Journal of Network and Computer Applications 35 (2012) 85–96

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Proactive replication for rare objects in unstructured peer-to-peer networks

Guoqiang Gao, Ruixuan Li, Kunmei Wen *, Xiwu Gu

Intelligent and Distributed Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, PR China

ARTICLE INFO

Article history:
Received 25 August 2010
Received in revised form
9 January 2011
Accepted 24 February 2011
Available online 5 March 2011

Keywords:
Peer-to-peer network
Rare object
Replication
Search
Object popularity

ABSTRACT

Unstructured peer-to-peer (P2P) networks have become a very popular architecture for content distribution in large-scale and dynamic environments. The search efficiency problem in unstructured P2P networks has not been adequately addressed so far, especially concerning search for rare objects. In this paper, we propose a proactive replication strategy to improve search efficiency for rare objects. It uses an object-probing technique for peers to decide whether or not to establish replications for their objects when they join the network. This strategy can effectively increase the popularity of rare objects in order to enhance search efficiency. We also present a rare object search algorithm to reduce the overhead caused by the replication strategy. When a peer forwards a search request, the forward probability is calculated according to its neighbors' degrees and the number of neighbors' objects. Therefore, the search request is forwarded to the peers more likely containing target objects. Simulations show that our proactive replication strategy greatly improves search efficiency for rare objects with moderate communication overhead. The rare object search algorithm not only improves search efficiency for rare objects, but also achieves load balance in search.

1. Introduction

The peer-to-peer (P2P) networks can be broadly categorized into two types: structured P2P networks and unstructured P2P networks. Structured P2P networks (Ratnasamy et al., 2001; Stoica et al., 2001; Gupta et al., 2003; Zhao et al., 2002) implement a Distributed Hash Table (DHT) and provide one basic operation: given a key, they map the key to a peer. By carefully guiding the queries in the network, these systems are able to drastically decrease search traffic and increase search efficiency. It is commonly believed that structured P2P networks are more expensive to maintain than unstructured P2P networks and the constraints imposed by the structure make them hard to improve scalability. Unstructured P2P networks (Heinla et al., 2001; Sarshar and Roychowdhury, 2007; Zhang and Hu, 2007; Clarke et al., 2000) account for millions of users dynamically connected in an ad hoc fashion, and queries, which are done through a path, are selected randomly according to a uniform distribution. However, unstructured P2P networks may have high search traffic and poor search efficiency since those networks are prone to failures, errors and malicious peer behaviors that are frequent in a large scale unsupervised P2P environment (Kong et al., 2006). Structured P2P networks are these ones that strictly control the underlying network structure, content publication strategy and query routing. Unstructured P2P networks, on the other hand, are those that impose minimal constraints on the network topology or content distribution. The search mechanism in unstructured P2P networks can be simply described as "to give a try", therefore, they are suited to retrieve highly replicated data, but have limitations for rare information retrieval.

A typical unstructured P2P network is Gnutella (Frankel and Pepper, 2000), which implements searches through a flooding technique with a certain scope. This type of network has excessive traffic forms that account for more than 50% of network traffic (Ripeanu, 2001). Moreover, unstructured P2P networks have poor search efficiency, especially for rare objects. Such a problem has been addressed by various algorithms, such as flooding, expanding ring (Lv et al., 2002) and random walks (Gkantsidis et al., 2004). Most of these algorithms are in fact very effective for locating popular objects. However, when it comes to finding rare objects in unstructured P2P networks, the similar operations using these algorithms become less effective and less efficient (Qiao and Bustamante, 2006). As shown in Loo et al. (2004), as much as 18% of all queries return no response even when results are available in the widely used Gnutella network.

This paper introduces a mechanism to address the above-mentioned problems occurred in unstructured P2P networks. Our main idea results from taking a proactive replication strategy to increase rare object popularity over unstructured P2P networks and using our search scheme to improve search efficiency for those rare objects. In this article, the object popularity is defined

* Corresponding author. Tel.: +86 27 8754 4285; fax: +86 27 8754 5004.
E-mail addresses: ggq@smail.hust.edu.cn (G. Gao), rxli@hust.edu.cn (R. Li), kmwen@hust.edu.cn (K. Wen), xwgu@hust.edu.cn (X. Gu).

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.02.007

as the proportion of the object number in the network to the network size. In unstructured P2P networks, the objects with high replication can be successfully found within a small number of hops in search. As for the rare objects, to locate them, we must extend the search scope, even to the entire network. This will bring greater communication overhead and search delay. If an item is very rare, it is hard for any query mechanisms to find it with limited number of steps. Therefore, we must increase the rare objects popularity instead of extending the search scope. However, we do not know which object is rare in a distributed P2P network because the peers only keep the local information. In this paper, we propose a proactive probing technique to address this issue. Peers search the objects they own in the network at a certain frequency. If there are no hits, they replicate the objects at other nodes found by some strategies. In this article, we use the terms of "node" and "peer" interchangeably. We call this strategy proactive replication. The objective of this strategy is to increase the number of rare objects and the probability of successful searches. We found that it obviously improves search efficiency for rare objects through simulated experiments.

To decrease communication overhead caused by the replication strategy, we need a search algorithm to successfully locate an object with minimal object popularity. We will introduce some characteristics of unstructured P2P networks which can be used in search strategies before discussing our search strategy for rare objects. Unstructured P2P networks such as Gnutella are power-law distribution networks. That is, the majority of the network nodes have low degree, while a few other nodes have high degree. Moreover, some nodes have a lot of resources, and some nodes have fewer objects. Therefore, searching the nodes with high degree and many objects is more likely to hit the target. In the search, we can take advantage of these features to forward queries to those peers more likely containing target objects in order to achieve the goal-oriented routing, which is similar to that of the structured P2P network. We call this strategy a rare object search. Experiments show that it not only reduces the communication overhead of the search algorithm, but also minimizes the cost of the replication strategy.

The rest of this paper is organized as follows. Section 2 compares our strategy to related works. In Section 3, we present a proactive replication strategy. In Section 4, we discuss a rare object search algorithm. We describe the experimental settings in further detail in Section 5. Section 6 demonstrates our algorithms through simulated experiments. Finally, in Section 7, we summarize our contributions.

2. Related work

To find an object, a node queries its neighbors in unstructured P2P networks. The most typical query method is flooding, where the query is propagated to all neighbors within a certain radius. Flooding introduces duplicate messages and duplicate queries, which create pure overhead. Since flooding has inherent limitations, some improved methods have been proposed, such as expanding ring (Lv et al., 2002). Although this approach has less message duplications than overall flooding, it produces many extra traffic messages, especially for querying rare objects. Filali and Huet (2010) propose a dynamic time-to-live (TTL) scheme. Instead of decreasing the query TTL by 1 at each hop, it is decreased by a value v where $0 < v < 1$. Their aim is not only to redirect queries towards the right direction but also to stimulate them in order to reliably discover rare resources. Xu et al. (2010) design a data structure called traceable gain matrix (TGM) that records every query's gain at each peer along the query hit path, and allows for optimizing query routing decision effectively. The query cache strategy (Yin et al., 2005) can also reduce the communication overhead, and shortens the length of the queries. With this strategy, nodes cache the results of successful queries so that the same subsequent queries can quickly hit the target through the caches. This mechanism not only improves search efficiency, but also reduces communication overhead. Search efficiency of these improvements is significantly increased for those hot resources or more replicated resources, but the efficiency remains low for rare objects.

Studies (Gkantsidis et al., 2004; Chawathe et al., 2003) have shown random walk to be significantly more efficient than flooding, which forwards a query message to some randomly chosen neighbors at each step until the object is found. When using the standard random walk (with one walker), it significantly cuts down the message overhead, by an order of magnitude compared to expanding ring across the network topologies. However, the scheme has low search efficiency (i.e., high delay). To decrease the delay, Gkantsidis et al. (2005) increases the number of "walkers". That is, instead of just sending out one query message, a requesting node sends k query messages, and each query message takes its own random walk. The simulations confirm that k walkers after S steps reach roughly the same number of nodes as one walker does after kS steps. In this paper, we will only take random walk as a search method in the analysis of object popularity and disregard the dimension of random walk.

