Popularity-Based Content Replication in Peer-to-Peer Networks

Yohei Kawasaki, Noriko Matsumoto, and Norihiko Yoshida

Department of Information and Computer Sciences,
Saitama University, Saitama 338-8570, Japan
{yohei, noriko, yoshida}@ss.ics.saitama-u.ac.jp

Abstract. Pure peer-to-peer (P2P) networks is widely used, however they broadcast query packets, and cause excessive network traffic in particular. Addressing this issue, we propose a new strategy for content replication which prioritizes popular and attracting contents. Our strategy is based on replication adjustment, being cooperated with index caching, as well as LRU-based content replacement, and a more effective replica placement method than well-known probabilistic ones. We also present some experiment results to show that our strategy is better than other ones in regards to network traffic and storage consumption.

1 Introduction

In a P2P network, as contents are distributed to all the member nodes (peers), we must consider mechanisms to search contents. Generally, the search mechanisms for P2P networks are classified into three groups. A centralized P2P system such as Napster [1] has a central server which manages all the locations (indices) of contents. A decentralized P2P system has no central server, and is again classified into two categories. A decentralized unstructured system such as Gnutella [2] has no specific structure in network topology, and use a flooding-based query algorithm. A decentralized structured system such as Chord [3] has a well-organized structure, and has a very efficient search mechanism using a distributed hash table (DHT) on P2P network.

In centralized P2P systems, a large amount of queries cause a high load on the central server, and the server may be a single point of failure. Decentralized structured P2P systems require strictly-organized network topology which is difficult to construct in reality. Both of them can search contents efficiently, however, they have disadvantage as mentioned above. On the contrary, decentralized unstructured P2P systems are easy to construct in reality, and fault-tolerant. Therefore, they are widely used, although they have issues that the number of query packets grows exponentially, and search areas are limited to reachable nodes of query packets.

It is one of the effective improvement way for the issues of decentralized unstructured P2P systems to distribute replicas of contents in a network [4, 5]. Some researches about content replication treat all the contents equally for replication.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part IV, LNCS 3994, pp. 436–443, 2006.

However, the popularity of a content (i.e. frequency of accesses to the content) is not uniform. It must be more effective to put more replicas for popular objects than unpopular ones.

This paper proposes a distributed replication mechanism for decentralized unstructured P2P networks which considers content popularity, is scalable, and is easy to implement. Section 2 summarizes related researches on content replication, and Section 3 proposes popularity-based content replication. Section 4 presents some experiment results and evaluation. Section 5 contains some concluding remarks.

2 Content Replication

Content replication in P2P network generally provides decrease of packet hops until a search succeeds, and improvement of search success rate. It also provides decrease of network load, and realizes efficient search. These effects become apparent in decentralized unstructured P2P systems in particular.

In an extreme case, allocating replicas of all contents to all nodes results in the ideal result in which reference to any content is done at no network cost. However, because a node has a limited storage resource, we cannot actually apply this extreme replication.

Thus, in the content replication, it is important to decide the number and the placements of replicas how many and where the replicas are allocated. There are some strategies already proposed to decide these.

Owner Replication. Replicas is allocated only on the requester (i.e. the node emitting a query) when the search succeeds. This strategy is simple and needs the smallest cost, but it needs enough time for replicas to spread over the network [5].

Path Replication. Replicas are allocated on all the nodes on the search path from the requester to the holder (i.e. the node having the requested content). Because one or more replications are allocated per search, a lot of storage or network resources are needed, however, a content spreads easily [5].

Path Random Replication. In this strategy, each node on the search path may or may not have a replica based on a pre-defined probability. We must decide an appropriate allocation probability [6].

Random Replication. In this strategy, replicas of the same number as of the nodes on the search path are allocated to randomly-chosen nodes in the whole network. There is no distributed algorithm presented to choose the nodes, therefore it is not certain whether the strategy can be implemented without too much cost [5].

3 Popularity-Based Content Replication

It is reported that the "Power law" can generally be applied to web content accesses [7]. There are a few contents which attract many accesses, while there are many contents which attract only a few accesses. This fact must be applied to content accesses in P2P networks as well.

Allocating more replicas of more popular contents brings improvement of search success as a whole because popular contents are more often searched. It must also benefit to reduction of network traffic because the number of hops to reach a searched content becomes small. However, too many replicas causes increase of the network traffic because many replicas are found by a single query, and many "hit" packets are generated. At the same time, suppressing the number of replicas of unpopular contents brings reduction of storage consumption at every node.

