment of any of the CON architectures cited above in order for the system to work at all, which means complete replacement of a large number of network elements.

In contrast to the architectures mentioned above, we consider the recently-proposed content distribution platform *Breadcrumbs* [6], which combines location ID and content ID in the search process. During content download, each router along the download path stores a *breadcrumb* entry for this content, logging the direction in which the content was sent. Consequently, the content search process alternates between two methods. Initially, a request is specified in location-ID terms, as in current-day IP networks. However, if and when the request arrives at a router that has a (recent) breadcrumb entry for the content of interest, it uses the information in this entry to route the request towards where the content was last sent. The search for content continues via breadcrumb pointers until content is found, or a node is reached with no breadcrumb for this content, at which point routing reverts back to being location-based. Note that, since location-based routing is still the default routing in the system, the *Breadcrumbs* architecture can work in a *partial deployment* scenario, which allows for incremental deployment in the network.

While partial deployment is optional with *Breadcrumbs*, certain mechanisms of this system will be activated more frequently under these conditions. One such property is the frequency of *trail invalidation*. If a router  $v$  receives a request for content  $f$  from the direction to which its downstream pointer indicates, the trail is invalidated from  $v$  and all the way upstream. This is done in order to avoid following trails of breadcrumbs that do not lead to content [6]. In a fully-

deployed system, this situation happens when there is no cached content downstream along the trail. However, in the partial deployment scenario, this situation can occur also when a request arrives at a non-breadcrumbs router. Upon arrival at that router, routing will switch to IP-based routing, and this in turn can cause forwarding of the request in the reverse direction, which will invalidate the upstream breadcrumbs.

In this paper, we evaluate the performance of Breadcrumbs when partially deployed using a simulation-based approach. The main contributions of this paper are:

- Using simulations, we demonstrate that Breadcrumbs can be deployed partially, and that this deployment reduces the load at servers from the outset.
- We show that performance - measured both in terms of server load and the number of trail invalidation - is poor until Breadcrumbs has been almost fully deployed. In our experiments, significant performance improvement was detected from 80% deployment and above.
- We propose and experiment with interconnecting Breadcrumbs routers via overlay network. In such a scenario, many of the challenges with limited deployment are mitigated. We find that for such a system, performance improvements become linear with deployment ratio in the network.
- For the overlay system, we also consider different degrees of transparency between the overlay and the underlying network. We find that with more information available to a request passing through an overlay node, performance is further improved.

## II. BREADCRUMBS: SYSTEM OVERVIEW

Breadcrumbs [6] considers a *Cache Network*  $G = (V, E)$ , in which each router  $v \in V$  is associated with its own cache, such that the cache stores content passing through the router. Throughout this paper we shall assume the content being stored in these caches are files, each with a unique ID. Also, we shall interchangeably refer to these cache-equipped routers as nodes, routers or caches, depending on the context.

Requests for a file  $f$  are routed initially towards a publicly-known source of  $f$ , and at each router en-route to this destination the cache is inspected to see if it holds a copy of  $f$ . If  $f$  is found in the cache it is downloaded directly from the cache where it was found. With Breadcrumbs, during the download of  $f$  each router en-route to the download destination stores a breadcrumb entry for  $f$ . A breadcrumb entry consists of the most recent direction and time that  $f$  was forwarded at this router in the past, as explained in the next section.

Figure 1 illustrates the basic behavior of Breadcrumbs for a fully-deployed system. A content request generated by host A is initially routed towards a server that has a copy of  $f$  (Fig. 1 (a)). The request-forwarding is based on the location-ID of the content server. When  $f$  is downloaded, Breadcrumbs routers along the download path store copies of  $f$  as well as breadcrumb entries with upstream and downstream pointers, as described below. To highlight the operation of breadcrumbs, consider the case where  $f$  was evicted from all caches, except