Published measurements of unstructured P2P networks such as Gnutella show that they have power-law degree distributions (Adamic et al., 2001). This distribution reflects the existence of a few nodes with very high degree and many with low degree. Adamic (Adamic et al., 2003) proposed a high degree seeking strategy, in which a search request is forwarded to the high degree node because they have more neighbors. However, this method easily leads to information gathering (almost all queries are processed in high degree nodes), and harms the query distribution. Based on power-law characteristic and some other features we observed, we improve the high degree seeking strategy through probability forward. Our scheme tries to balance the load of each node in queries with similar search efficiency.

Replication will increase search efficiency of popular objects through storing the object at the requester or all nodes along the searching path when a search is successful. Presently, the existing replication strategies are mainly passive, which cannot help rare objects. Plover (Shen and Zhu, 2009) is a proactive file replication scheme with low-overhead. By making file replicas among physically close nodes based on nodes' available capacities, Plover not only achieves high efficiency in file replication but also supports low-cost and timely consistency maintenance. In iDARE (Liao et al., 2010), peer can proactively replicate data chunks to stable cache servers for future sharing, when it has high possibility to leave the overlay. This strategy can solve the problem of that media data cached on peer disk turns offline and unavailable with frequent peer departure. Cohen (Cohen and Shenker, 2002) studied a replication strategy to improve search efficiency in unstructured P2P networks in which target objects establish replication with the square root of the number of visits. This strategy can greatly improve the search efficiency for hot resources, but for rare objects search efficiency has not been improved much.

When searching for rare objects, the researchers put forth the strategy of object replication. When a peer joins a P2P network, it actively installs its objects' references at a set with $O(\gamma\sqrt{n})$ of randomly selected peers (Ferreira et al., 2005). This strategy can improve the searching efficiency for rare objects, but it can introduce a high communication cost. Moreover, it is hard to estimate the network size n, especially for dynamical P2P networks. Krishna et al. (Puttaswamy et al., 2008) uses Supernode-Constrained Random

Walk (SCRW) to search rare objects with two-hop index replication, and proposes three two-hop index replication strategies: full replication, square-root replication and constant replication. In SCRW, nodes always forward the query to one of randomly selected supernode neighbors. Although this strategy can improve the search efficiency for rare objects, it needs P2P network with supernode and will incur heavy overhead.

Shen (2008) puts forward an adaptive replication strategy. Compared to existing algorithms, this strategy requires the replication popularity to be relatively low. To achieve better search efficiency, the replication location is not random, but on the so-called network routing node. It can improve the search hit rate, and reduce the search latency as well. It also refers to the maintenance problem of replication, as it does not maintain replication according to heartbeat, but rather its access rate. With this idea as our base, we can use the access rate of objects to distinguish the value of an object. We can build less replication or no replication at all for those low values of rare objects to reduce the overhead caused by our strategies. As for object value, E-ARL (Mondal et al., 2010) defines the price of a data item as that recent access frequency of object d relates to its popularity. However, it is difficult for peers to obtain object popularity in a distributed environment.

In this paper, we propose a proactive replication strategy which can replicate rare objects at the peers of a P2P network, and also propose a rare object search algorithm. By doing this, we not only enhance search efficiency for rare objects, but also reduce unnecessary communication overhead for replication. In the next section, we will introduce our strategies in detail.

3. Proactive replication strategy

To improve search efficiency for rare objects, we must replicate these rare objects so that they can be successfully searched with limited number of hops. In this section, we will discuss object popularity, object value, proactive probe and a proactive replication (PR) strategy for replicating rare objects. At the end of this section, we will also discuss applicability of PR.

3.1. Object popularity

Object popularity is the proportion of the number of peers which own an object to the network size. Object popularity is very important since it directly determines the efficiency of search in unstructured P2P networks.

Let N represent the number of peers in an unstructured P2P network, and an object R in this network has r copies, that is there are r peers owning object R, then R's object popularity P in Eq. (1) is refined to

$$P = \frac{r}{N} \tag{1}$$

In an unstructured P2P network, the job of search algorithms is an exploratory attempt. Therefore, the popularity of target objects directly affects search efficiency. If an object R is searched only at a peer in P2P network, then the search hit probability is P. Using standard random walk (one way random walk) as the search algorithm, the maximum number of hops is k, and then the search hit probability $P_r(R)$ for object R is refined in

$$P_r(R) = kP = \frac{kr}{N} \tag{2}$$

To make the standard random walk search successfully for object R, we need to ensure that $P_r(R)$ is equal to 1. According to Eq. (3), it can be seen that object R can be searched successfully if the number r of object R in P2P network is up to a certain standard:

$$P_r(R) = 1 \Rightarrow r = \frac{N}{k} \tag{3}$$

In an unstructured P2P network, the object popularity directly affects search efficiency, especially for rare objects. Therefore, we can build replication based on the object popularity. How to obtain this object popularity will be discussed in the following section.

3.2. Object value

Although the object popularity can be used as a basis for replication, some rare objects in P2P networks are valueless. For example, the node owner accidentally shares a private image, but it has no value to other users in the network. Therefore, we do not need to replicate this valueless object. The overhead of replication can be reduced through the introduction of object value. Object R's value V_R at a time 't' in Eq. (4) is refined to

$$V_R(t+1) = \mu V_R(t) + \eta \frac{R_q(t)}{N_q(t)} \times 100 \tag{4}$$

where $0 \le \mu,\eta \le 1$, the larger the value of μ, the more important the historical requests to V_R, while the larger the value of η, the more important the recent requests to V_R. $Rq(t)$ is the number of requests received for object R and $Nq(t)$ is the total number of requests received by the node at a time 't'. We can take V_R as another basis for replication.

3.3. Proactive probe

From the analysis above, we know that we can improve search hit rate of object R through increasing its popularity. However, in a distributed environment, each node only has the partial information, so to obtain the object popularity is very difficult. Therefore, it is not certain which objects should be replicated and how many copies should be made. The existing replication strategies for rare objects are to replicate all objects to build the same number of replication, and although this can improve the hit rate of search algorithms, it also brings unnecessary communication overhead.

To address this issue, this paper presents a simple solution: When a node joins a P2P network, the node initiates rare object search, which will be discussed in the next section, to search its own objects, and the search can be defined by a success probe or a failure probe. In this paper, we call this strategy proactive probe. Those objects whose proactive probe failed are considered rare resources, and they need to be replicated. The node i's proactive probe function for object R $proactiveProbe(i,R)$ is refined in

$$proactiveProbe(i,R) = \begin{cases} 1 + \frac{TTL-hops}{TTL} & i \text{ successfully search } R \\ -1 & else \end{cases} \tag{5}$$

Where TTL is the maximum hops search algorithm allowed, and "$hops$" is search steps. In hit queries, the fewer search hops, the larger $proactiveProbe$ value. This design is to reduce the cost of probe, which will be introduced in the following paragraph.

The basic idea of proactive probe is this: If one fails to find the source, the same thing may happen to the others. Although we can know which objects are rare through proactive probe, but we do not know how many copies should be made for those objects which proactive probe is failed. In this paper, we take a simple solution: the peer establishes only one copy for those objects who fail to probe, and runs proactive probe regularly, not just in the beginning. This strategy is not to obtain an accurate popularity of an object, but to maintain the minimal object popularity with which searches can be done successfully. The replication method

is running a random walk for the rare object and replicating the object to the last peer. This is to distribute replications evenly and ensure the network load is balanced. The random approach is less sensitive to peer churn and heterogeneity as well (Fu et al., 2011). In order to ensure the minimum objects popularity with which object can be searched successfully, each node in the network must pass on a regular probe to maintain object popularity. The next probe time for object R is refined in

$$T(x) = g \times 2^l, \quad l = \begin{cases} x+y, & x+y>0 \\ 0 & else \end{cases} \tag{6}$$

Where g is a configurable interval value, x is the return value of *proactiveProbe*(i, R), y is the last l value and the initial value of y is 0. Because the return value of proactive probe is large for the objects with high popularity, probe intervals will be great for these objects. As a result, the cost of proactive probe is mainly caused by rare objects. Once the amount of replications of rare objects reaches at a certain stage, probe intervals for them will also be increased rapidly. Therefore, for an object, PR will no longer establish replications for it as soon as its replications reached a certain level. At the same time, proactive probe can capture the changes in the number of objects replications. Once the number of replications of an object reduces to below a certain number of values for some reasons, such as the peers with this object are offline, PR will soon restart to establish replications for it.

By defining the above probe time, we can effectively reduce the number of probes. In the meantime, the network maintains the minimum popularity of objects through the joint efforts of the network nodes. The detail cost analysis will be described in the experimental section.

