On a Collaborative Caching in a Peer-to-Peer Network for Push-Based Broadcast

Kazuhiko Maeda[1], Wataru Uchida[2], Takahiro Hara[1], and Shojiro Nishio[1]

[1] Graduate School of Information Science and Tech., Osaka University
[2] Network Lab., NTT DoCoMo, Inc.
{k.maeda, hara, nishio}@ist.osaka-u.ac.jp
uchida@netlab.nttdocomo.co.jp

Abstract. In this paper, we propose a new collaborative caching strategy in a push-based broadcast environment where clients construct a peer-to-peer network by connecting with each other. In the proposed strategy, a client takes into account its own access probabilities and information on queries issued by other clients, and caches data items with large benefits of the response time. We confirm that the proposed strategy reduces the average response time by simulation experiments.

1 Introduction

Recently, there has been an increasing interest in research of a push-based broadcast system where a server delivers various data to clients, and they do not send any requests to the server but wait for the data to be broadcast. A key advantage of the push-based broadcast system is a higher throughput for data access from many clients. The push-based broadcast system is used for services where information with high publicity, such as movies, sounds, news, and charts. However, the server has to broadcast many kinds of data in order to satisfy clients' requests. This causes each client to wait data to be broadcast for a long time. To shorten the response time, several strategies for caching broadcast data at clients have been proposed [1,2]. These strategies calculate the benefit of response time from the client's access probability and the time factor (eg, broadcast cycle) of each data item, and cache data items with large benefits. These researches assume that clients have two ways to access data; access their own cache and listen broadcast data.

Today, there has been also an interest in a new type of information sharing called P2P systems [7,8]. In a P2P system, terminals called *peers* construct a logical network (*P2P network*) by connecting with each other. If a peer which wants a certain data item sends an access request (*query*) to its adjacent peers in the P2P network, the query be propagated until the query reaches a peer that holds the requested data item. Then, the data item is delivered to the peer that issued the query. Since each peer behaves in autonomous and distributed ways, this system has high scalability.

In a push-based broadcast system, it is expected that the average response time for data access could be further reduced if clients construct a P2P network

K.V. Andersen, J. Debenham, and R. Wagner (Eds.): DEXA 2005, LNCS 3588, pp. 879–889, 2005.

and they access requested data not only from the broadcast server and their own cache but also from the P2P network , i.e., other clients' cache. To the best our knowledge there is no conventional work that addresses caching strategies of broadcast data using a P2P network. In this paper, we assume that many clients that receive the push-based broadcast service construct the P2P network, and propose a new caching strategy by which clients collaboratively cache the broadcast data. In the proposed strategy, each client replaces its cache by taking into account not only its own access probabilities but also queries from other clients in order to reduce the average response time in the whole system.

The reminder is organized as follows. We introduce conventional caching strategies in section 2 and describe the system model in section 3. We propose a collaborative caching strategy in section 4, and evaluate it using simulation experiments in section 5. We show some related works in section 6. Finally, in section 7, we summarize this paper.

2 Conventional Caching Strategies

In PIX [1] and PT [2] strategies, it is assumed that clients can access data items from only their own cache or broadcast channel. When a client requests for a certain data item, it checks whether it caches the requested data item. If it does, it can access the item immediately. If not, it waits for the broadcast data item. The response time is the time interval until the data item is broadcast next.

PIX strategy

The algorithm of the PIX strategy is as follows.

1. The PIX value, $K(j) = p_j \cdot y_j$, is calculated by each client for each data item j $(1 \leq j \leq M)$. Here, M is the total number of data items which are broadcast by the server, p_j is the probability that the client accesses data item j, and y_j is the broadcast period of data item j.
2. The client caches γ data items which have the γ highest $K(j)$. Here, γ is the number of data items that the client can cache.

The PIX strategy reduces the response time of data access by caching items which have high access probabilities and long broadcast periods.

PT strategy

The algorithm of the PT strategy is as follow.

1. Every time when each data item, k, is broadcast, the PT values are calculated by each client for data items in the client's cache and data item k. The PT value, L_j, of data item j is calculated by $L_j = p_j \cdot (u_j(Q) - Q)$. Here, Q is the current time and $u_j(Q)$ is the time when data item j is broadcast next.
2. If a data item in the cache gives a lower PT value than L_k, data item j whose PT value is the lowest is replaced by k.