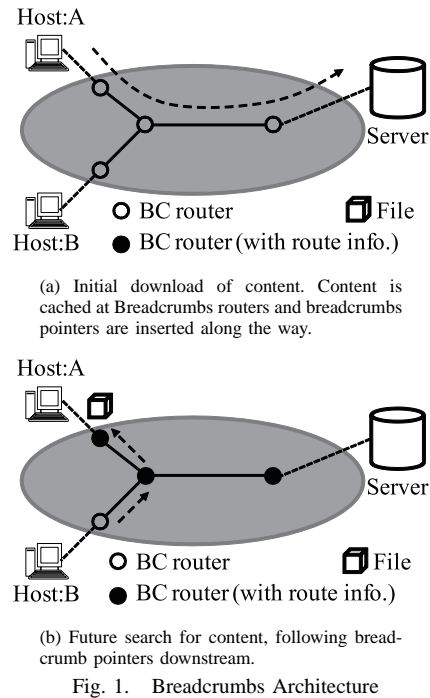


Fig. 1. Breadcrumbs Architecture

for the one-hop neighbor of host A (Fig. 1 (b)). At this state, we follow the path a request for  $f$  arriving from host B follows. If the request arrives at a router with a breadcrumb entry for  $f$ , as shown in the figure, that router forwards this request via the link specified by this breadcrumb entry. In such a hop-by-hop manner the request is forwarded in search for content. Note that this mechanism is content-centered and is not location-based, with each hop taken based on local information only. Thus, in addition to lowering the load on servers, this content-based forwarding enables a best-effort search for content in the network, where cache coordination is only implicit - no coordination protocol for cache content is relied upon [6].

We now discuss the system architecture in more detail. Each breadcrumb entry is a 5-tuple entry, consisting of the content ID, the location ID of the node from which content arrived (the “upstream” node) and the node to which it was sent (the “downstream node”), and the most recent time the content was requested and was seen.

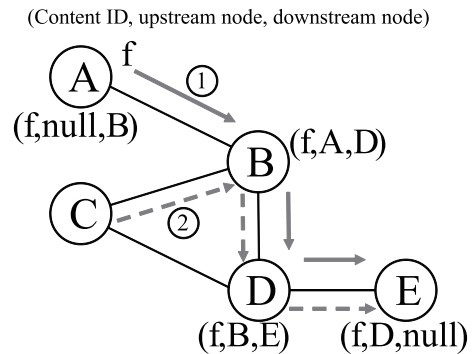


Fig. 2. Breadcrumbs mechanism

Figure 2 shows an example of Breadcrumbs in action. To avoid visual clutter, we omit the time information and just show a 3-tuple entry consisting of content ID, upstream and downstream node IDs. In this example, content  $f$  is stored originally at a server that is connected to router A. When  $f$  is downloaded to an end-host connected to router E, breadcrumbs for this content are left at all en-route routers - A, B, D and E (①) in this case. When an end-host connected to router C requests  $f$ , this request is initially forwarded towards router A by location ID-based forwarding. When this request encounters breadcrumbs at router B, this request is forwarded to router D as informed by the breadcrumbs' downstream node. This process is repeated again, so the request is forwarded to router E in this hop-by-hop manner, directed each time by breadcrumb stored at the next hop (②).

Once a file is located, it is downloaded to the user who requested it. In [6], two download policies were suggested for Breadcrumbs. With DFQ (Download Follows Query) the download route is the reverse request path, while with DFSP (Download Follows Shortest Path) the download route is the shortest path from where the file was located to the requesting user. In this paper, we will only consider systems employing DFSP, though we believe similar results can be shown with DFQ as well.

Caches have limited space and so might evict previously-cached content. As a result, a request forwarded along a breadcrumb trail might end up at a *dead end* - a node with no pointers or content copies. If this occurs, the request is forwarded to the original server by location-based forwarding. Additionally, since the breadcrumb trail did not lead to a copy of the content, we would like this trail to be invalidated and no longer used. Breadcrumbs supplies two invalidation mechanisms: timeout and trail invalidation. Timeout involves discarding old breadcrumbs. Trail invalidation, on the other hand, takes place when a request for content arrives at a cache from the direction to which its breadcrumb points (e.g., in Fig. (2), a request for  $f$  arriving at node B from node D). If this happens, the entire trail of breadcrumbs can be invalidated since it leads to no content copy. We refer the reader to [6] for a detailed discussion of this property.

### III. PARTIAL DEPLOYMENT OF BREADCRUMBS

#### A. Partial Deployment Properties

The deployment of the Breadcrumbs system requires upgrading existing router equipment, to support the storage of breadcrumb entries and their usage. However, the Breadcrumbs system can be deployed *incrementally* in the network, one router at a time, without limiting end-user access to content. This is due to the fact that location-based routing is still used and fully maintained with Breadcrumbs, allowing for a seamless transition between routing policies when moving between router types. Specifically, in a partial deployment scenario, a request currently using content-based routing that encounters a non-Breadcrumbs router (e.g., a conventional IP router), can and will be forwarded next only according to

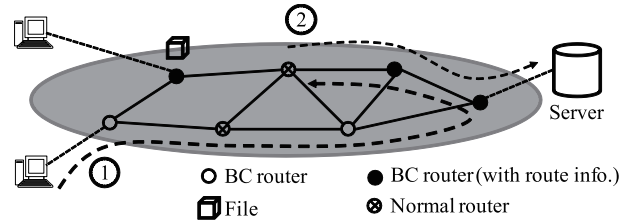


Fig. 3. Partial Deployment

location ID (such as the server IP address). In this section, we consider system performance under such partial deployment.