3.4. Proactive replication

We use proactive probe to determine whether an object is rare, and take regular probe to maintain minimal object popularity which can ensure successful search with limited number of hops. Proactive probe only judges whether an object is rare through the rare object search algorithm, and then a copy is established at the terminated peer in another random walk for the object whose probe failed. This strategy can distribute object replication into the network in a balanced way because replication building uses random walk as a search algorithm. Our strategy improves search efficiency for rare objects, but will result in extra communication overhead. To decrease communication overhead, we can just install the reference for rare objects with large size, instead of copying them to the goal peers. Table 1 shows the instance of object routing table in the peer B. The objects with "own" type are the resources owned by the current peer, and those with "replicate" type are the replication reference for those rare objects from other peers. As we can see, peer B only installs replication reference to reduce communication overhead. In the query, the replicated peers will guide searches to the object owner. For example, to the query for "Zhuoma.mp3", peer B first looks up its object routing table, and then forwards the query to peer E. The request times in Table 1 are used to compute the object value. For very small files, such as file blocks in file sharing systems, we can directly copy them to the target peers to increase system robustness.

Although we use some strategies to reduce the overhead caused by replication, the proactive replication strategy still leads to additional communication overhead. The biggest overhead of proactive replication comes from the regular probe. To reduce this cost, object value can also be introduced in the regular probe of proactive replication. However, it is very difficult to compute the global visit frequency of an object R_q in the distributed environment. In this article, we only need the local request times of an object because our replication strategy is done in a distributed way, not global replication. Therefore, we can obtain R_q from the request times in the object routing table. Since the node's own object value V_R is 0, when it joins the network, in order to enhance the popularity of valuable objects, the node must be allowed to run a certain number of times before the factor of object value is introduced into the regular probe. When the node's running times is greater than a predefined threshold T_p, and the object value of V_R is less than a certain threshold V_t, the probe is not triggered, and its next probe time is set to $T(2)$. This strategy can decrease the overhead caused by replicating valueless objects. We still set the next probe time for the current valueless object is to avoid missing any value objects.

Node i's proactive replication algorithm for object R is shown in Algorithm 1.

Algorithm 1. Node i's proactive replication algorithm for object R.

Procedure *proactiveReplication*(i,R,t)
1: if $t > T_p$ and $V_R(t) < V_t$ **then**
2: After $T(2)$, call *proactiveReplication*(i, R, $t+T(2)$) again;
3: else
4: k=*proactiveProbe*(i,R);
5: **if** $k=-1$ **then**
6: Launch a Randwalk;
7: Replicate R in the last peer;
8: **end if**
9: After $T(k)$, call *proactiveReplication*(i, R, $t+T(k)$) again;
10: end if

3.5. Applicability of PR

In this section, we discuss on the applicability of the proposed PR strategy. In many P2P applications, rare resources are very valuable. For example, the peers of BitTorrent (Levin et al., 2008) give priority to download rare blocks. This strategy is to ensure that each file block is evenly distributed in BitTorrent systems, and avoid the last block problem (Bharambe et al., 2006). This is because the unavailability of the peers possessing rare objects will lead to blocking of network downloads. However, a peer in BitTorrent determines whether a block is rare simply based on those peers communicating with it, which has some limitations

Table 1
The instance of object routing table.

Object	Type	Peer	Current request times	Accumulated request times
Gongfu.avi	Own		3	30
Dingdang.wma	Own		2	10
…	…	…	…	…
TimeMachine.mov	Replicate	Peer A	1	3
Zhuoma.mp3	Replicate	Peer E	0	2
…	…	…	…	…

such as some rare blocks being ignored. Therefore, the last block problem has not been well resolved by existing researches. If BitTorrent takes on our PR strategy, the peers possessing rare blocks (i.e., rare objects) can proactively push rare blocks to the other peers to build replications, thus increasing the robustness of P2P systems. In P2P streaming systems, such as PPLive (Spoto et al., 2009), rare resources are much more important. Unlike that a BitTorrent client can download any of data blocks from other peers, a PPLive client can only download data blocks in a playback order. Therefore, PPLive cannot use the rare blocks priority strategy of BitTorrent to balance the load of P2P networks. If the network has a large number of rare blocks, it will cause severe viewing problems for the audiences. Even the quantity of rare blocks is small, it will also cause playback jitter. If the PR strategy is used in PPLive, replications can be built for rare blocks proactively. These replications created by PR cannot only be chosen as the download sources for other peers, but can also be used to playback by the host peers. Therefore, the proactive replication strategy has a wide range of application scenarios.

4. Rare object search algorithm

To reduce the overhead caused by proactive replication, we proposed a rare object search algorithm which needs less object popularity than others such as random walk. In this section, we will analyze search efficiency for different search strategies first, then discuss forward probability, and finally present a rare object search (ROS) algorithm.

4.1. Analysis of search efficiency

Random walk (RW) significantly reduces network overhead, but the search hit rates of it are decreased. In the same hit rate, taking high degree seeking (HDS) strategy requires a lower popularity of objects than random walk. To analyze search efficiency, we design some simulation experiments with different network sizes for RW and HDS. Figure 1 depicts a diagram for the search hit rate and object popularity using RW under different networks, and Fig. 2 describes search efficiency of HDS. Figures 1 and 2 show that the search hit rate of RW only reaches about 90% under 10% objects popularity, whereas HDS only requires 0.5% objects' popularity in the same search hit rate. At the same time, the correlation of the search hit rate and the objects popularity do not change with different network sizes.

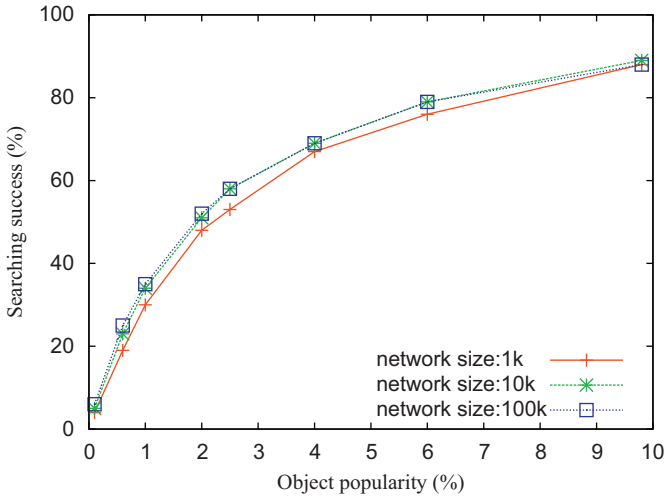

Fig. 1. Random walk hit rate with different object popularity.

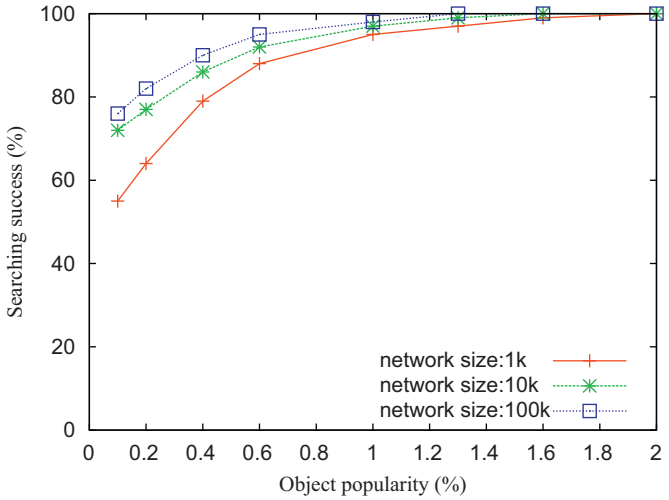

Fig. 2. High degree seeking hit rate with different object popularity.

As we discussed earlier, the higher popularity objects have, the higher hit rate the search has. In subsequent experiments, to improve the efficiency of simulation, we only use a medium-size network.

Since random walk is essentially a complete blind detection, it is ideal that the object popularity is relatively high, which is not suitable to our replication strategy characterized by maintaining minimum object popularity. The high degree seeking search only forwards search to neighbor nodes with relatively large degree, because these nodes have more neighbors and are more likely to contain the target resources. However, there is also the issue of information gathering when P2P networks use high degree seeking as search algorithm and high degree nodes may become a routing node, which results in network bottlenecks. Furthermore, there is the problem of fixed routing for high degree seeking. If we always choose high degree neighbor as the forwarding node, the search routing is fixed, which could lead to some nodes in the network not being covered. Therefore, we need a search algorithm that can utilize some characteristics of P2P networks such as the degree of nodes to improve efficiency. Moreover, this search algorithm should be able to avoid some problems arising from itself such as the heavy load of high degree nodes.