The PT value, L_j, represents the expected value of increase in response time if the client does not have data item j in its cache. The PT strategy compares the increases in response time of data items if they are discarded from the cache, and prefetches data items with larger gains in response time.

3 System Model

Figure 1 shows a system model assumed in this paper. In this system model, peers (clients) can send/receive data items to/from other peers in the P2P network. Each peer can cache a limited number of data items. When a peer wants to access a data item, it chooses a way that gives the shortest response time among the three access methods: accessing the item stored in its own cache, receiving the item from the broadcast, and receiving the item from another peer's cache.

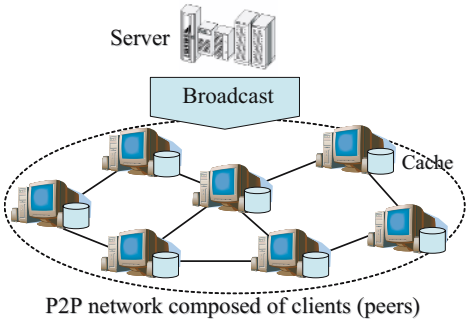

Fig. 1. Assumed environment

In this paper, it is assumed that the response time when accessing an item in its own cache is 0. Thus, if a request issuing peer holds the item in its own cache, it always accesses the cached item. If the peer does not hold, it compares the time remaining until the item is broadcast next with the time that is required to receive the item from another peer in the P2P network. If the former is shorter, the peer waits until the item is broadcast next. The response time when receiving a requested item from the broadcast is the time until the data item is broadcast next. If the latter is shorter, the peer checks whether another peer holds the requested item by using *flooding* [10]. In flooding, a peer issues a query with a certain TTL (Time To Live), and broadcasts the query to all its adjacent peers. If an adjacent peer does not hold the requested item, it re-broadcasts the query to all its adjacent peers, and this repeats until the query reaches a peer that holds the requested item or the logical hop count from the request issuing peer exceeds the TTL. If a peer that holds the requested item is found (in the following, it is denoted that the query "hits"), the peer sends a reply message to the request issuing peer. This message is sent to the request issuing peer through peers that relayed the query on the reverse direction. If the request issuing peer receives some reply messages, it receives the requested data item from the peer with the lowest logical hops. This data transmission is directly performed between the two peers using the physical network. If the query does not hit, the request issuing peer waits until the data item is broadcast next.

We also put the following assumptions:

- The system has a single broadcast server, and peers do not send any access requests to the server, i.e., pure push-based broadcast.
- All data items are of the same size and not updated. It takes one unit of time (one time slot) to broadcast one data item.
- Each peer knows the broadcast program. It can be realized by several ways, e.g., the server periodically broadcasts the program information.
- The delay of query propagation and the time to process a query is ignorable.
- The time to transmit a data item between every pair of peers is the same. We put this assumption for simplicity, but our proposed strategy can be easily extended to adapt an environment where transmission delays differ among peers.

4 Collaborative Caching Strategy

In this section, we propose a new collaborative caching strategy using a P2P network. In order to collaboratively cache data items, peers should know what data items are already cached by other peers and what data items are frequently accessed. However, since there are a huge number of peers in a push-based broadcast system, it is impractical that peers precisely know this information. Our main idea is that each peer guesses this information only using queries from other peers, e.g., arrival rate of query and results of data lookup. The proposed strategy shortens the average response time by determining cache replacement from this guessed information and its own access probabilities. This approach is reasonable because a query propagates only within a certain area determined by the TTL and thus the information guessed from queries indicates what items are cached and frequently accessed by neighboring peers within the TTL.

4.1 Query Information from Other Peers

To guess the above information, in the proposed strategy, each peer classifies queries that the peer issued or received from its neighbors. When a query arrives at a peer, the peer counts the query as one of the following three categories based on the result of looking up. Each of the three categories is counted for each data items.

- *F (Failure) query*: The query that did not hit, i.e., neither the peer nor further peers that the query propagated had the requested data item.
- *S (Success) query*: The query that hit, i.e., among the peer and further peers that the query propagated, at least one peer had the requested data item.
- *C (Connected) query*: The query that hit at the peer and the requested item was downloaded, i.e., the peer had the requested item and actually sent it to the query issuing peer.

