In-network Hop-aware Query Induction Scheme for Implicit Coordinated Content Caching

Kensuke Hashimoto, Yumi Takaki, Chikara Ohta, and Hisashi Tamaki
School of System Informatics, Kobe University
Kobe, Hyogo 657–8501
Email: {hashimoto, yumi, ohta, tamaki}@al.cs.kobe-u.ac.jp

***Abstract*—The breadcrumb scheme was proposed to realize content distribution in next-generation networks. In the breadcrumb (BC) scheme, each node maintains a small piece, called a "breadcrumb," of query induction information. The BC scheme can, however, increase the amount of traffic as well as file acquisition delay since a query for a file is may diverted away from the nearest server with the file even if the query passes close to the server. This paper proposes a hop-aware breadcrumb scheme, in which the query induction information includes latest file acquisition node ID, which enables a query to be diverted toward a closer node, the latest file acquisition node, or the server. Simulations show that the hop-aware breadcrumb scheme yields shorter file acquisition time than the breadcrumb scheme.**

***Keywords*-breadcrumbs; hop-aware; query induction; in-network cache.**

I. INTRODUCTION

The Internet is now being used to transfer large contents (or files) such as video and music more often. This has stimulated research on content-centric networking such as Contents Delivery Network (CDN)[1] and Peer-to-Peer (P2P)[2]. Most studies consider overlay systems on the current Internet architecture. Users are, however, interested in contents themselves rather than where they are. Therefore, more recently, content-centric architectures have been targeted such as the "clean slate" approach for the next-generation Internet[4], [7], [8], [9]. In particular, in-network processing for content naming, search, routing and storage is one of major issues for content-centric architectures.

[4] proposed a simple content caching, location, and routing system that adapts an implicit, transparent, and best-effort approach towards in-network caching. The system is based on Transparent En-Route Caches (TERC) [6], [7]. Each node maintains a small piece, called a "breadcrumb," of state for queries and the direction in which a file was transferred. For convenience, we call the system proposed in [4] the "Breadcrumb (BC)" scheme. A breadcrumb helps a query to locate content, so that cache hit-ratio increases, i.e. the access load for content server is reduced.

In the BC scheme, however, even if a query for a server passes close to the server of the file, the query may be diverted away from the server. This may increase the amount of traffic as well as the content acquisition delay. Furthermore, the BC scheme is weak against dynamic changes to the topology. This paper tackles these problems by proposing the Hop-aware BC (HBC) scheme, which utilizes hop-count information.

The rest of the paper is organized as follows: Section II briefly explains the BC scheme. Section III introduces the HBC scheme as a significant improvement over the BC scheme in terms of file acquisition delay and the amount of traffic. Section IV conducts simulations to confirm the effectiveness of the HBC scheme. Finally, Section V concludes this paper.

II. BREADCRUMB SCHEME

The Simple Best-Effort Content Search (S-BEACONS) scheme, described below, is a traditional implementation of BC [4].

In the BC scheme like the IP scheme, a query initiated for a file by a request node is transferred to the (original) server of the file. Each file (or content) is assumed to be indexed by a global file ID. In the BC scheme, routers are assumed to have the function to cache not only files but also their corresponding BCs. The cache policy is assumed to be Least Recently Used (LRU), which discards the least recently used item first.

A BC contains the following information:

- Global file ID.
- ID of node from which the file arrived (ID of upstream node).
- ID of node to which the file was forwarded (ID of downstream node).
- Previous file transfer time: most recent time the file passed through the node.
- Previous query transfer time: most recent time the file was requested at the node.

A BC is generated in a router when a file is transferred through the router for the first time, and it is updated every time the file or the corresponding query traverses the node.

A. Query Induction

In the BC scheme, if a query for a file traverses a router with a BC for the file, the query is diverted toward the downstream node of the file and it traces the corresponding BC trail. Suppose that a query for a file arrived at time t at a router and found that the file was not cached at the router. Then, with timeout thresholds T_{f} and T_{q}, the router forwards the query downstream if-and-only-if

- The file was cached or refreshed (via successful query) at the router within $[t - T_{\mathrm{f}}, t]$; or
- The previous query passed through the router within $[t - T_{\mathrm{q}}, t]$ and sent downstream.