4.2. Forward probability

In order to have a more effectively search for rare objects, this paper proposes a search scheme which combines the degree and the object number of neighbors to obtain a forward decision. Experimental results show that this strategy has good search efficiency for rare objects. Each node in the network keeps the object number r and the degree k of its neighbors. j_r and j_k represent the object number and the degree of neighbor node j, respectively. When a node i forwards the query to neighbor j, the forwarding probability Pr_{ij} and Pk_{ij} based on the object number and the degree of neighbor, respectively, are defined by

$$Pr_{ij} = \frac{j_r}{\sum_{p \in K} p_r}, \quad Pk_{ij} = \frac{j_k}{\sum_{p \in K} p_k} \tag{7}$$

Where K is the set of i's neighbors, p is the node of the set K, and p_r, p_k represent the object number and the degree of the node p, respectively.

Based on the above analysis, the high degree seeking strategy can improve search efficiency, but it will bring the load problem. The probability Pk_{ij} defined here is similar to high degree seeking. In order to reduce the load of high degree nodes, we introduce the

probability Pr_{ij} based on the object number of nodes. A node cannot contain all the information of its neighbors' resources. For example, in some P2P applications, the node builds the index based on terms of its documents. The index may be very big, so the nodes cannot store it in each other. Therefore, the number of objects, which may be the number of terms in some applications, can be used as an important factor in guiding query route. Pr_{ij}, like Pk_{ij}, can lead the query to those power nodes which are more likely to contain the target objects. The experiments show that this strategy is available. Node i forwards query to neighbor j with the forward probability defined by

$$P_{ij} = \lambda Pr_{ij} + (1-\lambda) Pk_{ij}, \quad 0 < \lambda < 1 \tag{8}$$

Where λ is an adjustable parameter with the initial value of 0.5. The parameter λ can be automatically adjusted in the running system. For example, since search success is due to the factor of objects, we can increase the value of λ. The above forwarding strategy not only finds rare objects more effectively, but also enables search distribution. This strategy solves the problem of fixed routing in high degree forward and will be conducive to explore new objects.

4.3. ROS algorithm

In a query, the peer firstly computes the forward probability for each of its neighbors. The forward probabilities of the neighbors with high degree and more objects will be larger. Secondly, the peer forwards query to the neighbors according to the forward probability. This strategy can ensure the neighbors with larger forward probability are more likely to obtain forwarded query, and the neighbors with lower forward probability are also likely to obtain forwarded query. This can make query to locate the target as quickly as possible, and distributed query into the network to balance the search load of peers. In the rare object search strategy, the node i's forwarding algorithm is shown in Algorithm 2

Algorithm 2. Node i's forwarding algorithm.

Procedure Forward(i)
1: for each neighbor j in peer i **do**
2: Calculate Pr_{ij};
3: Calculate Pk_{ij};
4: end for
5: for each neighbor j in peer i **do**
6: Calculate Pij;
7: Forward query to neighbor j according P_{ij};
8: end for

5. Experiment methodology

In this section, we introduce our experiment methodology, including simulation methodology, network topology, objects and queries. While our simulation experiments do not capture all aspects of reality, we hope they capture the essential features needed to understand the qualitative differences.

5.1. Simulation methodology

P2P networks can be of a very large scale such as millions of nodes, which typically join and leave continuously. These properties are very challenging to deal with. Evaluating a new protocol in a real environment, especially in its early stages of development, is not feasible. To save time and increase efficiency, we use PeerSim (Jelasity et al., 2007) as the simulation-driven kernel. PeerSim has been developed to cope with the properties of P2P networks and thus able to reach extreme scalability and support dynamism. In addition, the simulator structure is based on components and makes it easy to quickly prototype a protocol, combining different pluggable building blocks, which are in fact Java objects. PeerSim is designed to encourage modular programming based on objects (building blocks). Every block is easily replaceable by another component implementing the same interface. In our experiments, we implement three PeerSim's interfaces: topo, protocol and control. Based on the topology data, the topo interface can form the topological structure we want. We implement our strategies and other researchers' strategies which are then compared in the protocol interface. We distribute objects (or files should be called) and queries into each peer through a control interface, and the interface can provide observation information for us as well.

In the experiments, simulation process is divided into a number of cycles. Each peer in P2P network will run its own protocol and control in each cycle. We can take cycle as time. In the simulation, we set time threshold T_p to 50 cycles, value threshold $V_t = 100$, $\mu = \eta = 0.5$, $g = 5$ cycles, $\lambda = 0.5$. In our preliminary research (Gao et al., 2010), we do not change the value of these parameters for simplicity. In this extended simulation, in order to bound the performance improvement of our scheme, we try to use different parameters to get a better performance.

5.2. Network topology

By the network topology, we mean the graph formed by the P2P overlay network. Each P2P member has a certain number of "neighbors" and the set of neighbor connections forms the P2P overlay network. In this paper when we refer to the "network" we are referring to P2P networks, not the underlying Internet. For simplicity, we assume that P2P network topology does not change during the simulation of our algorithms for most of experimental scenarios. The initial topology is generated by Brite (Medina et al., 2001) which is a topology generation tool that provides the option of generating topologies based on the AS Model. Using Brite, we generate an overlay topology with 10,000 nodes. In order to better represent real-world P2P networks, the node placement of Brite is set to "Heavy Tailed", which makes the topology generated by this strategy to have power-law distribution. The topology data is loaded into PeerSim through the interface topo which is implemented by us. Node degree information of the topology graph is shown in Fig. 3. In this graph, we use

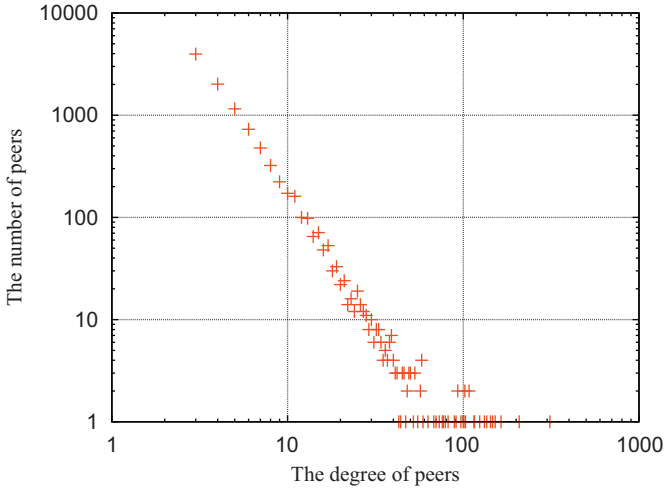

Fig. 3. Distribution of node degrees in topology.

a log scale to clearly describe degree distribution. As we can see, most peers have low degrees, and only a few peers have high degrees.

5.3. Objects and queries

Studies have shown that Napster, Gnutella and Web queries tend to follow Zipf-like distributions (Sripanidkulchai, 2001). In order to reveal the real environment, the object popularity in our experiments follows a Zipf-like distribution. This distribution states that some personal content is highly popular and the rest has more or less the same low popularity. In Eq. (9) the Zipf-like probability mass function (Breslau et al., 1999) is provided, where C denotes the number of personal content items and α is the exponent characterizing the distribution:

$$P_{Zipf\text{-}like}(x) = \frac{x^{-\alpha}}{\sum_{j=1}^{C} j^{-\alpha}} \tag{9}$$

$P_{Zipf\text{-}like}(x)$ determines the probability that a personal content object having rank x is requested, where $x \in \{1, \ldots, C\}$. Backx et al. (2002) show, with a number of practical experiments using popular P2P file sharing applications, that α is usually between 0.6 and 0.8.