Let us suppose a situation in which peer a is adjacent to peers b, c, and d as shown in Figure 2 and only peer d caches data item i. When a query requesting

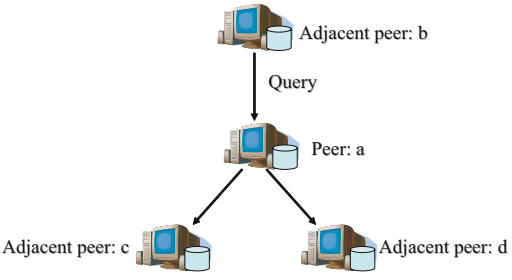

Fig. 2. Query propagation

item i propagates from peer b to peer a, the query is further broadcast to adjacent peers c and d since a does not have i in its own cache. Since only peer d caches item i, it sends a reply message to the query issuing peer via the path dab. From the reply message, peers a and b know that they are on the propagation path of the query that found data item i, and thus, a, b, and d count the query as an S query. If the query issuing peer received data item i from peer d, peer d counts the query as a C query instead of S query.

Here, it should be noted that a query is counted only once even if the same query or its results reached through multiple routes. The priority is given in the order of C, S, and F queries. For example, in the above case, if peer a receives the same query via another route and the TTL of the query is 0, the query cannot reach peer d and thus cannot find a peer that has the requested data item. In this case, while the query can be categorized to both S and F queries at peer a, only S query is counted according to the priority mentioned above.

By categorizing and counting queries, the following facts can be found.

- If a peer counts many F queries for a data item, it is shown that the data item is frequently requested by its neighboring peers including itself, but there is no neighboring peer that caches it.
- If a peer counts many S queries for a data item, it is shown that the data item is frequently requested by its neighboring peers including itself and some peers or itself cache it.
- If a peer counts many C queries for a data item, it is shown that the data item is frequently requested by its neighbors and the item cached by the peer is actually sent to the neighbors.

Increasing rates of the three categories dynamically change every time when a peer replaces its cache. For example, when data item i that is cached by no neighboring peers is frequently requested, many F queries are counted for item i. However, if one of the neighboring peers caches item i, many S queries will be counted at the peers, whereas many F queries had been counted until now.

4.2 Proposed Strategy

The collaborative caching strategy proposed in this paper extends the PIX strategy to take into account data accesses from other peers in the P2P network. The

proposed strategy, *C-PIX* (*Collaborative PIX*), calculates the benefits of the expected response time in the entire system when a peer replaces one of the cached data items with the broadcast data item. Based on the calculation, the C-PIX strategy determines the cache replacement.

For data item j in a peer's cache, the expected value of increase in response time in the entire system when the peer discards j from its cache is defined by the following equation:

$$U_j = P_j \cdot y_j/2 \ + C_j \cdot (y_j/2 - l). \tag{1}$$

We call this *the C-PIX value*. Here, l denotes the time required for sending a data item between two peers, P_j denotes the access probability of item i per unit time, and F_j, S_j, C_j denote the arrival rates of F, S, C queries per unit time.

For broadcast data item k which is not in the peer's cache, the expected value of decrease in response time in the entire P2P network when the peer caches k is defined as the k's C-PIX value, U_k. If $S_k = 0$ at peer A, it is likely that A's neighboring peers do not cache data item k, and peer A and its neighbors have to access k from the broadcast channel. Therefore, for peer A and its neighbors, the expected response time of accessing k is $y_k/2$. On the other hand, if $S_k > 0$ at peer A, at least one neighboring peer caches data item k and peer A can receive the data item from the peer. Thus, the expected response time of accessing k is l. From the above discussions, U_k is expressed by the following equation:

$$U_k = \begin{cases} P_k \cdot y_k/2 + F_k \cdot \alpha_k \cdot (y_k/2 - l) & (S_k = 0) \\ P_k \cdot l + F_k \cdot \alpha_k \cdot (y_k/2 - l) & (S_k > 0). \end{cases} \tag{2}$$

Here, α_k denotes the forecast ratio of F queries that will change to C queries when the peer caches data item k.