 ISBN: 978-1-61208-148-9

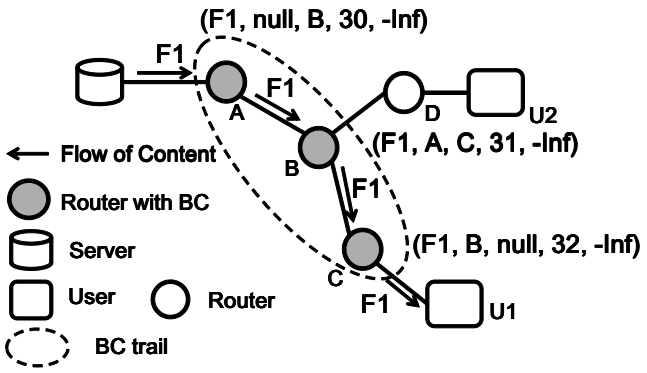

Fig. 1. BC trail and query transfer.

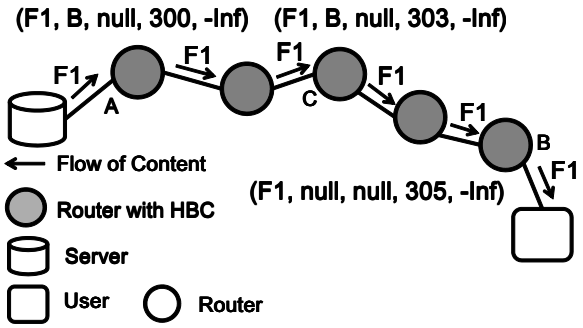

Fig. 2. Example of HBC trail.

1) Invalidation of BC: If the query can not find the file over the BC trail and reaches a dead end, the BC trail is regarded as being stale, so the invalidation procedure is invoked for the trail. More precisely, when the query encounters a node with its downstream entry null (i.e. dead end) and the file is not cached there, the query turns back along the trail and deletes the corresponding BCs along the trail.

2) Update of BC: If a file being transferred finds the corresponding BC in a router, then the BC entries, i.e. the upstream node ID, downstream node ID, and the most recent time the file passed through the node, are updated. If a query for a file finds the corresponding BC in a route, the BC entry of the previous query transfer time is updated.

B. Example of Query Induction

Fig. 1 shows an example of the query induction in the BC scheme. Given this figure, suppose that file F1 is transferred via routers A, B and C and reaches user U1, so that BCs are newly generated on the path. Note that the entry of the previous query transfer time is set to $-\mathrm{Inf}$, and the entry of the upstream node ID in the router A, to which the server is attached, and that of the downstream node in the router C, to which the user U1 is attached, are set to "null." The BC of each router is updated every time the file or the query traverses the router. Next, suppose that user U2 requests the same file and sends its query to the server. It is then transferred via routers D and B and finds an available BC for the file at router B. In this case, the query is diverted downstream toward router C instead of upstream router A. If the query finds the file on the way, the file is transferred from there instead of the server to user U2. This reduces the access load of the server. On the other hand, if the query reaches router C where the downstream BC entry is "null," BCs are invalidated in the upstream direction from router C to the server since the file is not cached on the path.

C. Problems with BC Scheme

In the BC scheme, even if a query for a file encounters a BC for the file near the server of the file, it is once diverted in the downstream direction away from the server. This may cause not only the requesting node to suffer long acquisition delay but also an increase in the amount of traffic in the network. Furthermore, since a BC trail is created as a sequence of upstream nodes (or downstream nodes), it is weak against dynamic topology changes.

III. HOP-AWARE BREADCRUMB SCHEME

To tackle the above problems, we propose the HBC scheme in which a piece, called HBC, of query induction information contains the ID of the node that acquired the file most recently instead of the downstream node ID. This modification enables a query for a file to be transferred to a closer node, the source of the file or the node that acquired the file most recently. We assume that the hop information can be obtained from the routing table.

An HBC contains the following information:

- Global file ID.
- File acquisition node ID: ID of node that acquired the file most recently.
- ID of the node with HBC for the same global file ID from which the query arrived (ID of upstream node)
- Previous file transfer time: most recent time the file passed through the node.
- Previous query transfer time: most recent time the file was requested at the node.

Note that we assume that each query includes a server ID.

It is highly possible that the node that acquired the file most recently still has the file in its cache. Since a HBC for a file directly points to the node that acquired the file most recently rather than the downstream node, the HBC scheme is robust against dynamic changes in network topology.