There are some replica placement strategies proposed as summarized above. However, none of them considers content popularity. Therefore, these strategies result in excessive network traffic to create a content replica for low popularity contents (except for Owner Replication) and excessive storage consumption.

Square-Root Replication is one of a few exceptions, which considers content popularity [4]. The number of replicas is determined proportional to the square root of the access counts relative to other contents. This strategy is reported to exhibit, in a rough (not an exact) simulation, significant reduction of whole search traffic [5]. However, it would be necessary that any single node must have knowledge on popularities of all the contents, which is impractical.

Consequently, we construct our replication strategy based on the following design.

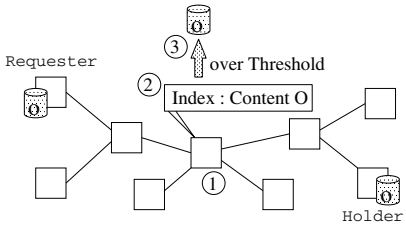

Fig. 1. Delayed content replication

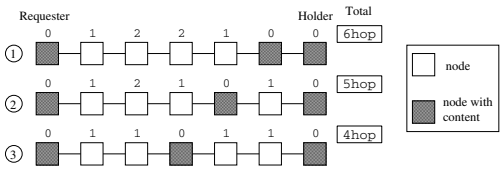

Fig. 2. Replication on search path

(1) Decision of Replica Allocation. A simple method to decide replica allocation is introduced which does not cause heavy network traffic.

Fig. 1 shows an overview of the method. A replica is allocated on the requester like Owner Replication. Another will be allocated on the node halfway between the requester and the holder, namely the most distant node from both the requester and the holder (Fig. 1-①), however it is not placed at the moment of searching immediately. As shown below, the node evaluates popularity, i.e. access rate, for that content, and decides whether placing a replica or not eventually.

Fig. 2 shows the reason why the halfway node is the best to place a replica. Sum of the number of hops is minimized if a replica is placed on the node halfway on the search path. This is more effective than probabilistic allocation.

(2) Provisional Replica Placement. A replica is not placed immediately at the halfway node. If the replica is never accessed afterward, it is just a waste

of copying cost and storage. Instead, only an index of the content, i.e. a pair of the search key and the content location, is initially placed (Fig. 1-②). The size of an index is much smaller than that of a content, therefore the cost of keeping an index is much smaller that the one of copying and storing the content.

The node has an index containing two content locations of both the holder and the requester, and it replies either of these locations in an alternate (round-robin) fashion to a query. Accesses to the content are dispersed in this way.

The index not only contributes to the efficient search, but also indicates the potential location of the replica. This cooperation of indexing and content replication is the major novelty of this research.

The node counts the number of references to each index it has, and when the number exceeds a certain threshold, a replica of the content is placed at this moment (Fig. 1-③).

(3) Replacement of Replication. Each node has limited capacity of storage, and when its storage is saturated, it decides which replicas to keep according to their popularities. When a new content or replica is added to the node whose storage is full, a replica is discarded in a LRU (Least Recently Used) manner.

4 Experiments and Evaluation

To verify the advantages of our strategy, we present some results of experiments using a simulator for virtual P2P networks. Table 1 summarizes parameter settings for the experiments.

Table 1. Simulation settings

Number of nodes	2000
TTL of query packet	5
Initial max number of contents	60
Capacity of storage	100
Threshold of replication	15

Table 2. Popularities vs. contents disposition

popularity	request rate (%)	contents
(Low) 1	5	2500
2	10	1000
3	20	400
4	25	150
(High) 5	40	50

Settings for Contents and Search. We put at each node some contents randomly up to 60 in the beginning. These initial contents will not be discarded even if the node's storage is full. We allow each node to have up to 1,000 indices, and to discard indices in a FIFO (First In First Out) manner when the index capacity overflows.

We prepare 4,100 contents as a whole, each of which has a popularity level shown in Table 2. This is to follow the Power-Law described in 3.

Our simulator selects a node randomly from 2,000 nodes per trial, and makes it search a content. We tried 100,000 trials in every experiment. The result figures below show transition at every 2,000 trials.