When a peer discards data item i from its cache, $\acute{\alpha}_i$ is set as the value of α_i, and then, α_i is changed by the following equation:

$$\alpha_i = x \cdot \acute{\alpha}_i \ + (1 - x) \cdot C_i / \{F_i + C_i\}. \tag{3}$$

Here, x $(0 \leq x \leq 1)$ is the parameter that determines how much the new α_i is influenced by the former one. When x is set to an unnecessary large value, the system cannot sensitively adapt to changes of the environment. It is important to determine an appropriate value of x considering the feature of the system.

In the C-PIX strategy, each peer calculates the C-PIX values for all data items stored in its cache and finds the minimum one, U_m, among them. If U_m is smaller than U_k, item m is replaced with broadcast item k. If the cache replacement occurs, the query counts of F, S, and C queries for the item discarded from the cache and the newly cached item are set to 0.

Now, we define the warmup time, T, that represents the time necessary for receiving enough queries for calculating the C-PIX value after caching a new data item. Until T units of time passes after caching data item i, U_i is calculated not by equation (1) but by the following equation:

$$U_i = P_i \cdot y_i/2 \ + \acute{F}_i \cdot (y_i/2 - l) \tag{4}$$

Here, $\acute{F}_i$ denotes the value of F_i before caching data item i.

5 Performance Evaluation

5.1 Simulation Environment

It is known that an unstructured P2P network constructed on the Internet follows the *power-law* [6]. Based on this fact, we determined the degree of peer j, d_j, which is the number of j's adjacent peers in the P2P network, by the following equation:

$$d_j = \left\lfloor w_{max} \cdot r_j^{\mathcal{R}} \right\rfloor . \quad (\mathcal{R} < 0) \tag{5}$$

Here, r_j denotes the rank of peer j, which is its index in the descending order of outdegree (number of adjacent peers), and w_{max} denotes the maximum number of adjacent peers. For simplicity, we assume $r_j = j$. A network which is constructed by connecting peers at random according to the power-law is called a *PLRG* (*Power-law Random Graph*). In our simulations, the number of peers was set to 500 and we used a PLRG network, where (w_{max},$\mathcal{R}$) is $(240, -0.8)$ as many conventional works did [6]. Here, it is known that $\mathcal{R}$ of the real network is approximately -0.8.

The access probability at each peer was determined based on the Zipf distribution [14], where the following two different distributions were used:

Access distribution 1 (A. D. 1)F
The smaller the identifier of each data item is, the higher the probability that the data item is accessed. The order of access probabilities of data items is the same at all peers. However, the values of the access probability of each data item are not the same among peers but vary a little. Specifically, access probability p_{ji} of item i at peer j was given by the following equation:

$$p_{ji} = \frac{i^{-\theta_j}}{\sum_{k=1}^{M} k^{-\theta_j}}. \tag{6}$$

Here, θ_j is called a Zipf coefficient, and if this is set to a large value, a small number of data items are accessed frequently. In our simulations, θ_j was also determined based on the Zipf distribution by the following equation:

$$\theta_j = \frac{j^{-0.8}}{\sum_{k=1}^{MAX\_PEER} k^{-0.8}}. \tag{7}$$

Here, MAX_PEER denotes the total number of peers.

Access distribution 2 (A. D. 2)F
Every peer has different orders of access probabilities of items. Access probability p_{ji} of item i at peer j was given by the following equation:

$$p_{ji} = \frac{\{(i - h_j + 1) \bmod M\}^{-0.5}}{\sum_{k=1}^{M} k^{-0.5}}. \tag{8}$$

Table 1. Average response time varying x and T (A. D. 1)

T (Time slot) \ x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	36.7	33.8	31.9	31.5	34.6	42.6	56.8	88.4	168.4
200	29.9	38.0	51.4	69.7	94.4	128.1	180.6	258.8	306.7
300	38.2	53.1	73.0	97.2	129.0	173.1	222.4	259.7	280.0
400	50.5	69.3	93.1	123.0	162.0	193.2	209.9	220.8	218.8
500	48.8	64.2	83.1	106.0	119.9	128.1	132.4	134.9	136.5
600	63.8	91.8	122.1	106.3	205.2	231.6	249.2	263.0	265.2
700	51.7	68.2	89.5	112.9	138.2	165.9	186.9	197.6	193.4
800	44.6	56.7	72.1	88.5	108.9	134.4	161.0	167.3	159.3
900	42.7	53.7	66.8	84.2	106.2	127.1	142.8	144.5	135.5
1000	83.7	92.3	98.2	100.7	101.0	103.2	102.5	102.8	104.7

Here, h_j denotes the item which peer j accesses most frequently. The probability that peer j accesses data item i most frequently (namely, $h_j = i$) was also determined based on the Zipf distribution by the following equation:

$$q_i = \frac{i^{-0.8}}{\sum_{k=1}^{M} k^{-0.8}}. \tag{9}$$

The probability that each peer issues an access request at each time slot was set to 0.1. Therefore, the access frequency of data item i at peer j becomes $P_{ji} = p_{ji} \times 0.1$.