We assume each object i is replicated on r_i nodes, and the total number of objects stored in the network is S_r:

$$\sum_{i=1}^{m} r_i = S_r \tag{10}$$

To assign objects to each of peers in the network, we generate 5,000 distinct terms and assign each term a frequency according to Eq. (9). We write the set of these terms as S_T. In this paper, we use "term" and "object" interchangeably. The terms in S_T are distributed into the network based on their frequency, and the terms with high-frequency will have more copies. We allocate 20–200 terms for each peer randomly. The total number of terms' copies of the network should be equal to S_r. As for queries, based on the terms' frequency, each peer obtains query keywords from S_T randomly. That is the terms with high-frequency are more likely to be selected as the query keywords. The queries generated by this strategy are more like those from real networks. The query distribution is shown in Fig. 4. As we can see, the more queries there are, the fewer the number of corresponding terms. This implies that a term having a higher frequency is more popular (more queries).

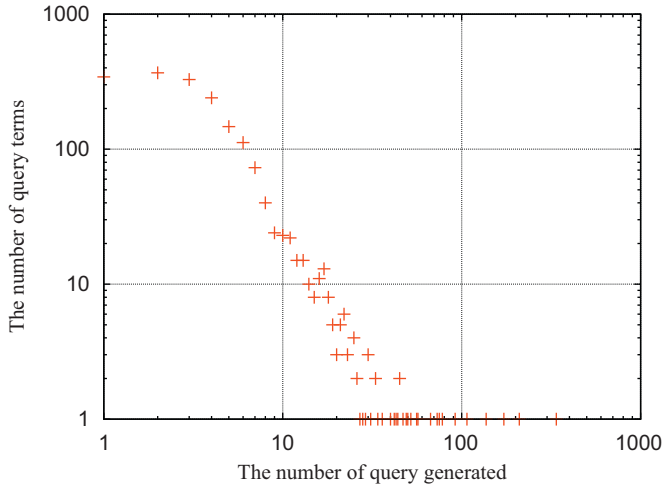

Fig. 4. Query distribution.

6. Performance evaluation

In this section, we analyze the efficiency of proactive replication (PR) strategy and rare object search (ROS) algorithm presented in this paper through the experiments.

6.1. Effectiveness of proactive replication

To evaluate the efficiency of PR strategy, we run ROS algorithm proposed in this paper in the network which PR strategy is applied to as well as in the network without any replication strategies. Figure 5 depicts the correlation of object popularity and search hit rate in the networks in which ROS is run. Experiments show that search efficiency of ROS has been greatly improved as PR strategy is applied in P2P network. The object popularity of the horizontal axis in Fig. 5 represents the system initialization popularity of the object itself, rather than the object popularity after using PR strategy. This is to visually show search efficiency of rare objects. ROS algorithm forwards queries to those power nodes, so it has high search efficiency with PR strategy which provides the necessary popularity of objects for successful search. Therefore, our PR strategy is available to improve search efficiency for rare objects according to the experimental results.

Due to object probing, PR strategy produces some additional communication overhead compared to other algorithms. In order to analyze the overhead, we collect communication messages of object probing and replication in P2P networks with PR strategy. The average object number of each node is 10 in the experiments. We take the replication strategy with probability (WP) proposed by Ronaldo (Ferreira et al., 2005) to conduct a comparative analysis. WP replicates $\gamma\sqrt{n}$ copies for each object of node, n is the number of network nodes. WP has the query success rate at 85% when $\gamma = 1$. The success rate of ROS reaches 95% after using PR strategy. To be fair, we take γ as 1 in the experiments. Table 2 shows the average amount of communication messages processed by the two kinds of replication strategies in each node in the simulations. PR strategy proposed in this paper reduces the

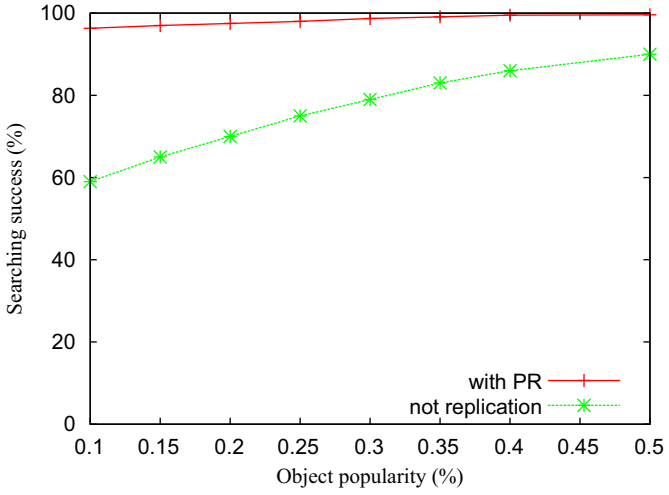

Fig. 5. ROS hit rate with pr and not replication.

Table 2
Amount of message in different replication strategies.

	WP	PR
Average message processed in each node	2162	1351
Factor	1	1.6

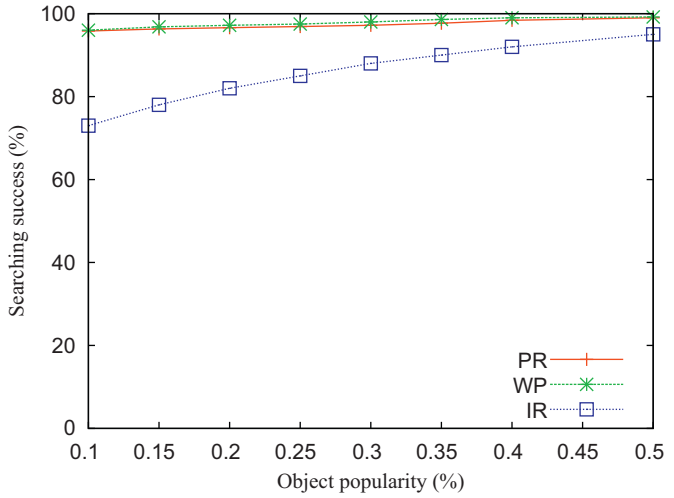

Fig. 6. HDS hit rate with different replication strategies.

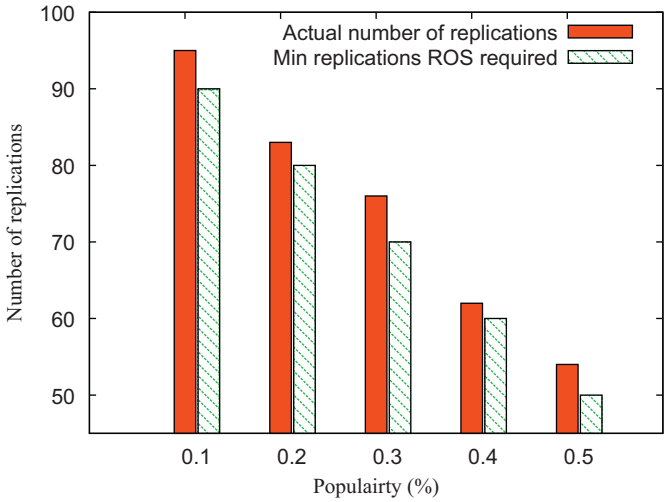

Fig. 7. Replication level of proactive replication.

communication traffic by a factor of 1.6 compared to the existing WP algorithms. Obviously, the traffic of both PR and WP increases with the increase in simulation cycles. However, if the object probe cycle interval is properly handled, we can maintain low communication overhead while achieving a reasonable popularity of objects.

To analyze the performance of PR, we introduce another replication strategy: two-hop index replication (IR) similar to the strategy proposed in Puttaswamy et al. (2008). In IR, each node sends its index to all of its two-hop neighbors in its routing table. Figure 6 shows the efficiency of high degree seeking (HDS) strategies with three different replication strategies. As we can see, IR is the least effective, and the efficiencies of both PR and WP for rare objects are similar. However, PR strategy generates less communication overhead, thus the proactive replication strategy PR is superior to the replication strategy with probability WP in overall performance.

6.2. Replication level

In this set of experiments, we measure the replication level of PR. When the object popularity is equal to 1%, we find that ROS can reach 99% hit rate through previous experiments. We call this popularity (1%) the minimum popularity required by ROS for successful searching. Therefore, in a P2P network with 10,000 peers, if an object only has 10 copies (a popularity of 0.1%), PR should establish another 90 replications for the object for searching it successfully. We design some experiments to collect the actual number of replications for the objects with different popularity, and compare it with the theoretical minimum amount of replications. The results are shown in Fig. 7. The *X*-axis represents the objects with different popularities, and the *Y*-axis shows the average amount of replications established by PR for them. As it can be seen from Fig. 7, the number of replications created by PR is just a little bit than the theoretical minimum value. The objects with popularity 0.3% are the worst, but PR only establishes 8.5% more replications for them. In general, PR only creates 5.7% extra replications compared to the minimum number of replications. Therefore, there is a point in which PR is no longer to further replicate the objects in P2P networks. The extra 5.7% is perhaps the point, which is derived from the experiments. In PR, whether to create a replication or not depends on the result of proactive probe. When an object's popularity is greater than the minimum popularity required by ROS, proactive probe for it is usually successful. As a result, PR would not create too many extra replications.