Settings for Network Topologies. It is reported that actual structures of P2P network topology have the nature of Power-Law [8]. In fact, a small number of nodes has many links, and most nodes has only a few links. Therefore, we follow such network topology especially in experiments described in 4.2. The Power-Law Random (PLR) topology used in the experiments is the random network in which the number of links of each node follows the Power-Law, and connections between the nodes are random.

4.1 Effect of Each Factor

Our strategy consists of replica allocation which cooperates with index allocation, and the LRU-based replacement of replicas. Hence, to clarify each effect, we compare four cases: (1) Index OFF - FIFO, (2) Index OFF - LRU, (3) Index ON - FIFO, and (4) Index ON - LRU. "Index OFF" means not allocating an index but a replica immediately. In this experiment, we simplify the network composition in which each node randomly connects with 2 to 4 neighbors.

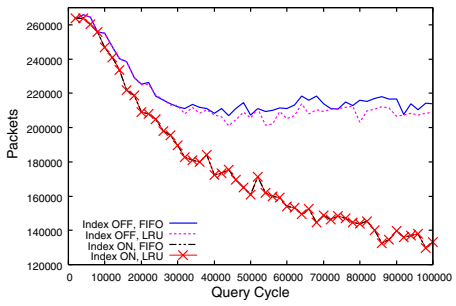

Fig. 3. Transitions of total packets

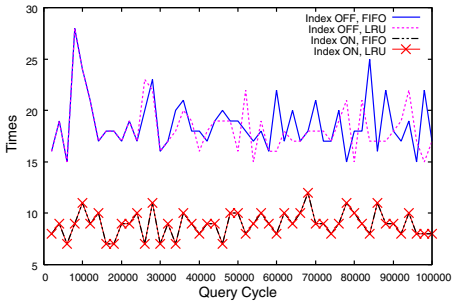

Fig. 4. Max. number of content delivery

Fig. 3 shows transitions of the total number of packets, i.e. sum of query packets and hit packets. (4) shows identical result as (3), because there is no saturation of storages in this experiment. "Index ON" brings delayed replica placement, which suppresses storage saturation. Comparing (1) and (3), or (2) and (4), we can conclude that cooperation with indexing reduces the network traffic significantly. Comparing (1) and (2), LRU-based replacement is also effective when storage saturation occurs.

Fig. 4 shows transitions of the maximum number of content deliveries. If this value is high, it means that accesses for a content are concentrated to a single node. Again, (4) shows identical result as (3). Comparing (1) and (3), or (2) and (4), accesses are dispersed by cooperation of indexing, and by round-robin handling of queries.

We also verify, although not apparent in the figure, that (3) (and (4)) shows the highest success rate of search out of the four cases, and the total number of content delivery is the highest.

4.2 Comparisons with Other Strategies

Next, we compare our strategy with other related strategies: (1) Our strategy (Proposal), (2) Owner Replication (Owner), (3) Path Replication (Path), and (4) Path Random Replication in which the allocation probability to each node is 10% (Path Random). The network composition we use in these experiments are the above-mentioned PLR topology.