We assumed that the server broadcasts all data items periodically. Initially, data items were cached at each peer according to the PIX strategy. The initial value of α_i for each data item at each peer was set to 1. The total number of data items was set to 1,000, the TTL of each query was set to 3. The download time l of a data item from the P2P network was set to 10 time slots.

Based on the above simulation environment, we evaluated the average response time of the proposed strategy during 300,000 time slots. For the purpose of comparison, we also evaluated the average response times in the cases where peers can receive data items from the P2P network and determine the cache replacement based on the PIX and PT strategies.

5.2 Impact of x and T

In order to determine appropriate values of x and T, we evaluated the average response times of the proposed strategy where x varies from 0.1 to 0.9 and T varies from 100 to 1,000. The maximum number of data items which a peer can cache (cache size) was fixed to 100, and the access probability of each peer was given according to A. D. 1. Table 1 shows the result. A gray part in both tables indicates the minimum value of average response time for each value of T.

From this result, it is shown that two parameters, x and T, are correlated. As T gets smaller, x that gives the shortest average response time gets higher. This is because the proposed strategy determines the cache replacement by guessing what items neighboring peers cache based on received queries and their results. If T is too small, C queries cannot be counted sufficiently since there is not

enough time to collect the query information. Therefore, items which are in fact needed from neighboring peers may be judged to be unnecessary, and thus, are discarded from the cache. Moreover, if C queries are not counted sufficiently, α_i decreases as shown in equation (3), and thus, the cached item is discarded in a short time. We can solve this problem by setting x large (see equation (3)).

The shortest response time is given where $x = 0.1$ and $T = 200$ in the simulation environment. Therefore, we use these values in the following simulations.

5.3 Impact of Cache Size

Fig.3 and Fig.4 show the average response times of the proposed strategy and the other two strategies where the cache size varies from 0 to 1000. We applied A. D. 1 in Fig.3, and A. D. 2 in Fig.4.

From these results, the C-PIX always gives the shortest average response time. Moreover the difference in performance is larger when using A. D. 1 than using A. D. 2. When A. D. 1 is used, all peers have the similar access characteristics, and thus, in the PIX and PT strategies, they cache the same items. As a result, they can hardly find requested items in the P2P network. On the contrary, in the C-PIX strategy, each peer determines the cache items by taking into account items cached by neighboring peers. Even when the cache size of each peer is small, this strategy can improve the hit ratio of queries and shorten the average response time.

6 Related Works

P2P systems are classified into two categories; *structured* [11,13] and *unstructured* [3]. Structured systems have precise control over the network topology and locations of data items in the whole network, while unstructured ones do not. Since blind methods such as flooding are used to look up requested data items in unstructured one, they also have a disadvantage that network traffic and looking up delay are larger than structured ones. Instead, unstructured ones have an advantage of being built easily and flexibly. This is because most P2P systems currently in service use unstructured networks for data looking up [7,8]. In this paper, we assumed an unstructured system.

There are many conventional works that address collaborative caching in some research fields such as web caching [5], distributed file systems [4,12], and adhoc network [9]. For example, in the research field of web caching, several strategies in which proxy servers collaboratively cache Web contents. These strategies aim to reduce the network traffic and balance the processing load of Web servers which hold original contents. In one of strategies [5], proxies are hierarchically coupled like DNS (Domain Name System), where the root of the hierarchy is the server which holds original contents. When a client requests contents, it first asks whether the proxy which is responsible to its domain caches them. If not, the request is forwarded to proxies of higher level in the hierarchy. These approaches are similar to structured P2P systems since proxies are hierarchically coupled and data requests are routed based on particular rules, and thus, contrary to our approach.