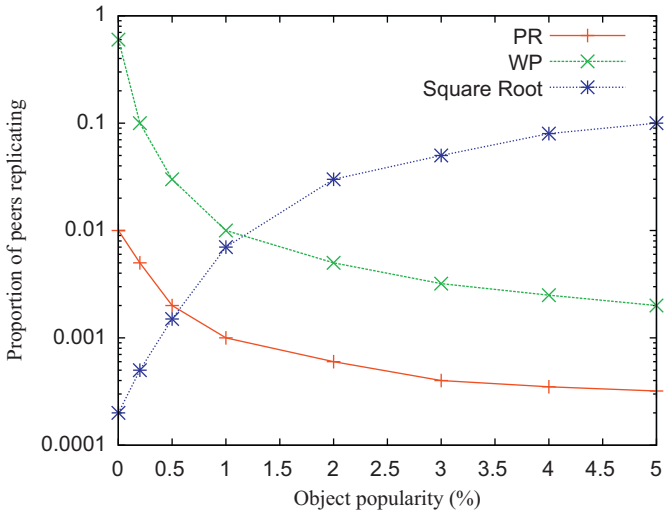

Fig. 8. Percent of peers installing references.

6.3. Replication overhead

Another parameter that we investigate in our scheme is the percentage of peers owning an object who install references to an object in the network. In this scenario, the popularity of an object is varied from 0.001% to 5%. The experiment is conducted for different replication strategies. We compare PR with WP and Square Root presented in Cohen and Shenker (2002). Figure 8 shows, on a logarithmic scale, the percentage of object owners who install references to an object for different replication strategies. As for PR and WP, when almost 5% of the peers own a copy of an object, only a very small percentage (0.0003–0.002) of the peers owning this object install references to this object. PR takes ROS as the search algorithm which needs less object popularity to locate objects. Therefore, the overhead of PR is less than that of WP. Square Root has completely different results. When an object has low popularity, the percentage of the peers who installed references to the object is small. However, there are many peers replicating hot objects (the objects with high

popularity). This is because square root establishes replication with the square root of the number of visits for the objects. Therefore, this strategy can only improve the search efficiency of hot resources, but does not fit for rare objects.

In order to obtain a better replication performance, we adjust the parameter of interval value g defined in Eq. (6). Figure 9 shows the percentage of the peers that install references to an object for a different parameter g. We set a logarithmic scale for y-axis as previous experiments. As we can see, the overhead of PR will decrease with the increase of the parameter g. Although we can obtain a better replication performance in overhead through increasing the parameter g, this will lead to a decline in search efficiency. When the interval value g is set to 20 cycles, shown in Fig. 9, the percentage of peers that establish for the objects with low popularity will obviously decrease. This will be difficult to ensure that the rare objects have the minimal popularity with which searches can be done successfully. Therefore, the parameter g should be carefully set to obtain the better performance of replication in overhead and search.

Figure 10 shows the percentage of the peers that install references to an object for different parameters μ and η. We set a logarithmic scale for the y-axis as well. We found that the results of three parameter setting are very similar. A reasonable explanation is that the object value has little effect on replication in our experiments because query keywords are selected randomly from the set of terms S_t. We trust that the adjustment of parameters μ and η will obtain a better performance in real networks.

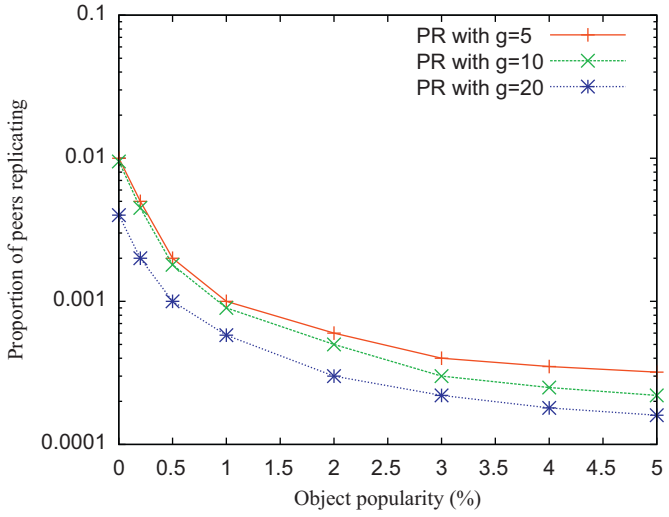

Fig. 9. Percent of peers installing references with different parameter g.

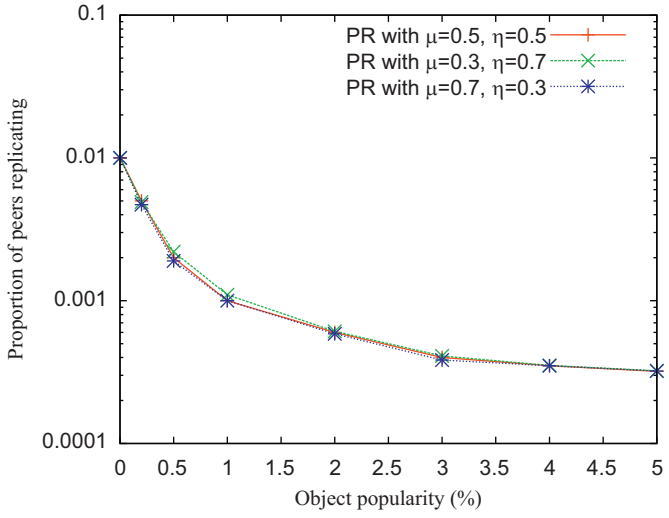

Fig. 10. Percent of peers installing references with different parameter μ and η.

6.4. Analysis of peer collaboration

In unstructured P2P networks, we do not know which objects should be replicated and how many replications should be established because there are no central servers. We solve these problems through peer collaboration. Proactive probe is the signal of collaboration for the peers which only know local information. A peer can know whether an object is rare through proactive probe. Once proactive probe for an object fails, the object is considered to be rare and should be replicated. For an object, multiple peers in the network maintain its replications collaboratively through proactive probe. Once proactive probe for an object is successful, the peers owning the object no longer create replications for it. Therefore, peer collaboration coordinated by the proactive probe strategy is important, and such collaborations maintain a reasonable number of object copies.

We designed an experiment to evaluate the cumulative distribution function (CDF) for peer collaboration. We count all the peers who perform replication operation. Figure 11 shows the results for five different popularities. As we can see, the objects with popularity 0.1% and 0.2% show the best load balancing result. Top 30% most contribution peers account for only about 45% of total contributions. For the objects with popularity 0.3%, 0.4% and 0.5%, it is approximately 55%. Overall, all popularities show better load balancing results. Even for the least effective objects with popularity 0.5%, only 12% of all peers do not contribute to replication. On average, if an object needs PR to establish replications, nearly 95% of the peers owning the object will contribute their efforts. This shows that PR establishes replications through peer collaboration.