Like the BC scheme, when a file is transferred through a router, an HBC is newly generated or updated like the BC scheme. LRU is assumed as the cache policy of HBC scheme, too. Fig. 2 shows an example of query induction in the HBC scheme. In this figure, file F1 has already been transferred from the server to the user. In this case, routers A through B generate HBCs whose entry indicating the acquisition node is the ID of the requesting user. Router B is the access router of the user and its HBC entry of file acquisition node is "null" to denote the access router itself.

 ISBN: 978-1-61208-148-9

Unlike the BC scheme, for the invalidation process explained later, a query has a field to convey an upstream node ID, which is initially set to "null." Note that the upstream node does not have to be a physically neighboring node in the HBC scheme. The query also has a field to convey an invalidated node ID, which is also set to "null," initially.

1) Query Induction by HBC: Suppose that a query for a file is traveling to the source node of the file and the query finds an available HBC for the file on the way for the first time. (The definition of availability will be explained later.) The router diverts the query toward the closer (hop-count basis) node, server, or previous acquisition node whose ID is stored in the HBC.

When the acquisition node is selected, the router enters its own node ID into the upstream node ID field of the query before sending it.

Also the router sets the entry of the upstream node ID in the HBC to "null." The router then sends the query to the acquisition node. Every time the query finds an HBC that points to the same acquisition node for the same file at another router on the way, the upstream node ID field in the query is copied into the entry of the upstream node ID in the HBC of the router, and the router writes its own node ID into the upstream node ID field of the query before sending it to the next hop. This process makes the HBCs point to the same acquisition node for the same file to be linked in a serial manner in the reverse (or upstream) direction.

If the file is found on the path toward or at the access router of the acquisition node, the file is transferred from there to the new request user. Otherwise, the invalidation procedure explained in Section III-2 is invoked. In this case, the file will eventually be transferred from the server to the requesting node.

The availability of the HBC is determined as follows. Suppose that a query for a file arrives at time t at a router. If the router has the HBC for the file, the HBC is available if-and-only-if

- The file was cached or refreshed (via successful query) at the router within $[t - T_{\mathrm{f}}, t]$; or
- The previous query passed through the router within $[t - T_{\mathrm{q}}, t]$.

Otherwise the HBC is deleted.

2) Invalidation of HBC: If the query could not find the file on the path toward or at the access router for the acquisition node, the HBCs pointing the acquisition node over the path should be invalidated (Note that the access router has the HBC whose the acquisition node field is set to null.) This can occur due to the cache policy, LRU in this paper. The access router returns the query toward the upstream node whose ID is stored in the HBC after the invalidated node ID field of the query is set to the acquisition node ID. The query is successively sent back following the upstream node IDs of the HBCs over the reverse path until it arrives at the router whose HBC entry of the upstream node ID is null. (Note that the router is the first one where the query found the HBC for the file.) After that, the query is diverted to the source, and eventually the file is transferred to the request node. On the way, if the query finds an HBC whose entry of the acquisition node for the file is the same as the invalidated node, the HBC is also invalidated.

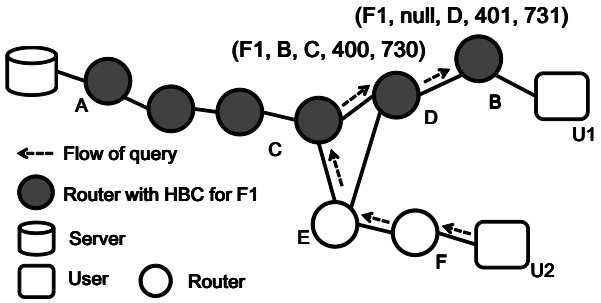

Fig. 3. Example of HBC query induction.

3) Update of HBC: When a file traverses a router, an HBC for the file is newly generated or updated in the router. At this time, the entries of the upstream node ID and the previous query transfer time are set to null and the current time, respectively. The entry of the file acquisition node is set to the ID of the closer (hop-count basis) node, current requesting node, or previous file acquisition node. As a result, the next query will be sent to the closer node. Recall that we assume that the hop-count is obtained from the routing table. Similar to the BC scheme, when a query for a file traverses a router with an existing HBC for the file, the entry of the previous query transfer time is updated to the current time.