In [6], as mentioned in the previous section, the authors describe how trail invalidation occurs when a request is forwarded in the reverse direction of a breadcrumb trail. With partial deployment, such occurrences are likely to happen often, because many routers do not support Breadcrumbs, resulting in reverting back to location-based routing, which can lead to routing along the reverse breadcrumb trail. Indeed, this case happens quite often since the breadcrumbs trails tend to form along the shortest path to the original server (Fig. 3). Additionally, when trail invalidation does not occur in such a fashion, user delay is probably increased, for several reasons, among them the futile search for this user and the delay of future users who will use the same trail for content search. Also, if the shortest-path to the server, from the dead end, is the reverse path, this indicates the search path itself was not the shortest path from the requesting host.

#### B. Evaluation for Partial Deployment

In this section, we present the results of evaluation, via simulation, of the performance of Breadcrumbs in a partial deployment scenario. As performance measures we use (a) server load (i.e., number of downloads satisfied at servers) and (b) the number of trail invalidations that occur over the simulation. For network topology, we used the BA (Barabasi-Albert) model [7] with  $|V| = 1000$  nodes (routers), generated by the BRITE topology generator [8][9]. There are a total of 5000 users and 50 content servers, each connected to router  $v \in V$  with probability  $1/|V|$ . We assume that all users generate the same request rate, and so user distribution reflects the distribution of request arrivals. Each server holds 10000 equal-sized unique files, such that each file is stored in exactly one server, and each cache can store five files ( $|v| = 5$ ). At each server, content popularity follows Zipf law [10]. Finally, let the *Partial Deployment Ratio* (PDR) of Breadcrumbs be defined as  $b/|V|$ , where  $b$  is the number of routers in the network equipped with Breadcrumb capabilities. Note that non-Breadcrumbs nodes also lack caching capabilities.

In addition to partial deployment, we also assume here that users can elect to use or not use the content-based search offered by Breadcrumbs. Thus, Breadcrumbs information and request routing are used only when both router has Breadcrumbs capability and the user has elected to use Breadcrumbs. In this paper, we assume that the proportion of users that elect to use Breadcrumbs is identical to the PDR.

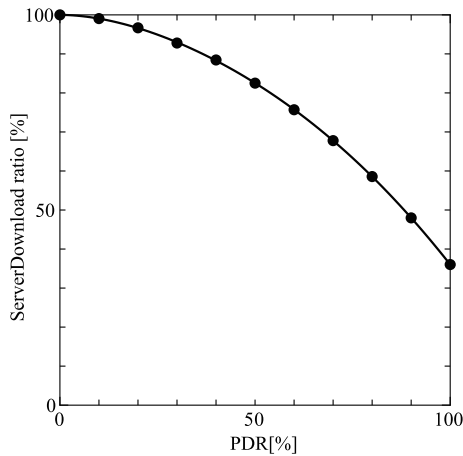


Fig. 4. Server Download

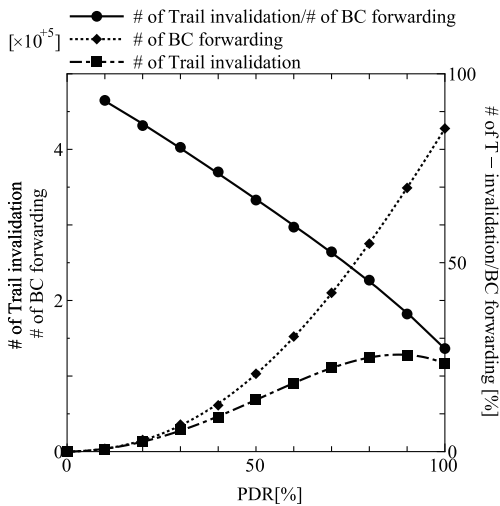


Fig. 5. Trail invalidation

For each set of results shown here and in the next section, we ran 5 simulations of each setting, and 95% confidence intervals were so small that they are omitted from the shown graphs. For each PDR, the assignment of servers to nodes was done independently from past simulations, i.e., the deployment for low PDR cases are not necessarily subsets of high PDR deployments.