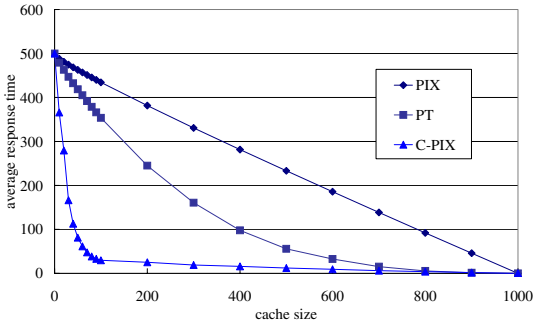

Fig. 3. Cache size vs. average response time (A. D. 1)

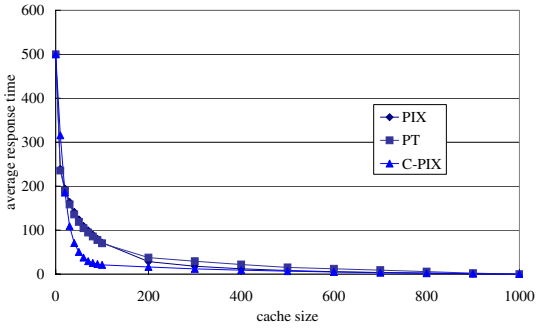

Fig. 4. Cache size vs. average response time (A. D. 2)

7 Conclusion

In this paper, we propose a new caching strategy in a push-based broadcast system where clients compose a P2P network. To reduce the average response time in the entire system, in the proposed strategy, each client autonomously determines cache replacement by taking into account not only its own access probabilities to data items but also queries issued from other peers. From the results of simulation experiments, we confirmed that the proposed strategy, C-PIX, gives better performance than the conventional caching strategies.

The C-PIX strategy is an extension of the PIX strategy which is a typical caching strategy in a push-based broadcast system. We also plan to consider another collaborative caching strategy that is based on the PT strategy.

Acknowledgements

This research was supported by The 21st Century Center of Excellence Program "New Information Technologies for Building a Networked Symbiotic Environ-

ment" and Grant-in-Aid for Scientific Research on Priority Areas (16016260) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Acharya, S., Alonso, R., Franklin, M., and Zdonik, S.: Broadcast Disks: Data Management for Asymmetric Communication Environments, Proc. ACM SIGMOD'95, pp. 199–210 (1995).
2. Acharya, S., Franklin, M., and Zdonik, S.: Prefetching from a Broadcast Disk, Proc. ICDE'96, pp. 276–285 (1996).
3. Cohen, E., and Shenker, S.: Replication Strategies in Unstructured Peer-to-Peer Networks, Proc. ACM SIGCOMM'02, pp. 177–190 (2002).
4. Dahlin, M., Wang, R., Anderson, T., and Patterson, D.: Cooperative Caching: Using Remote Client Memory to Improve File System Performance, Proc. Symp. on Operating Systems Design and Implementation, pp. 267–280 (1994).
5. Fan, L., Cao, P., Almeida, J., and Broder, A.: Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol, Proc. ACM SIGCOMM'98, pp. 254–265 (1998).
6. Faloutsos, M., Faloutsos, P., and Faloutsos, C.: On Power-Law Relationships of the Internet Topology, Proc. ACM SIGCOMM'99, pp. 251–262 (1999).
7. FreeNet, <URL:http://freenet.sourceforge.net>.
8. Gnutella, <URL:http://gnutella.wego.com>.
9. Hara, T.: Cooperative Caching by Mobile Clients in Push-based Information Systems, Proc. ACM CIKM'02, pp. 186–193 (2002).
10. Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks, Proc. Int'l Conf. on Supercomputing, pp. 84–95 (2002).
11. Ratnasamy, S., Francis, P., Handley, M., and Karp, R.: A Scalable Content-Addressable Network, Proc. ACM SIGCOMM'01, pp. 161–172 (2001).
12. Sarkar, P., and Hartman, J.: Efficient Cooperative Caching Using Hints, Proc. Symp. on Operating Systems Design and Implementation, pp. 35–46 (1996).
13. Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan. H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, Proc. ACM SIGCOMM'01, pp.149–160 (2001).
14. Zipf, G. K.: Human Behavior and the Principle of Least Effort, Addison-Wesley (1949).