A. Example of Query Induction

Fig. 3 shows an example of query induction in the HBC scheme. In this figure, suppose that file F1 is transferred via routers A through C and reaches user U1, so that HBCs are newly generated. Next, suppose that user U2 requests the same file and sends a query to the server. It is transferred via routers F and E and finds an available HBC for the file at router C. Router C refers to its routing table, and then diverts the query to router B which is closer (hop-count basis) than the server from router C. For the invalidation process, the query conveys the ID of router C which will be written in the entry of the upstream node of the HBC of router D. On the path from router C to router B, if the query finds the corresponding file, it will be transferred from there to user U2, so that HBCs are newly generated or updated on the file transfer path. Otherwise, the query deletes the HBCs from router B by following the upstream entries of the HBCs, and then the query arrives at router C which has an HBC whose entry of the upstream node is null. Therefore, the query is transferred to the server, and the file is downloaded from the server to the request node.

IV. Performance Evaluation

In order to compare the HBC scheme with the BC scheme and the IP scheme for the case of several thousand routers, we developed an event-driven simulator in C++ instead of utilizing ns-2 or ns-3. Router topologies used in the simulations were generated based on the BA model by BRITE [3].

 ISBN: 978-1-61208-148-9

TABLE I
BASIC PARAMETER.

Item	Value
Link capacity	1 packet/unit time
Number of files	10,000
Number of routers	6,000
Number of servers	300
Number of users	600
Query size	1 packet
File size	100 packets
Router cache size	100 files

We suppose that servers and users are located in the core and the edges of networks, respectively. Therefore, in the scenarios of the simulations, the servers are attached to the routers one by one in decreasing order of link degree. The users are attached to the routers one by one in increasing order of link degree. Furthermore, since routes in the core network cannot be expected to have enough high-speed memory even in the future, we assume that each router has only a cache to store BC or HBC, and only edge routers to which users are attached have enough cache capacity to store files. The query interval of each user follows an exponential distribution. A requested file is selected according to a Zipf-like distribution with $\alpha = 0.75$[5]. We also assume that query and file transfers are without packet loss.

Table I shows the parameters used in the simulations.

As shown in Table I, each link is assumed to have the capacity that one packet can be sent in one unit time, that is, the transmission time $T_{\rm s}$ is one. Queueing delay $T_{\rm w}$ of an output link of a router is estimated from the amount of traffic by using the M/M/1 queueing model[10]. Let $D_{\rm q}$ denote the average file discovery delay which is the time from the epoch that a query is issued to the epoch that the query finds the file. Using the average number $h_{\rm q}$ of hops taken by a query, which is obtained from the simulation results, the average file discovery delay $D_{\rm q}$ is roughly estimated as

$$D_{\rm q} = L_{\rm q} + h_{\rm q}(T_{\rm w} + T_{\rm s}), \tag{1}$$

where $L_{\rm q}$ denotes query length in packets, and $L_{\rm q} = 1$ in this case. Next, let $D_{\rm f}$ denote the average file transfer delay which is the time from the epoch that file is found to the epoch that the file is received by the requesting node. Using the average number $h_{\rm f}$ of hops taken by a file, which is also obtained from the simulation results, the average file transfer delay $D_{\rm f}$ is roughly estimated as

$$D_{\rm f} = L_{\rm f} + h_{\rm f}(T_{\rm w} + T_{\rm s}), \tag{2}$$

where $L_{\rm f}$ denotes file length in packets, and $L_{\rm f} = 100$ in the simulations. Finally, total file acquisition delay D is given as

$$D = D_{\rm q} + D_{\rm f}. \tag{3}$$

A. Simulation Results

1) Influence of Timeout Threshold: Figs. 4 and 5 show the relative delay ratio in the IP scheme and the download ratio from cache (i.e. cache hit ratio), respectively. Here, the average query interval per user is set to $T_{\rm i} = 70$. The timeout threshold $T_{\rm f}$ is varied from 100 to $1,000$ while the timeout threshold $T_{\rm q}$ is fixed to $T_{\rm q} = 10$.