The results of our simulation-based experiments are shown in Figures 4 and 5. Figure 4 shows server load as a function of deployment, with the horizontal and vertical axis showing the PDR and server download ratio (i.e., percentile of how many server downloads took place, w.r.t. no-breadcrumbs scenario), respectively. At PDR = 0, all content requests are directed towards and are satisfied by the original server, since no caching takes place. As this figure indicates, the server load decreases with the increase of PDR, as content is located in caches more often. In small and medium deployment regions, requests encounter a Breadcrumbs router with high probability, but in mid-search standard routers are encountered with high probability as well, and the search reverts back to location-based routing. Since location-based routing routes

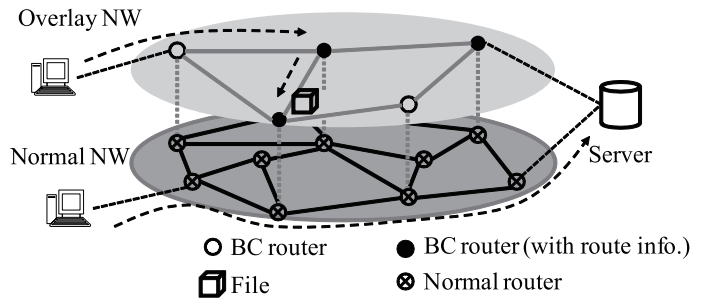


Fig. 6. Breadcrumbs Overlay Architecture

to the server, in this region the server download ratio is not noticeably improved.

Figure 5 presents the trail invalidation characteristics. Here we consider the ratio between the number of requests that moved over to content-based search and the number of such requests that ended up with trail-invalidation, termed the *trail invalidation ratio*, abbreviated as TIR. This figure shows that when PDR is small, TIR is large. This is to be expected, since with limited deployment most trails will not reach searched-for content before encountering a non-Breadcrumbs node. We see that the number of trail invalidations monotonically increases up to about 80% PDR, for the same reason. Thus, we can see that performance improvement by Breadcrumbs is only brought on by wide deployment of the system in the network.

#### IV. OVERLAY APPROACH FOR PARTIAL DEPLOYMENT

##### A. Overlay Approach

Inspired by the limitations of Breadcrumbs under limited deployment we saw in the previous section, we consider in this section the benefits of interconnecting the Breadcrumbs routers so as to form an overlay network. As we shall see, this approach is one promising way to help obtain good performance when Breadcrumbs is only partially deployed.

Figure 6 depicts a system where Breadcrumbs routers form an overlay network. As before, end-users can elect whether or not to access the overlay when a request passes via node that is part of the overlay. On the overlay network, a content request is managed as in a fully-deployed Breadcrumbs network. As a result, trail invalidation no longer occurs as a result of reaching a non-Breadcrumb router, but can still occur (as with fully deployed Breadcrumbs systems) if and when a request arrives at a dead-end.

By default, the overlay network is decoupled from the underlying network. Still, in addition to having full access to the overlay, we consider here different levels of *overlay transparency* w.r.t. the underlying network. We consider here two such levels:

- **Open Cached Content (OCC)** - when a content request encounters a router which is part of the overlay, content at this node can be inspected and downloaded to the requesting user.
- **Open Breadcrumbs (OB)** - In addition to access to the cache, we also allow routing on the overlay using

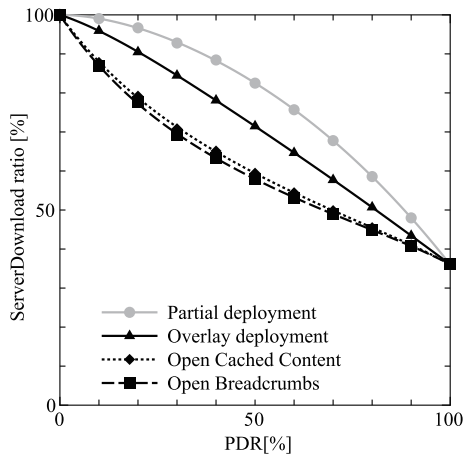


Fig. 7. Server Download with Overlay. “Partial Deployment” indicates performance with no overlay, “Overlay Deployment” is the case where the overlay is completely decoupled, “Open Cached Content” and “Open Breadcrumbs” are as defined in the text.