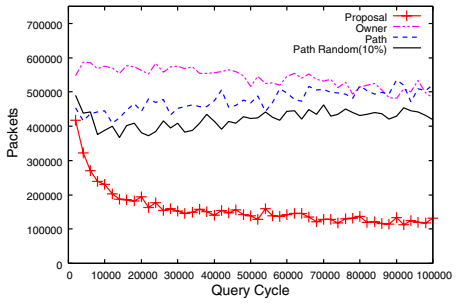

Fig. 5. Transitions of total packets

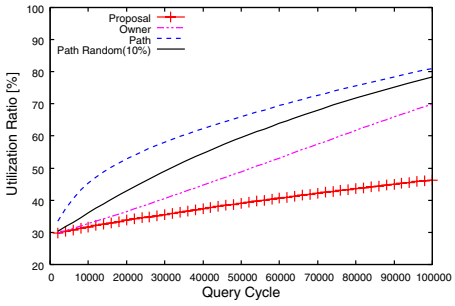

Fig. 6. Transitions of storage utilization

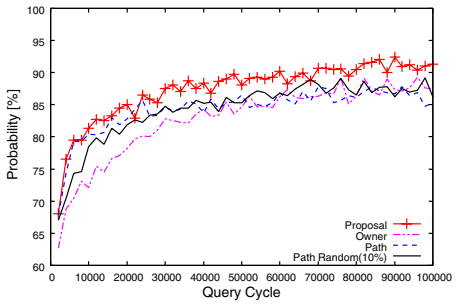

Fig. 7. Transitions of search success ratio

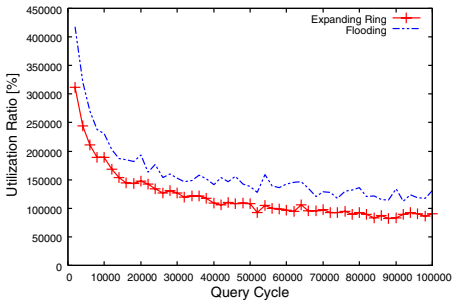

Fig. 8. Flooding vs. expanding ring

Fig. 5 shows transitions of the total number of packets of all the strategies. Only our strategy shows the steep decline from the beginning, while the others do not.

Actually, all the strategies show declines when applied to networks of random topology. Therefore, the results shown in this figure must be affected by the characteristic of PLR topology. In the PLR topology, most packets tend to concentrate on hub nodes (i.e. nodes with many links), and our LRU-based replacement of replicas occurs more often on them. In the other strategies, search success ratio on the hubs gets lower, and query packets tend to spread much wider.

In this figure, Path Replication ("Path") shows increase of traffic. We investigated and found that many "hit" packets occupy the network which are

generated by too many replicas. In fact, about 30% of total packets are "hit" packets.

Fig. 6 shows transitions of the consumption ratio of all the node storages. Our strategy shows the lowest ratio, therefore we can conclude ours achieves good performance in regard to both network traffic and storage consumption.

Fig. 7 shows transitions of search success ratio. Our strategy shows the best, that implies ours suppresses flooding of query packets.

4.3 Using Expanding Ring for Search

The flooding-based search algorithm used in decentralized unstructured P2Ps spreads query packets in the networks, and causes heavy network traffic. One of the proposals addressing this issue is "Expanding Ring", which expands search area gradually by incrementally increasing TTL (time-to-live) [5]. Here we show comparison of plain flooding and expanding ring when applied in cooperation with our strategy.

Fig. 8 shows transitions of the total number of packets under plain flooding and expanding ring. Expanding ring brings lower traffic from the beginning, and it is effective in particular at the beginning where replicas are not so many yet. However, we observe that using expanding ring, spreading of replicas becomes slow, and search success rate decreases, because searches tend to succeed within a small area.

5 Conclusions and Future Work

This paper focuses on content replication in decentralized unstructured P2P networks for the purpose of reduction of excessive traffic and improvement of search efficiency, proposes a strategy which makes cooperative use of indexing and content replication.

Our strategy is composed mainly of controlled (or delayed) content replication which cooperates with index allocation. Additionally, it implements LRU-based replica replacement, and a very simple but efficient method to decide locations of replicas. We showed advantages of our strategy by several simulated experiments. Our strategy achieved more efficient search and lower network traffic than other strategies.

Future works are as follows. First, a threshold of reference counts for switching from an index to a replica placement must be appropriately determined. Second, other P2P network topologies must be examined as well. Third, real-world experiments must be performed.

Acknowledgments

This research was supported in part by JSPS in Japan under Grants-in-Aid for Scientific Research (B) 17300012, and by MEXT in Japan under Grants-in-Aid for Exploratory Research 17650011.

References

1. Napster website, http://www.napster.com/.
2. Gnutella website, http://www.gnutella.com/.
3. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", Proc. ACM SIGCOMM 2001, 2001.
4. E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks", Proc. ACM SIGCOMM 2002, 2002.
5. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstrctured Peer-to-Peer Networks", Proc. 16th ACM Int'l Conf. on Supercomputing, 2002.
6. D. Maruta, H. Yamamoto, Y. Oie, "Replication Strategies to Enable Storage Load Balancing in P2P Networks" (in Japanese), IEICE Technical Report, NS2003-319, pp.131–136, 2004.
7. S. Sato, K. Kazama, and S. Shimizu, http://www.ingrid.org/w3conf-japan/97/sato/paper.html (in Japanese).
8. L. A. Adamic, R. M. Lukose, A. R. Puniyani and B. A. Huberman, "Search in Power-Law Networks", Physical Review E, Vol.64, pp.46135–46143, 2001.