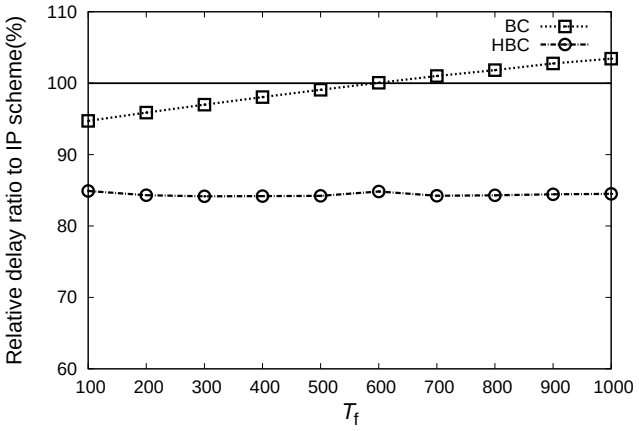

Fig. 4. Characteristics of relative delay ratio in IP scheme as a function of $T_{\rm f}$ ($T_{\rm q} = 10$, $T_{\rm i} = 70$).

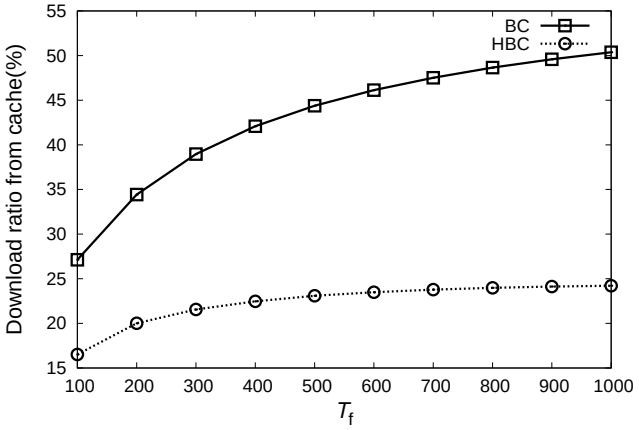

Fig. 5. Characteristics of download ratio from cache as a function of $T_{\rm f}$ ($T_{\rm q} = 10$, $T_{\rm i} = 70$).

From the figures, we can see that the relative delay ratio of the BC scheme increases while the download ratio from cache increases as the timeout threshold $T_{\rm f}$ increases. This is because the life time of BC increases and the possibility that files are transferred from further away increases, which means that, in one sense, cache is utilized efficiently. On the other hand, it is shown that the HBC scheme can suppress the relative delay ratio by about 10 to 18 points compared to the BC scheme. The download ratio from cache, however, does not increase as much as that of the BC scheme.

2) Influence of Query Interval: Next, the average query interval $T_{\rm i}$ per user was varied from 65 to 200. The timeout threshold $T_{\rm q}$ and $T_{\rm f}$ were fixed to 10 and $1,000$, respectively. Figs. 6 to 7 show the relative amount of traffic in the IP scheme, the relative delay ratio in the IP scheme, and the

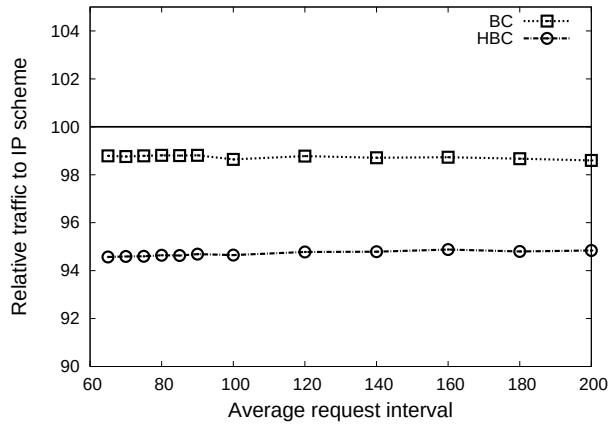

Fig. 6. Characteristics of relative traffic ratio in IP scheme as a function of request interval.

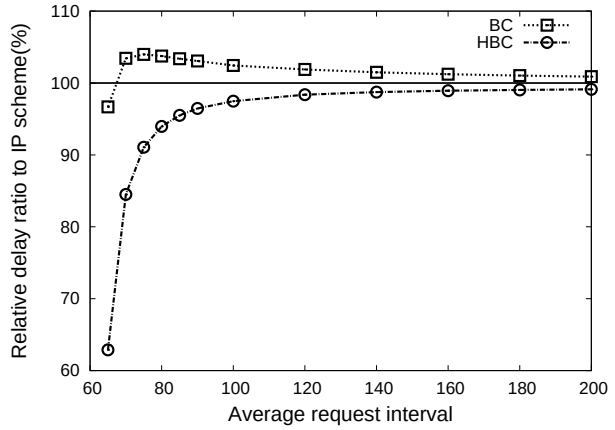

Fig. 7. Characteristics of relative delay ratio in IP scheme as a function of request interval.