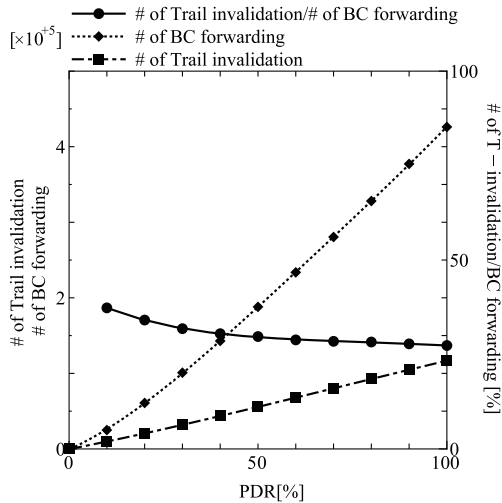


Fig. 8. Trail invalidation with Overlay

Breadcrumb information at the node. OB differs from full Breadcrumbs usage in that, when the content is located, it is transported in the underlying network (by using IP address of the requester of content, i.e., the source IP address), and the download does not set, refresh or affect in any way the state of breadcrumb pointers in the overlay network.

### B. Evaluation for Partial Deployment with Overlay

In this subsection, we evaluate overlay approach, using OCC and OB. We use the same simulation parameters and notation as in the previous section. Furthermore, once the Breadcrumbs nodes are selected, the overlay network they form also follows the Barabashi-Albert model.

Figure 7 shows the performance in terms of server load. When the overlay network is isolated from the regular IP network (“Overlay deployment” in Figure 7), server download ratio is improved linearly with incremental deployment. Thus, when comparing to the system with no overlay (“Partial

Deployment”), adding an overlay can improve Breadcrumbs performance in partial deployment scenario. Allowing for increased interaction between the overlay and the underlying network can further improve performance.

Figure 8 shows the trail invalidation characteristics. As shown in this figure, trail invalidation rates are much lower when an overlay is in place, which can help explain the performance improvement of server load for both small and medium deployment regions. When compared with the performance we observed in Fig. 5 for the non-overlay architecture, the number of content-based searches increases while the trail invalidation ratio is significantly improved. This means using a simple overlay network of Breadcrumbs routers can seriously enhance system performance, even with limited deployment.

## V. CONCLUSIONS

In this paper, we evaluated the performance of Breadcrumbs when only partially deployed. In principle Breadcrumbs can operate even when partially deployed, but its positive impact on performance is negligible when deployed in a limited manner. As a way to deal with this issue, we considered linking the Breadcrumb routers via a network overlay, and demonstrated that this greatly improves performance, avoiding many of the problems that arise without the overlay. Further exposure of the Breadcrumbs capability of routers can improve performance even more. Considering the need for new architectures that can be easily deployed, in a backwards-compatible manner, and which can help deal with the increasing focus on content transfer, we find that Breadcrumbs presents an attractive candidate for such an architecture.

## ACKNOWLEDGMENT

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

## REFERENCES

- [1] <http://www.akamai.com>.
- [2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs and R. L. Braynard, “Networking Named Content,” in *Proc. ACM CoNEXT 2009*.
- [3] A. Ananda, A. Akella, S. Sesha and S. Shenker, “Packet Cache on Router: The Implications of Universal Redundant Traffic Elimination,” in *Proc. SIGCOMM 2008*.
- [4] S. Arianfar, P. Nikander and J. Ott, “On Content-Centric Router Design and Implications,” in *Proc. ACM ReArch 2010*.
- [5] X. Tang, S. T. Chanson, “Coordinated En-Route Web Caching,” in *Proc. IEEE Transactions on Computers 2002*.
- [6] E. J. Rosenswief and J. Kurose, “Breadcrumbs: efficient, best-effort content location in cache networks,” in *Proc. IEEE INFOCOM 2009*.
- [7] R. Albert, A. L. Barabasi, “Statistical mechanics of complex networks,” *Reviews of Modern Physics* Vol. 74, 2002.
- [8] BRITE, <http://www.cs.bu.edu/brite/>
- [9] A. Medina, I. Matta, and J. Byers, “On the Origin of Power Laws in Internet Topologies,” in *Proc. ACM SIGCOMM 2000*.
- [10] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, “Web Caching and Zipf-like Distributions: Evidence and Implications,” in *Proc. IEEE INFOCOM 1999*.