6.5. Effective of rare object search

We can see, from Figs. 5 and 6, search efficiency of ROS algorithm presented in this paper is similar to HDS strategy for searching rare objects. However, HDS forward routing is fixed, which can easily lead to information gathering. ROS uses the probability of forwarding, which can improve the problem of load balance posed by queries. The system with HDS strategy may lead

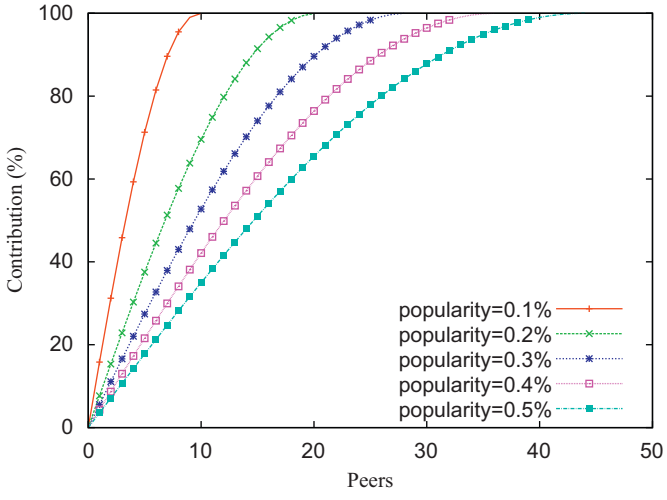

Fig. 11. CDF of peer collaboration.

the high degree nodes to become a bottleneck. To alleviate the load of high degree nodes, we can first run random walk and HDS after search failure to improve HDS. We call this strategy improved HDS (IHDS). Although IHDS could solve the problem of load, it increases latency for rare objects, and brings additional communication overhead as well. Therefore, taking ROS as search scheme cannot only ease network traffic load, but also lessen search delay of rare objects. Figure 12 describes search delay of ROS and IHDS for rare objects in P2P network with PR strategy. The vertical axis shows the maximum number of hops for successful search to target objects. IHDS first uses random walk to search an object. However, random walk generally fails with the specified TTL value for the objects with low popularity. Therefore, IHDS has high latency overhead.

In order to carry out the load analysis, we design an experiment to discuss load balance of nodes with different degrees as follows: The network topology is generated by Brite (Medina et al., 2001), the minimum degree of nodes is 3, the maximum degree is 206, and the degree distribution of nodes obeys power-law distribution. Figure 13 is the simulated results of ROS, HDS and SCRW (Supernode-Constrained Random Walk) proposed in Puttaswamy et al. (2008) which nodes always forward the query to one of its randomly selected supernode neighbors. As we can see, the load of ROS is balanced, and the load of the node with the greatest degree is lower than 100%. While the high load of the HDS and SCRW is mainly concentrated at the high degree nodes, the utilization of the nodes with small degree is relatively lower. Therefore, ROS algorithm presented in this paper not only has a better search efficiency, but also improves the load balance of nodes compared to the existing algorithms.

6.6. Performance in churn

PR strategy can withstand the high rate of node dynamics. Churn arises from continued and rapid arrival and failure (or departure) of a large number of peers in P2P networks. We evaluated the efficiency of PR strategy in unstructured P2P networks in churn. Experiment results verified the resilience of PR in churn. In the simulation, node join and voluntary departure are modeled by a Poisson process with a mean rate of R, which ranges from 0.05 to 0.5. A rate of $R=0.05$ corresponds to one node joining and leaving every 20 s on average. Figure 14 plots the average lookup path length versus the node join/leave rate. As we can see, Search delay of PR is 4.45 hops at $R=0.5$, however, WP and IR exceed this value at $R=0.05$. This is because regular probes of peers with PR strategy ensure the objects popularity, but WP and IR have no mechanisms to cope with network churn.

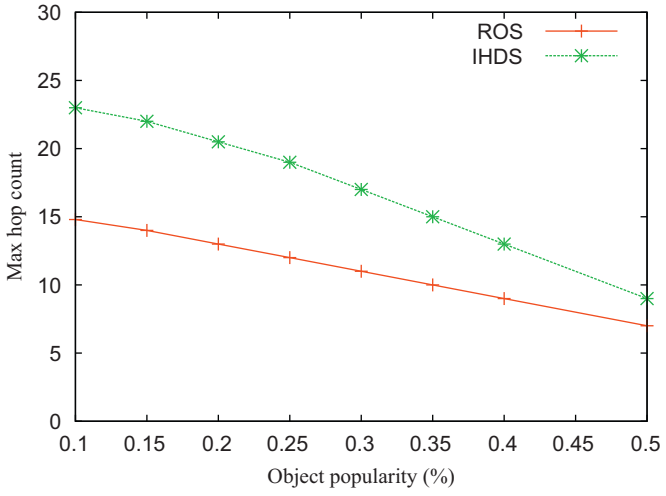

Fig. 12. Search delay of ROS.

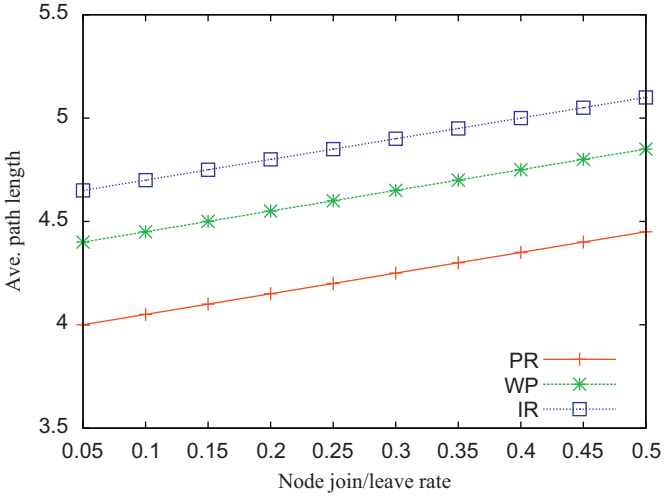

Fig. 14. Performance of different replication strategies in network churn.

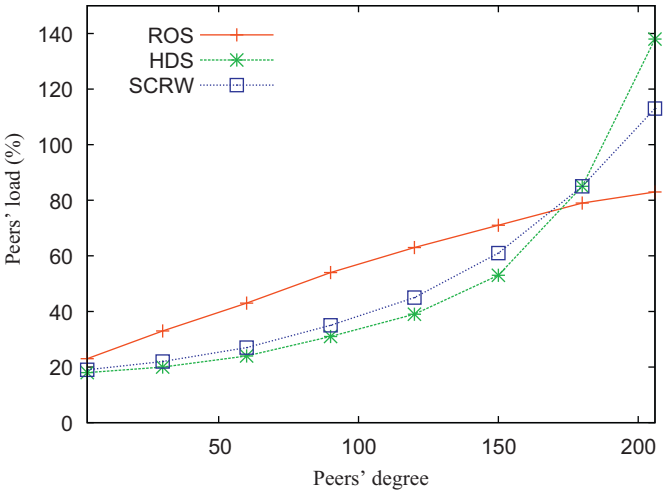

Fig. 13. Peers' load with ROS.

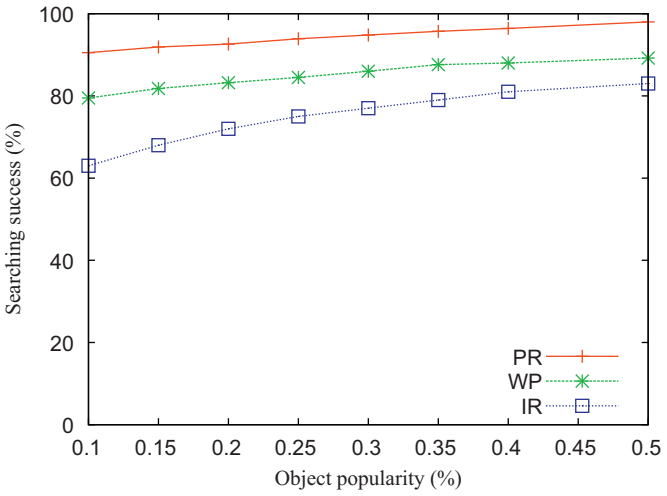

Fig. 15. Performance of different replication strategies in content churn.

Churn not only appears in the network, but also exists in the content. We take another experiment to analyze the dynamicity of the content (popularity changing over time). In the simulation, the set of search objects varies with time periods, that is, the popular objects in a certain time period will become unpopular in another time period. We can adjust μ in Eq. (4) to increase the replication of popular rare objects. That is PR can detect changes in popularity of those objects, and quickly start replication operation for them. However, the other replication strategies cannot adapt to such changes. PR strategy ensures high resistance to content churn and this is evident from Fig. 15 in which the system generates a good success rate. PR can guarantee an average search success rate of 95% for the objects with popularity below 0.5%. As for WP and IR, it is 85% and 75%, respectively.