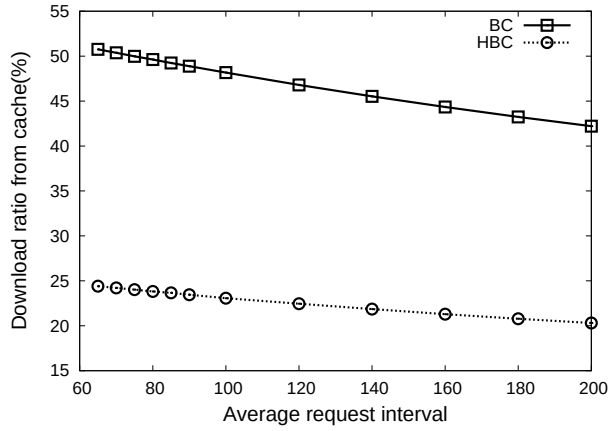

Fig. 8. Characteristics of download ratio from cache as a function of request interval.

download ratio from cache, respectively.

Fig. 6 shows that the HBC scheme reduces the amount of traffic compared to the BC scheme. This is because the HBC scheme shortens the length of the path on which the query and the file are transferred.

Fig. 7 shows that the relative delay ratio of HBC becomes smaller than that of BC as the query interval shortens, i.e. the offered load increases. For instance, the HBC scheme reduces the relative delay ratio in the IP scheme by about 34 points compared to the BC scheme. This is because even a slightly difference in traffic amount leads to a quite different queueing delay at high offered loads.

Fig. 8 shows that the download ratio from cache increases in both the BC scheme and the HBC scheme as the query interval shortens, i.e. the offered load increases. This is because more frequent requests for a file lead to more frequent updates of BCs or HBCs, so they are available for longer times. It is also shown that the BC scheme has larger download ratio from cache than the HBC scheme regardless of the offered load.

V. Conclusions

In this paper, we proposed the HBC scheme which takes account of hop-count information to reduce file acquisition delay and decrease the amount of traffic.

Simulations were conducted to compare the HBC scheme to the BC scheme and the IP scheme. Consequently, it was shown that the HBC scheme is effective since it suppresses not only the file acquisition delay but also the amount of traffic, even though the download ratio from cache increases.

Acknowledgement

We would like to express our deepest gratitude to Prof. Jim Kurose and Mr. Elisha Rosensweig who provided valuable comments and suggestions. We gratefully appreciate the financial support of Information and Communications Technology (NICT), Japan.

References

[1] Akamai Technologies, http://www.akamai.com, accessed May 16, 2011.
[2] BitTorrent, http://www.bittorrent.com/, accessed May 16, 2011.
[3] BRITE, http://www.cs.bu.edu/brite/, accessed May 16, 2011.
[4] E. J. Rosensweig and J. Kurose, "Breadcrumbs: efficient, best-effort content location in cache networks," Proc. IEEE INFOCOM 2009, pp. 2631–2635, April 2009.
[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," Proc. IEEE INFOCOM 1999, pp. 126–134, March 1999.
[6] P. Krishnan, D. Raz, and Y. Shavit, "The Cache Location Problem," IEEE/ACM Transactions on Networking, vol. 8, no. 5, pp. 568–582, Oct. 2000.
[7] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose, "The cache-and-forward network architecture for efficent mobile content delivery services in the future internet," Proc. Innovations in NGN: Future Network and Services 2008, K-INGN 2008, pp. 367–374, May 2008.
[8] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirection Infrastructure," ACM SIGCOMM Computer Communication Review, vol. 32, no. 4, pp. 73–86, Oct. 2002.
[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," Proc. CoNEXT 2009, pp. 1–12, Dec. 2009.
[10] L. Kleinrock, "Queueing Systems: Volume I —Theory," Wiley Interscience, New York, 1975.

 ISBN: 978-1-61208-148-9