7. Conclusion

This paper presents a new proactive replication strategy for rare objects, as well as proposes a rare object search algorithm which needs a smaller number of replications. Our contributions are: First, based on the analysis of object popularity, we propose a proactive probe strategy and an object value scheme, which are used in proactive replication. Second, we propose forwarding probability based on the object number, and this probability can be used to improve high degree seeking. Through the proposed algorithms, we can greatly improve search efficiency for rare objects, and keep communication cost associated with search relatively low. Experiments show that there is a significant improvement in efficiency for rare objects after using proactive replication strategy, and communication overhead brought by search is lower than other replication strategies. We design comparative experiments with other search algorithms, and the results show that the search algorithm in this article has better efficiency for rare objects, and ensures load balance of search. In order to reduce unnecessary communication overhead, the replication strategy proposed in this paper only establishes the least amount of replication, which can decide whether to create a replication through the probe of objects. This will result in the communication overhead of the probe. However, experiments have shown that this type of overhead is less than what is generated by other replication strategies. As a result, it will be interesting to see if the communication overhead brought by probing can be further reduced by fine-tuning the interval between detection cycles. The dynamic adjustment for the proportion of degree and objects in the rare object search algorithm is also worth exploring. The future work will focus on the cycle of object probing and the optimization of the search strategy. At the same time, we also plan to use P2P trace in simulation experiments to gain more insightful connection and valuable information.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grants 60873225, 60773191, 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403, Natural Science Foundation of Hubei Province under Grant 2009CDB298, Wuhan Youth Science and Technology Chenguang Program under Grant 200950431171, Open Foundation of State Key Laboratory of Software Engineering under Grant SKLSE20080718, and Innovation Fund of Huazhong University of Science and Technology under Grants 2010MS068 and Q2009021.

References

Adamic LA, Lukose RM, Huberman BA. Local search in unstructured networks. In: Bornholdt, Schuster, editors. Handbook of graphs and networks: from the genome to the internet; 2003.

Adamic LA, Lukose RM, Puniyani AR, Huberman BA. Search in power-law networks. Phys. Rev. E 2001;64:46135.

Backx P, Wauters T, Dhoedt B, Demeester P. A comparison of peer-to-peer architectures. In: Proceedings of Eurescom 2002 powerful networks for profitable services; 2002. p. 1–8.

Bharambe AR, Herley C, Padmanabhan VN. Analyzing and improving a bittorrent networks performance mechanisms. In: INFOCOM. IEEE; 2006.

Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web caching and Zipf-like distributions: evidence and implications. In: Proceedings annual joint conference of the IEEE computer and communications societies; 1999. p. 126–34.

Chawathe Y, Ratnasamy S, Breslau L, Lanham N, Shenker S. Making Gnutella-like p2p systems scalable. In: Proceedings of the 2002 annual conference of the Special Interest Group on Data Communication; 2003. p. 407–18.

Clarke I, Sandberg O, Wiley B, Hong TW. Freenet: a distributed anonymous information storage and retrieval system. In: International workshop on design issues in anonymity and unobservability; 2000. p. 311–20.

Cohen E, Shenker S. Replication strategies in unstructured peer-to-peer networks. In: Proceedings of the 2002 annual conference of the Special Interest Group on Data Communication; 2002. p. 177–90.

Ferreira RA, Ramanathan MK, Awan A, Grama A, Jaganathan S. Search with probabilistic guarantees in unstructured peer-to peer networks. In: Proceedings of the fifth international conference on peer-to peer computing; 2005. p. 165–72.

Filali I, Huet F. Dynamic TTL-based search in unstructured peer-to-peer networks. In: CCGRID. IEEE; 2010. p. 438–47.

Frankel J, Pepper T. 2000. Gnutella. Available online at: 〈http://www.gnutella.com〉.

Fu S, Xu C-Z, Shen H. Randomized load balancing strategies with churn resilience in peer-to-peer networks. Journal of Network and Computer Applications 2011;34(1):252–61.

Gao G, Li R, Wen K, Gu X, Lu Z. Proactive replication and search for rare objects in unstructured peer-to-peer networks. In: Proceedings of the 11th international conference on web-age information management; 2010. p. 74–85.

Gkantsidis C, Mihail M, Saberi A. Random walks in peer-to-peer networks. In: Proceedings of the 23rd annual joint conference of the IEEE computer and communications societies; 2004.

Gkantsidis C, Mihail M, Saberi A. Hybrid search schemes for unstructured peer-to-peer networks. In: Proceedings of the 24th annual joint conference of the IEEE computer and communications societies; 2005. p. 1526–37.

Gupta I, Birman K, Linga P, Demers A, Renesse RV. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. In: Proceedings of the third international conference on peer-to-peer computing; 2003. p. 149–60.

Heinla A, Kasesalu P, Tallinn J. Kazaa. Available online at: 〈http://www.kazaa.com/〉; 2001.

Jelasity M, Montresor A, Jesi GP, Voulgaris S. The Peersim simulator. Available online at: 〈http://peersim.sourceforge.net/〉; 2007.

Kong JS, Bridgewater JSA, Roychowdhury VP. A general framework for scalability and performance analysis of DHT routing systems. In: Proceedings of the 2006 international conference on dependable systems and networks; 2006. p. 343–54.

Levin D, LaCurts K, Spring N, Bhattacharjee B. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In: Bahl V, Wetherall D, Savage S, Stoica I, editors. Proceedings of the ACM SIGCOMM 2008 conference on applications, technologies, architectures, and protocols for computer communications, Seattle, WA, USA, 17–22 AugustACM; 2008. p. 234–54.

Liao X, Zhang F, Jin H, Yu L. iDARE: proactive data replication mechanism for P2P voD system. In: CIT IEEE Computer Society; 2010. p. 682–9.

Loo BT, Hellerstein JM, Huebsch R, Shenker S, Stoica I. Enhancing p2p file-sharing with an internet-scale query processor. In: Proceedings of the 13th international conference on very large data bases; 2004. p. 432–43.

Lv Q, Cao P, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th annual ACM international conference on supercomputing; 2002. p. 84–95.

Medina A, Lakhina A, Matta I, Byers JW. BRITE: an approach to universal topology generation. In: MASCOTS IEEE Computer Society; 2001. p. 346.

Mondal A, Madria SK, Kitsuregawa M. E-ARL: an economic incentive scheme for adaptive revenue-load-based dynamic replication of data in mobile-P2P networks. Distributed and Parallel Databases 2010;28(1):1–31.

Puttaswamy K, Sala A, Zhao BY. Searching for rare objects using index replication. In: Proceedings of the 27th annual joint conference of the IEEE computer and communications societies; 2008. p. 1723–31.

Qiao Y, Bustamante FE. Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use. In: Proceedings of the 2006 USENIX annual technical conference; 2006. p. 341–55.

Ratnasamy S, Francis P, Handley M, Karp RM, Shenker S. A scalable content-addressable network. In: Proceedings of the 2001 annual conference of the Special Interest Group on Data Communication; 2001. p. 161–72.

Ripeanu M. Peer-to-peer architecture case study: Gnutella network. In: Proceedings of the first international conference on peer-to-peer computing; 2001. p. 99–100.

Sarshar N, Roychowdhury VP. An end-to-end solution to scalable unstructured p2p networking. In: Proceedings of the seventh IEEE international conference on peer-to-peer computing; 2007. p. 123–31.

Shen H. Ead: an efficient and adaptive decentralized file replication algorithm in p2p file sharing systems. In: Proceedings of the eighth international conference on peer-to-peer computing; 2008. p. 99–108.

Shen H, Zhu Y. A proactive low-overhead file replication scheme for structured P2P content delivery networks. Journal of Parallel and Distributed Computing 2009;69(5):429–40.

Spoto S, Gaeta R, Grangetto M. Analysis of PPLive through active and passive measurements. In: IPDPS. IEEE; 2009. p. 1–7.

Sripanidkulchai K. 2001. The popularity of Gnutella queries and its implications on scalability. In: O'Reilly's 〈www.openp2p.com〉.

Stoica I, Morris R, Karger DR, Kaashoek MF, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 annual conference of the Special Interest Group on Data Communication; 2001. p. 149–60.

Xu M, Zhou S, Guan J, Hu X. A path-traceable query routing mechanism for search in unstructured peer-to-peer networks. Journal on Network and Computer Applications 2010;33(2):115–27.

Yin Z, jin H, Zhang C, Yuan Q, Zhao C. Adaptive query-caching in peer-to-peer systems. In: Proceedings of the second IFIP international conference on network and parallel computing; 2005. p. 97–104.

Zhang R, Hu YC. Assisted peer-to-peer search with partial indexing. IEEE Transactions on Parallel Distribution System 2007;18(8):1146–58.

Zhao BY, Kubiatowicz J, Joseph AD. Tapestry: a fault-tolerant wide-area application infrastructure. Computer Communication Review 2002;32(1):81.